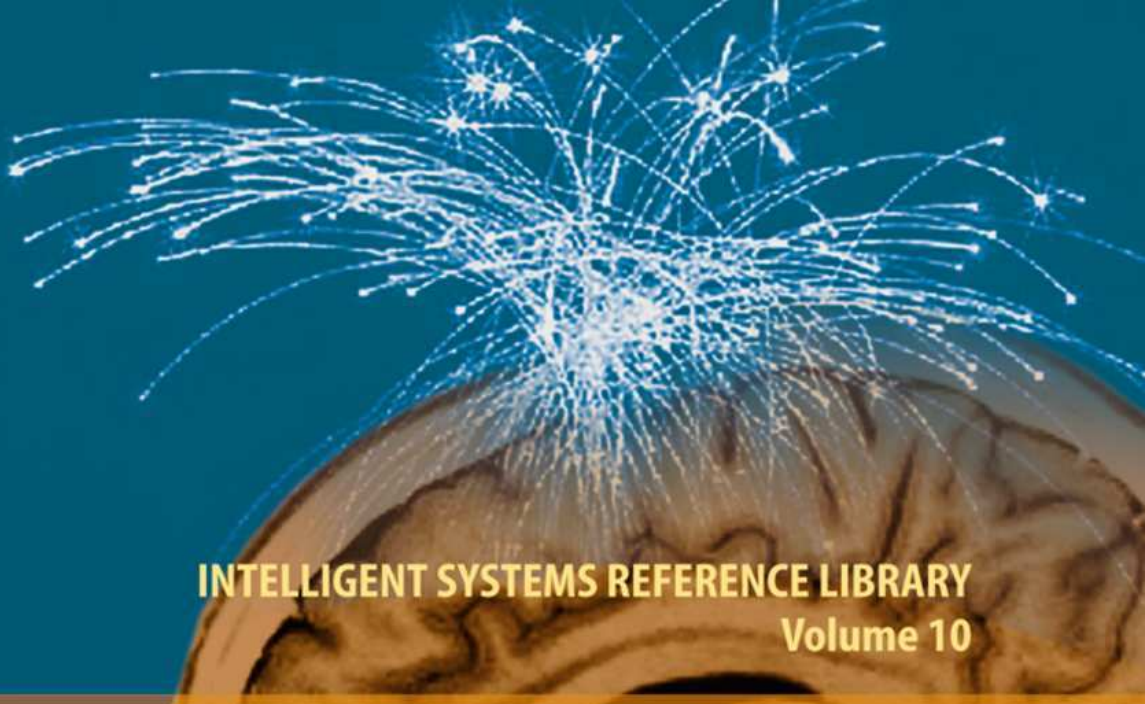


INTELLIGENT SYSTEMS REFERENCE LIBRARY
Volume 10

Andreas Tolk
Lakhmi C. Jain (Eds.)

Intelligence-Based Systems Engineering

Springer

Andreas Tolk and Lakhmi C. Jain

Intelligence-Based Systems Engineering

Intelligent Systems Reference Library, Volume 10

Further volumes of this series can be found on our homepage: springer.com

Vol. 1. Christine L. Mumford and Lakhmi C. Jain (Eds.)
Computational Intelligence: Collaboration, Fusion and Emergence, 2009
ISBN 978-3-642-01798-8

Vol. 2. Yuehui Chen and Ajith Abraham
Tree-Structure Based Hybrid Computational Intelligence, 2009
ISBN 978-3-642-04738-1

Vol. 3. Anthony Finn and Steve Scheding
Developments and Challenges for Autonomous Unmanned Vehicles, 2010
ISBN 978-3-642-10703-0

Vol. 4. Lakhmi C. Jain and Chee Peng Lim (Eds.)
Handbook on Decision Making: Techniques and Applications, 2010
ISBN 978-3-642-13638-2

Vol. 5. George A. Anastassiou
Intelligent Mathematics: Computational Analysis, 2010
ISBN 978-3-642-17097-3

Vol. 6. Ludmila Dymowa
Soft Computing in Economics and Finance, 2011
ISBN 978-3-642-17718-7

Vol. 7. Gerasimos G. Rigatos
Modelling and Control for Intelligent Industrial Systems, 2011
ISBN 978-3-642-17874-0

Vol. 8. Edward H.Y. Lim, James N.K. Liu, and Raymond S.T. Lee
Knowledge Seeker – Ontology Modelling for Information Search and Management, 2011
ISBN 978-3-642-17915-0

Vol. 9. Menahem Friedman and Abraham Kandel
Calculus Light, 2011
ISBN 978-3-642-17847-4

Vol. 10. Andreas Tolk and Lakhmi C. Jain
Intelligence-Based Systems Engineering, 2011
ISBN 978-3-642-17930-3

Andreas Tolk and Lakhmi C. Jain

Intelligence-Based Systems Engineering

Springer

Prof. Andreas Tolk
Engineering Management & Systems Engineering
242B Kaufman Hall
Old Dominion University
Norfolk, VA 23529
USA
E-mail: atolk@odu.edu

Prof. Lakhmi C. Jain
School of Electrical and Information Engineering
University of South Australia
Adelaide
Mawson Lakes Campus
South Australia SA 5095
Australia
E-mail: Lakhmi.jain@unisa.edu.au

ISBN 978-3-642-17930-3 e-ISBN 978-3-642-17931-0

DOI 10.1007/978-3-642-17931-0

Intelligent Systems Reference Library ISSN 1868-4394

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

The International Council on Systems Engineering (INCOSE) defines Systems Engineering as an interdisciplinary approach and means to develop successful systems. It focuses on defining the customers needs and requirements early in the development cycle. It then documents the requirements. It then proceeds with the design synthesis and system validation and develops an overview of the complete problem which involves Manufacturing, Operations, Cost & Scheduling. The Performance, Training & Support, Testing, and Disposal are then developed. Systems Engineering integrates all of the disciplines and specialty groups into a joint team effort to form a structured development process which proceeds from the concept stage of production to full final operation. The full Systems Engineering operation considers both the business and the technical needs of all customers. The goal is to provide a quality product that meets the user needs and hopefully without unwanted surprises in the completed item.

In the present time, these activities and processes are increasingly supported by means of Information Technology (IT). Support using IT always leads to the question of how much such processes can be either automated or semi-automated. In other words: is it possible to increase the quality of systems by using intelligence-based systems engineering. The intention of this book is to answer the questions such as what emerging methods and solutions are able to use intelligence-based systems engineering, what current solutions already exist, what theoretic constraints are known, and other questions ranging between theory and practice. The chapters contain contributions from conferences, research, PhD theses, and the experience of the experts in this area. In this book, we establish a research agenda and begin to fill the gaps in this body of knowledge.

We hope to gain the support of practitioners and scholars by this volume. It is also hoped to help researchers identify domains of interest and to develop systems engineering to an even higher level.

Andreas Tolk
USA

Lakhmi C. Jain
Australia

Contents

Chapter 1

Towards Intelligence-Based Systems Engineering and System of Systems Engineering 1
Andreas Tolk, Kevin MacG. Adams, Charles B. Keating
1 Introduction 1
2 Intelligence-Based Systems 2
2.1 Characteristics of Intelligence-Based Systems 2
2.2 How to Capture Intelligence 4
3 Systems Engineering 6
3.1 Traditional Systems Engineering 7
3.2 System of Systems 8
3.3 System of Systems Engineering 10
3.4 System of Systems Engineering Methodology 11
3.5 Intelligence-Based Systems Engineering 16
4 Contributions to These Topics within This Volume 18
References 20

Chapter 2

Future Directions for Semantic Systems 23
John F. Sowa
1 The Knowledge Acquisition Bottleneck 23
2 Natural Language Processing 24
3 Reasoning and Problem Solving 27
4 Semantic Web 30
5 Language Analysis and Reasoning 35
6 Integrating Semantic Systems 43
References 45

Chapter 3

Defining and Validating Semantic Machine to Machine Interoperability 49
Claudia Szabo, Saikou Y. Diallo
1 Introduction 49
2 State of the Art in Interoperability 50

2.1 Semantics of Data for a Machine 53
2.2 Formal Representation of Data for a Machine 55
2.3 Semantic Machine to Machine Interoperability 58
3 Formal Validation of Interoperable Federations 63
3.1 Knowledge Representation 66
3.2 Formal Validation of Model Execution 68
3.3 Reference Model 68
3.4 Formal Validation Process 69
4 Summary and Recommendations 72
References 72

Chapter 4

An Approach to Knowledge Integration Applied to a Configuration Problem 75
Maria Vargas-Vera, Miklos Nagy, Dietmar Jannach
1 Introduction 75
2 Related Work 77
2.1 Expert Systems - Knowledge Bases 77
2.2 Ontologies View 78
2.3 Databases 80
2.4 Knowledge Management 80
3 Scenario 81
3.1 Constraint Satisfaction Problem (CSP) 82
3.2 Case Study: Computer Configuration Problem 83
3.3 Constraint Graph 84
4 Mapping Process 84
5 Knowledge Integration Framework 92
5.1 Algorithms for Detecting and Correcting Overlappings 94
6 Evaluation 97
6.1 Mapping Quality 99
6.2 Configuration Quality 100
7 Conclusions 102
References 103

Chapter 5

Simulation-Based Systems Design in Multi-actor Environments 107
Michele Fumarola, Mamadou D. Seck, Alexander Verbraeck
1 Introduction 107
1.1 Outline of the Chapter 108
2 Designing Systems 108
3 Systems Approaches 111
3.1 Systems Simulation in Design 112
3.2 Soft Systems Methodology 113

4 Designing a Multimethodological Approach 118
4.1 Component Based Modeling 118
4.2 Different Levels of Abstraction 119
4.3 Structing Alternatives 121
4.4 Participatory Design 123
5 Conclusion 123
References 124

Chapter 6

Distributed Simulation Using RESTful Interoperability Simulation Environment (RISE) Middleware 129
Khaldoon Al-Zoubi, Gabriel Wainer
1 Introduction 129
2 Background on Distributed Simulation 132
3 RISE Middleware API 136
4 RISE-based Distributed CD++ Simulation Algorithms 137
4.1 Distributed CD++ (DCD++) Architecture 139
4.2 DCD++ Simulation Synchronization Algorithms 143
5 Distributed Simulation Interoperability Standards 148
References 155

Chapter 7

Agile Net-Centric Systems Using DEVS Unified Process 159
Saurabh Mittal
1 Introduction 160
2 Related Technologies 162
3 DEVS Unified Process with DEVS/SOA 163
3.1 Discrete Event Systems Specification 163
3.2 Web Services and Interoperability Using XML 165
3.3 An Abstract DEVS Service Agent 166
3.4 DEVS/SOA Framework for Net-Centric Modeling and Simulation 166
3.5 DEVS Unified Process a.k.a DUNIP 169
4 Multi-layered Agent-Based Test Instrumentation System Using GIG/SOA 171
4.1 Deploying Test Agents over the GIG/SOA 172
4.2 Implementation of Test Federations 173
5 Abstract DEVS Service Wrapper 174
6 Workflow Composition and DoDAF-Based Mission Threads 175
6.1 Web Service Work Flow Formalism 177
6.2 Mapping of DEVS, BPEL and DoDAF Artifacts with WSWF Formalism 180
7 Case Study 182

7.1 DEVS Wrapper Agent 182
7.2 Workflow Design, Analysis and Execution 185
8 Agility in DEVS Unified Process 191
9 Conclusions and Future Work 193
10 Acronyms 196
References 197

Chapter 8

Systems Engineering and Conversational Agents 201
James O'Shea, Zuhair Bandar, Keeley Crockett
1 Introduction 201
2 The Scope of CAs 202
2.1 Spoken Dialogue Systems 202
2.2 Chatterbots 203
2.3 Natural Language Processing Based Dialogue Management Systems 203
2.4 Goal-Oriented CAs 204
2.5 Embodied CAs 206
3 Practical Applications of CAs 207
3.1 CAs for Selling 207
3.2 A GO-CA Student Debt Advisor 210
4 Design Methodology for GO-CAs 212
4.1 Knowledge Engineering 212
4.2 Implementation 213
4.3 Scripting Language 214
4.4 Evaluation 217
4.5 Maintenance 219
5 Novel Algorithms – Short Text Semantic Similarity 221
5.1 The STASIS Algorithm 222
5.2 Latent Semantic Analysis 224
6 Research Opportunities 225
7 Conclusions 226
References 227

Chapter 9

Advanced Concepts and Generative Simulation Formalisms for Creative Discovery Systems Engineering 233
Levent Yilmaz, C. Anthony Hunt
1 Introduction 233
1.1 Motivation 236
1.2 Scientific Problem Solving with Computational Models 236
2 Models and Principles of Creative Problem Solving 239
2.1 Background 239
2.2 Models of Creative Cognition 240

3 Generative Parallax Simulation: Basic Concepts 242
3.1 An Abstract Model of Creative Cognition 242
3.2 Abstract Specification of the Structure and Dynamics of GPS 243
3.3 Implications of the Ecological Perspective 246
4 Meta-simulation of GPS 246
4.1 Conceptual Model for GPS Simulator 246
4.2 Meta-simulation Parameters 249
4.3 Qualitative Analysis of Results and Discussion 249
5 Discussion and Future Work 255
5.1 Improving Autonomy in Schema Evolution and Diffusion 255
5.2 Toward Adaptive Growth of Analogue Ensembles for Creative Discovery Systems 256
5.3 Strategic and Context Sensitive Exploration 256
6 Conclusions 257
References 257

Chapter 10

Establishing a Theoretical Baseline: Using Agent-Based Modeling to Create Knowledge 259
Jose J. Padilla, Saikou Y. Diallo, Andres A. Sousa-Poza
1 Introduction 259
2 Systems Engineering and Its Challenges 260
3 Theory and Theory Creation 263
4 Building Theory through M&S 266
4.1 Existing M&S Methodologies/Methods for Theory Building 270
4.2 A Methodology for Theory Building Using M&S 274
4.3 Selecting the Modeling Paradigm 275
5 Test Case: Building a Theory of Understanding Using Agents 276
5.1 Brief on ABM and Its Relevance on Theory Building 276
5.2 Importance of Understanding to Problem Situations 277
5.3 Implementing the Methodology for Theory Building Using M&S 277
6 Final Remarks and Conclusion 281
7 List of Acronyms 282
References 282

Chapter 11

"*The User Around the Marketplace*": Automatic Engineering of Interactive E-commerce Applications 285
Martín López-Nores, Yolanda Blanco-Fernández, José J. Pazos-Arias
1 Introduction 285
2 Background 287
3 Elements to Engineer Personalized Interactive Applications 290
4 The Personalization Procedures 293
4.1 Reasoning-Driven Recommendation of Items 293
4.2 Composition of Interactive Commercial Applications 295
4.3 Feedback 296
5 Our Proposal in DTV Advertising 297
5.1 A Simple Example 298
5.2 Experimental Evaluation 300
6 Conclusion 303
References 303

Chapter 12

Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies 309
Raja Jurdak, X. Rosalind Wang, Oliver Obst, and Philip Valencia
1 Introduction 309
2 Types of WSN Anomalies 310
2.1 Network Anomalies 313
2.2 Node Anomalies 315
2.3 Data Anomalies 316
2.4 Other Anomalies 318
3 Anomaly Detection Strategies 318
3.1 Architecture 320
3.2 Usability 321
4 Design Guidelines and Conclusions 323
References 324

Chapter 13

Enterprise Ontologies – Better Models of Business 327
Ian Bailey
1 Introduction – Intelligence-Led Systems Engineering 327
1.1 Introduction – Business Ontologies 329
1.2 Information System Requirements Gathering 329
1.3 Driving-Out Complexity 331
1.4 Stovepipes 331
2 What Is Needed for Better Information Systems? 332

2.1 Better Analysis – Getting Your Hands Dirty 333
2.2 Flexibility – Using the Full Range of Logic 334
2.3 Consistency – Sophisticated, Repeatable Analysis ... 335
2.4 Implementation – New Ways of Storing 335
3 A New Approach to Information Systems Development 336
3.1 Introducing the BORO Method 336
3.2 Managing Time 337
4 Addressing Arguments against Ontology 340
5 Conclusions 341
5.1 Literature Search 341
References 341

Author Index 343

List of Contributors

Kevin MacG. Adams
National Centers of System of Systems Engineering
Old Dominion University
Norfolk, VA, USA

Khaldoon Al-Zoubi
Dept. of Systems and Computer Engineering Carleton University
Ottawa, ON. K1S 5B6
Canada

Ian Bailey
Model Futures Limited
London, United Kingdom

Zuhair Bandar
School of Computing, Mathematics & Digital Technology
Manchester Metropolitan University
Manchester M1 5GD
United Kingdom

Yolanda Blanco-Fernández
Department of Telematics Engineering
University of Vigo
Vigo, Spain

Keeley Crockett
School of Computing, Mathematics & Digital Technology
Manchester Metropolitan University
Manchester M1 5GD
United Kingdom

Saikou Y. Diallo
Virginia Modeling Analysis and Simulation Center
Old Dominion University
Norfolk, VA, USA

Michele Fumarola
Delft University of Technology
The Netherlands

C. Anthony Hunt
Department of Bioengineering and Therapeutic Sciences
University of California
San Francisco, CA, USA

Dietmar Jannach
Technical University of Dortmund
Baroperstraße 301
D-44227 Dortmund, Germany

Raja Jurdak
CSIRO ICT Centre
and University of Queensland/School of ITEE
Brisbane, QLD
Australia

Charles B. Keating
Engineering Management and Systems Engineering
Old Dominion University
Norfolk, VA, USA

Martín López-Nores
Department of Telematics
Engineering
University of Vigo
Vigo, Spain

Saurabh Mittal
Dunip Technologies
Tempe, AZ, USA

Miklos Nagy
KMI, Open University
Milton Keynes, MK7 6AA
England, UK

Oliver Obst
CSIRO ICT Centre
Sydney, NSW
Australia

James O'Shea
School of Computing, Mathematics &
Digital Technology
Manchester Metropolitan University
Manchester M1 5GD
United Kingdom

José J. Padilla
Virginia Modeling Analysis and
Simulation Center
Old Dominion University
Norfolk, VA, USA

José J. Pazos-Arias
Department of Telematics
Engineering
University of Vigo
Vigo, Spain

Mamadou D. Seck
Delft University of Technology
The Netherlands

Andres Sousa-Poza
Engineering Management and
Systems Engineering
Old Dominion University
Norfolk, VA, USA

John F. Sowa
VivoMind Research, LLC
Croton on Hudson, New York
USA

Claudia Szabo
Department of Computer Science
National University of Singapore
Singapore

Andreas Tolk
Engineering Management and
Systems Engineering
Old Dominion University
Norfolk, VA, USA

Philip Valencia
CSIRO ICT Centre
Brisbane, QLD
Australia

Maria Vargas-Vera
Computing Department
Open University
Milton Keynes, MK7 6AA
England, UK

Alexander Verbraeck
Delft University of Technology
The Netherlands

Gabriel A. Wainer
Dept. of Systems and Computer
Engineering Carleton University
Ottawa, ON. K1S 5B6
Canada

X. Rosalind Wang
CSIRO ICT Centre
Marsfield, NSW
Australia

Levent Yilmaz
Department of Computer Science and
Software Engineering
Auburn University
Auburn, AL, USA

Resumes of Contributing Authors

Kevin MacG. Adams is a Principal Research Scientist at the National Centers for System of Systems Engineering (NCSoSE) of Old Dominion University in Norfolk, Virginia. He holds a Ph.D. in systems engineering from Old Dominion University, dual Master's degrees in Materials Engineering and Naval Architecture and Marine Engineering from the Massachusetts Institute of Technology, and a Bachelor's degree in Ceramic Engineering from Rutgers University. His research focuses on system of systems engineering, systems engineering methodologies, software engineering project management frameworks, the philosophy of science, and the use of enterprise architectures. He is a retired Navy submarine engineering duty officer, a senior member of the Institute of Electrical and Electronics Engineers (IEEE), a member of the American Association of University Professors, and the United States Naval Institute.

Khaldoon Al-Zoubi is a Ph.D. student in Electrical Engineering within the Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada. He is also a senior software analyst and programmer with over 13 years of industry experience occupying a number of seniority and leadership positions. His industry experience spreads over wide range of areas such as embedded software and mobility, air-traffic software management and telecommunications, and security software for explosive and narcotics detections.

Ian Bailey founded Model Futures in 2004 to provide software, consultancy and training in information management. He specializes in enterprise architecture and ontology. He was the technical lead in the development of the UK MOD Architecture Framework (MODAF) and co-developed the SOA views for the NATO Architecture Framework. He is the lead modeler in the multi-nation IDEAS ontology project, which targets to become a common foundation ontology for defense enterprise architecture. Previous to working with MODAF, Ian was editor of the ISO10303-233 systems engineering standard. Most of his professional life prior to Model Futures was spent integrating and re-engineering large scale information systems for customers such as Amec, BAE Systems, BP, Shell and Volvo. He has a Ph.D. in data mapping and a first degree in mechanical engineering. He is a fellow of the Institute of Engineering and Technology (IET).

Zuhair Bandar is a Reader in Intelligent Systems at MMU. He received his Ph.D. in AI and Neural Networks from Brunel University, his M.Sc. in Electronics from the University of Kent and his B.Sc. in Electrical Engineering from Mosul University. He is a co-founder of the ISG and his research interests include the application of AI to

psychological profiling. He is the Technical Director of Convagent Ltd, an MMU spinout company which provides business rule automation with natural language interfaces using conversational agents.

Yolanda Blanco-Fernández was born in Orense, Spain in 1980. She received the Telecommunications Engineering Degree from the University of Vigo in 2003, and the Ph.D. degree in Computer Science from the same University in 2007. Nowadays, she is an assistant professor in the Department of Telematics Engineering, teaching in courses related to network management systems, multimedia services and operating systems. Her main research activity involves development of personalization services for Interactive Digital TV and e-commerce, by applying technological foundations borrowed form the Semantic Web, Web 2.0 and cloud computing.

Keeley Crockett is a Senior Lecturer at MMU. She received her Ph.D. in Machine Learning from MMU and her B.Sc. in Computation from the University of Manchester Institute of Science and Technology. She is a committee member of the IEEE Women into Computational Intelligence Society and a full member of the IEEE Computational Intelligence Society. Her main research interests include fuzzy decision trees, applications of fuzzy theory, and data mining. She is a knowledge engineer and founding member of Convagent Ltd.

Saikou Y. Diallo is a Research Assistant Professor with the Virginia Modeling, Analysis and Simulation Center (VMASC) at Old Dominion University, Suffolk, VA. He received his B.Sc. in Computer Engineering and his M.S. and Ph.D. in Modeling and Simulation from Old Dominion University. He is author of several awarded articles on interoperability and composability. His research focus is on the theory of interoperability.

Michelle Fumarola is a Ph.D. student at the Systems Engineering Group of Delft University of Technology. His Ph.D. research is focused on developing simulation games for decision making with a strong visual component.

C. Anthony Hunt is Professor of Bioengineering and Therapeutic Sciences at the University of California in San Francisco, where he directs the BioSystems Group. He earned a Ph.D. in pharmaceutical chemistry from the University of Florida and B.Sc. degrees in both chemistry and applied biology from the Georgia Institute of Technology. He chairs the BioSystems Group that develops and uses advanced modeling and simulation methods to achieve deeper insight into the networked micromechanisms that link molecular level events with higher level phenomena and operating principles at cell, tissue, organ, and organism levels in the presence and absence of interventions. He is a member of the Editorial Boards of Simulation, Transactions of the SCS, the International Journal of Knowledge Discovery in Bioinformatics, and the Journal of Computational Biology and Bioinformatics Research. He is a Fellow of the American Association for the Advancement of Science and the American Association of Pharmaceutical Scientists, a Director of The McLeod Modeling and Simulation Network, and a member of several professional scientific and engineering associations.

Dietmar Jannach is a full professor at Technische Universität Dortmund, Germany and the head of the e-Services Research Group. His research interests include interactive recommender systems and conversational preference elicitation, engineering of knowledge-based systems and web applications as well as the application of Artificial Intelligence in industry. He has authored and co-authored more than 100 scientific papers in these areas and published papers in journals such as Artificial Intelligence, AI Magazine, IEEE Intelligent Systems and on conferences such as IJCAI and ECAI.

Raja Jurdak is a Principal Research Scientist at CSIRO ICT Centre. He holds a Ph.D. in Information and Computer Sciences and an MS in Computer Engineering from the university of California, Irvine and a BE in Computer and Communications Engineering from the American University of Beirut. He is an adjunct Associate Professor at University of Queensland's School of Information Technology and Electrical Engineering. He is also a member of the IEEE and the IEEE Communications Society. His current research interests focuses on modeling, optimization, and real world deployments of energy-efficient and highly responsive sensor networks. He has over 40 peer-reviewed journal and conference publications, as well as a book published by Springer titled Wireless Ad Hoc and Sensor Networks: A Cross-Layer Design Perspective.

Charles B. Keating is a Professor of Engineering Management and Systems Engineering at Old Dominion University located in Norfolk, Virginia. He also serves as the Director for the National Centers for System of Systems Engineering (NCSoSE) where his research is focused on development and testing of theory, methodologies, and technologies to more effectively deal with complex system problems. Prior to joining the university, Dr. Keating had over 12 years of experience in command and technical engineering management positions in the U.S. Army, Texas Instruments, and Newport News Shipbuilding. Dr. Keating holds a B.Sc. in Engineering from the United States Military Academy (West Point), an M.A. in Management from Central Michigan University, and a Ph.D. in Engineering Management from Old Dominion University. His current research interests include: System of Systems Engineering, Complex System Problem Domains, and Complex System Governance.

Martín López-Nores was born in Pontevedra, Spain in 1980. He received the Telecommunications Engineering Degree from the University of Vigo in 2003, and the Ph.D. degree in Computer Science from the same University in 2006. Nowadays, he is an associate professor in the Department of Telematics Engineering, teaching in courses related to computer networks, operating systems and information services. Starting from works on applied formal specification techniques, his research interest have evolved to embrace the design and development of interactive services for a range of consumer electronics devices, the design and evaluation of communication protocols and innovative applications for mobile ad-hoc networks, and the management of health-related data in semantics-based recommender systems and pervasive computing environments.

Saurabh Mittal is the President and founder of Dunip Technologies, Tempe, AZ, USA and is also a Research Scientist at US Air Force Research Lab (AFRL), 711th Human Performance Wing, for L-3 Communications, Link Simulation and Training Branch in Mesa, AZ. His work at AFRL involves multiformalisms, cognitive modeling and net-centric systems engineering. He holds a Ph.D. (2007) and an M.S. (2003), both in Electrical and Computer Engineering from the University of Arizona, Tucson. He was recognized with Golden Eagle Award, highest recognition to any civilian by Joint Interoperability Test Command and US Air Force, supporting the US Warfighter for his work on Generic Network System Capable of Planned Expansion (GENETSCOPE) in 2006. His research focuses on Discrete Event modeling using DEVS Formalism, net-centric system of systems engineering with DEVS Unified Process, executable architectures using Department of Defense Architecture Framework (DoDAF), simulation-based computing and large scale M&S infrastructures using Service oriented architecture. He is a member of Institute of Electrical and Electronics Engineers (IEEE), Association of Computer Machinery (ACM), and Society for Modeling and Simulation International (SCS).

Miklos Nagy is a Ph.D. candidate at the Open University's Knowledge Media Institute. His research interests are Uncertain Reasoning, Ontology Mapping, Multi-agent systems and information integration using Semantic Web technologies. His current research focuses on the development of intelligent multi-agent systems that can exploit the emerging Semantic Web's large-scale data. Miklos Nagy received his MSc in Information Engineering from the University of Miskolc, Hungary.

Oliver Obst is Research Scientist at the Adaptive Systems Team at the Australian Commonwealth Scientific and Research Organization (CSIRO) in Sydney. He holds a Ph.D. and a M.Sc. in computer science, both from the University of Koblenz-Landau in Koblenz, Germany. He is affiliated as an honorary associate with the School of Information Technologies of the University of Sydney. His research fields are information processing in neural networks, the application of information-theory to guide self-organization in complex systems, the representation of sensory information, as well as the emergence of coding for both technical and biological systems. His work involves development of new machine learning algorithms and architectures, as well as their applications to real world problems, for example in fault detection in distributed systems, such as smart electrical grids or sensor networks. He is a member of the International Neural Network Society (INNS), and the German Informatics Association (GI e.V.).

James O'Shea is a senior lecturer at Manchester Metropolitan University (MMU). He received his B.Sc. in Chemistry from Imperial College. He worked in computer R&D at International Computers and as an independent consultant under the UK Microelectronics Applications Project until 1985. After joining MMU, he developed a research interest in AI and co-founded the Intelligent Systems Group (ISG). In addition to his work in the field of CAs, he is one of the inventors of the Silent Talker lie detector, which has attracted worldwide interest.

Jose J. Padilla is a Post Doctoral Research Associate with the Virginia Modeling, Analysis and Simulation Center (VMASC) at Old Dominion University, Suffolk, VA.

He received his Ph.D. in Engineering Management from Old Dominion University. He also holds a B.Sc. in Industrial Engineering from the Universidad Nacional de Colombia, Medellín, and an M.B.A. International Business from Lynn University, Boca Raton, Florida. His research interests are on the nature of the process of understanding and how it contributes to the perception of complexity for a human or a computer agent.

José J. Pazos-Arias is Full Professor at Department of Telematics Engineering at the University of Vigo (Spain). He received his degree in Telecommunications Engineering from the Technical University of Madrid (Spain-UPM) in 1987, and his Ph.D. degree in Computer Science from the Department of Telematic Systems Engineering at the same University in 1995. He is the director of the Interactive Digital TV Laboratory, which is currently involved with national and international projects, receiving funds from both public institutions and industry. With the aim of combining the power of semantic reasoning technologies and the participation phenomena arising in the knowledge society, he is currently involved in the use of social-semantic technologies to assist the users when it comes to facing complex decision takings in the cloud. In this regard, he is highly interested in gaining deeper knowledge in social network analysis and emergent semantics.

Mamadou D. Seck received his Ph.D. degree from the Paul Cezanne University of Marseille and his MS and M.Eng Degrees from Polytech' Marseille, France. He is currently an Assistant Professor in the Systems Engineering section at the Technology, Policy, and Management department of Delft University of Technology. His research interests include modeling and simulation formalisms, dynamic data driven simulation, human behavior representation and social simulation, and agent directed simulation.

Andres Sousa-Poza is an Associate Professor of Engineering Management and Systems Engineering at Old Dominion University. He holds a B.Sc. in Mechanical Engineering from the University of Cape Town, South Africa, and M.S. and Ph.D. degrees in Engineering Management from the University of Missouri-Rolla, USA. He is the co-founder and present chair of the Management and Engineering in Complex Situations Forum (MECS Forum). He is affiliated with the National Centers for System of Systems Engineering (NCSoSE) as a senior researcher.

John F. Sowa spent thirty years working on research and development projects at IBM and is a cofounder of VivoMind Research, LLC. He has a B.Sc. in mathematics from MIT, an MA in applied mathematics from Harvard, and a Ph.D. in computer science from the Vrije Universiteit, Brussel. He is a fellow of the American Association for Artificial Intelligence. With his colleagues at VivoMind, he has been developing novel methods for using logic and ontology in systems for reasoning and language understanding. He designed the language of conceptual graphs, which has been adopted as one of the three normative dialects of the ISO/IEC standard for Common Logic.

Claudia Szabo is a Research Assistant at the Department of Computer Science, National University of Singapore. She received her B.Sc. from the "Politehnica" University of Bucharest and is a Ph.D. candidate in Modeling and Simulation at the

National University of Singapore. Her research evaluates simulation composability and interoperability, with a focus on formal validation. A unique aspect of her work is to enable trade-off analysis between validation accuracy and computational cost. She is the author of several awarded articles of composability and validation, including the 2009 ACM SIGSIM Best Ph.D. Student Paper Award.

Andreas Tolk is Associate Professor for Engineering Management and Systems Engineering at Old Dominion University in Norfolk, Virginia. He holds a Ph.D. and an M.S. in computer science, both from the University of the Federal Armed Forces of Germany in Munich. He is affiliated as a Senior Research Scientist with the National Centers for System of Systems Engineering (NCSoSE) in Norfolk, Virginia, and the Virginia Modeling Analysis and Simulation Center (VMASC) in Suffolk, Virginia. His research focuses on integratability and composability of model-based solutions and modeling and simulation based systems engineering. He is member of the American Society for Engineering Management (ASEM), Association for Computing Machinery (ACM), Institute of Electrical and Electronics Engineers (IEEE), Military Operational Research Society (MORS), National Defense Industrial Association (NDIA), Simulation Interoperability Standards Organization (SISO), and Society for Modeling and Simulation International (SCS). He was recognized with the Excellence in Research Award of the Frank Batten College of Engineering and Technology and received the first Technical Merit Award of SISO.

Philip Valencia is a Research Engineer at the Autonomous Systems Laboratory within the Commonwealth Scientific and Research Organization (CSIRO) ICT Centre. He holds bachelor degrees in Engineering (Electronic) and IT from the Queensland University of Technology, Brisbane, Australia and is undertaking Ph.D. studies at the University of Queensland. He has eight years research experience with wireless sensor network technologies as well as background in machine learning which he has brought together to research distributed online learning for wireless sensor and actuation networks.

Maria Vargas-Vera is a Lecturer in Computing at the Open University, England UK. She received her Ph.D. from the Artificial Intelligence Department at Edinburgh University and her MSc in Computer Science from the National University of Mexico (UNAM). She was awarded Fellow of the British Computer Society (FBCS) from November 2009. Her current research focuses on Automatic Construction of Ontologies from Text, Ontology Mapping and E-Learning Applications using Semantic Web Technologies. Maria Vargas-Vera has published many research papers in prestigious journals and international conferences and she is a member of program committees of international conferences and workshops. Also, she is an Associated Editor of the International Journal of Knowledge and Learning (IJKL) and the Journal of Emerging Technologies in Web Intelligence (JETWI).

Alexander Verbraeck got his M.Sc. in mathematics in 1987 and his Ph.D. in 1991 from Delft University of Technology in the Netherlands. Until 1992 he also had his own software company, focusing on consultancy and software development for educational institutes. He worked as assistant professor in information systems until 1995, when he was appointed associate professor in the systems engineering group of

the faculty of Technology, Policy and Management (TPM) of TU Delft. He is the chair of the Systems Engineering research group and he has been department chair of the Department of Information, Communication and Systems of the TPM Faculty of TU Delft. He was the original author and program manager of the BETADE strategic research program of TU Delft. He has also been appointed part-time research professor at the R.H. Smith School of Business of the University of Maryland, USA.

Gabriel Wainer is Associate Professor at the Department of Systems and Computer Engineering of Carleton University, Ottawa, ON, Canada. He is head of the Advanced Real-Time Simulation lab, located at Carleton University's Centre for advanced Simulation and Visualization. He received the M.Sc. (1993) and Ph.D. (1998) in Computer Science from the Universidad de Buenos Aires, Argentina, and IUSPIM (now Polytech de Marseille), Université Paul Cézanne, Aix-Marseille III, France. Previously, he was Assistant Professor in the Computer Sciences Department of the Universidad de Buenos Aires, and held visiting positions in numerous places, including the Arizona Center of Integrated Modeling and Simulation (ACIMS, University of Arizona), Laboratory of Systems Sciences of Marseille (LSIS-CNRS), University of Nice, Polytech de Marseille, INRIA Sophia-Antipolis (France). He is the Vice-President Publications, and was a member of the Board of Directors of the Society for Computer Simulation International (SCS). He is the author of three books and over 200 articles in different venues. He has collaborated in the organization of over 100 conferences and is co-founder of the SIMUTools Conferences. He has been the recipient of various awards, including the IBM Eclipse Innovation Award, a Leadership award by SCS, and various Best Paper awards. He has been awarded Carleton University's Research Achievement Award and the First Bernard P. Zeigler DEVS Modeling and Simulation Award.

Rosalind Wang is currently a research scientist in the ICT Centre, CSIRO. She received her Ph.D. in Mechatronics Engineering from the University of Sydney, Australia in 2009. Her research interests are machine learning, graphical models, and pattern recognition.

Levent Yilmaz is Associate Professor of Computer Science and Software Engineering and holds a joint appointment with the Industrial and Systems Engineering department at Auburn University. He received his B.Sc. degree in Computer Engineering and Information Sciences from Bilkent University and M.S. and Ph.D.. degrees in Computer Science from Virginia Tech. His research interests are in Modeling & Simulation, Agent-directed Simulation, and Complex Adaptive Systems, focusing in theory and methodology of modeling and simulation to advance scientific discovery and theory formation, and to develop robust decision support systems and models of socio-technical, cognitive, and cultural systems, such as science of science and innovation policy. He is a member of the Board of Directors of SCS and is the Editor-in-Chief of Simulation: Transactions of the Society for Modeling and Simulation International. He is member of ACM, IEEE Computer Society, Society for Computer Simulation International (SCS), and Upsilon Pi Epsilon.

Chapter 1
Towards Intelligence-Based Systems Engineering and System of Systems Engineering

Andreas Tolk, Kevin MacG. Adams, and Charles B. Keating

Department of Engineering Management and Systems Engineering
National Centers of System of Systems Engineering
Old Dominion University
Norfolk, Virginia 23508-2563, USA
atolk@odu.edu, kmadams@odu.edu, ckeating@odu.edu

Abstract. This introductory chapter defines intelligence-based systems with focus on semantic systems, simulation systems, and intelligent agents. Semantic systems define the foundation to communicate systems engineering challenges using logic, simulation systems introduce the dynamic component, and intelligent agents can introduce alternatives roles. It then gives an overview of traditional systems engineering as well as system of systems engineering showing the need to emphasize the system of systems perspective in modern engineering approaches. Finally, both views are aligned, providing a scope for intelligence-based systems engineering and the contributions of the following book chapters are summarized in relationship to this scope.

Keywords: intelligent agents, ontology, semantic system, simulation system, system of systems engineering, systems engineering.

1 Introduction

The definition of *insanity* as "doing the same thing over and over again and expecting different results" is attributed to Albert Einstein. In contrast, a collective definition for *intelligence* is the ability to comprehend, to understand and profit from experience, or to make sense out of the environment and react appropriately. In the light of these two extremes, this introductory chapter defines what intelligence-based systems are, and what this means for systems engineering and systems of systems engineering.

Starting with a summary of the state of the art, as among others identified by Buede [1], it can be observed that most of our current systems have been designed starting with a set of well defined requirements. These requirements are often based on operational concepts that identify context and external systems and that are used to derive (a) input and output requirements that identify what a system shall accept and produce, (b) system-wide and technology requirements that are building a set of operational constraints, (c) trade-off requirements that allow optimizing system design decisions within these constraints, and (d) qualification requirements that allow validation and verification to be conducted. These requirements lead to building a functional architecture describing the capabilities of the system, a physical architecture that describes the resources that comprise the system, and finally an allocated

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 1–22.
springerlink.com

architecture that merges the functional and the physical view, including interface design, integration and qualification. The result is a well-defined system that has a well defined behavior for all identified input constellation in the form of expected output produced. As a rule, the capabilities defined in the functional architecture are fixed. The system will do the same thing over and over again. Under many circumstances, this is exactly what we would want. Nobody wants to push down the brake pedal of a car expecting anything else but that the car stops. We expect the same results. However, what if the environment changes? What if the world in which a system was originally defined no longer exists?, like we currently see it in so many military systems that were defined at the time of the Cold War, but still have to be used today? Simply expecting the system to change its behavior qualifies as insanity, so we need intelligent systems that are able to comprehend, understand and profit from experience.

The next section will define intelligence-based systems. Following these definitions and examples, the third section will evaluate the relation of such systems with systems engineering. The fourth section will do the same for the new and emerging field of system of systems engineering that adds at least one additional layer of complexity to the challenges to be addressed. Finally, the last section will describe the contributions comprised in this book in the light of these findings.

2 Intelligence-Based Systems

Intelligence-based systems should not be confused with the often narrowly used term intelligence system, which refer to a variety of Artificial Intelligence (AI) methods, such as neural networks, evolutionary algorithms, expert systems, diagnostic systems, symbolic AI, and other related topical areas. These systems are limited to AI applications, and intelligent systems engineering describes the engineering of such intelligent systems, not the use of intelligence to support systems engineering. The scope we take in this chapter – and in this book in general – includes the design and engineering of such intelligent systems, but is not limited to this view. We are interested in merging the state of the art of intelligence as it can be provided via AI methods to support systems engineering and system of systems engineering. How can these three aspects be of mutual support, resulting in better systems that are able to comprehend, understand and profit from experience. This is the objective of intelligence-based systems engineering: to base systems and their design on AI methods to build better systems.

2.1 Characteristics of Intelligence-Based Systems

In order to support this objective of intelligence-based systems engineering, it is first important to better understand the characteristic properties of intelligence-based systems. The following list is neither complete nor exclusive, but it reflects the collective definition of various views on AI, intelligence-based solutions, model-based prediction and control, and similar contributions. Figure 1 depicts these characteristics that are used in the collective definition, which are self-explaining, robust, fault tolerant, adaptive, self-optimizing, deductive, learning, cooperative, autonomous, and agile. As

we will see, these terms have partly overlapping definitions and have to be understood in the context of the collective definition, which means that not all definitions use all terms.

Fig. 1. Characteristic Properties of Intelligence-based Systems

Self-explaining doesn't mean that the system is obvious without any explanation necessary, but that the system can explain how it came to a certain decision. In traditional systems, the system behavior does not change. If a system is able to modify its behavior, it is often needed to understand how and why a decision has been made by the system. The explanation component of expert systems used for diagnosis, which traditionally could be generated by tracing the line of reasoning used by the underlying inference engine to answer the questions: "Why is your answer to the question the one you recommend?" For systems that are able to modify themselves being able to explain their reason is mandatory to ensure credibility.

Robust as a characteristic property of a system means that the system behaves well and adequate not only under ordinary conditions, but also under unusual conditions that stress the original requirements and derived assumptions. In other words, robust systems do not break easily, but are able to continue to behave well even under variant circumstances that could lead to failure of system.

Fault tolerant systems behave well and continue to adequately perform even if one or more of its internal system components fail or break. It may be important to differentiate between a fault, which is a defect in the system that can cause an error, which is a subset of the system status that may lead to system failure, which is a deviation in actual system behavior and its desired behavior according to the requirements.

Adaptive systems in general react to changes, in particular to changes in the environment or the context of the system. Whenever the environment or context of the

system changes the system itself changes as well in order to accommodate these changes. As a consequence, adaptive systems behave well and adequate even in changing environments.

Self-organizing systems organize their internal components and capabilities in new structures without a central or an external authority in place. These new structures can be temporal and spatial. In some cases, instead of self-organizing the term self-optimizing is used synonymously, although not all self-organizing structures represent the optimal structure, but the assumption is that self-organizing systems are organizing themselves to become better.

Deductive systems are well known from mathematics: based on a set of axioms and rules, they can deduct new insights by applying the rules to the axioms as well as to the resulting new facts. This is done using an underlying inference engine. Applying these ideas, deductive systems can discover new facts that they can use for their decision process on how to modify themselves to behave well and adequate.

Learning systems generally observe the achieved results and compare them with the desired outcome. Using methods such as reinforcement learning, decisions that led to positive results are enforced while those with negative results are avoided. Learning can also occur by observing other systems and the results of their activities. In every case, learning is connected with the observation of cause and effects.

Cooperative systems expose social capabilities. This means that cooperative systems interact with other systems – and potentially humans as well – via some kind of communication language. This interaction is not limited to pure observation, but such a system can exchange plans, distribute tasks, etc. Whiteboard technologies are as often used as direct communication. An interesting side effect is that such cooperative systems can themselves then become a self-organizing system of systems.

An *autonomous* system performs the desired tasks and behaves well and adequate even in unstructured environments without continuous human guidance. In the domain of robotics, autonomy is described as a collection of additional characteristics, in particular sensor capabilities to observe chaotic, unpredicted variables and to react to keep the system on track utilizing the available degrees of freedom.

In general, *agile* systems are able to manage and apply knowledge effectively so that they behave well and adequate in continuously changing and unpredicted environments. In systems engineering, agility is often in particular connected with the development phase of systems and reflects the ability to immediately react on changes in the requirements.

Without doubt, additional characteristic properties can be identified that are desirable for such systems, such as self-healing. However, if a system is adaptive, elf-optimizing, and fault-tolerant, self-healing is a result. Similar arguments can be made for the quest to reduce risk and vulnerability and other desirable characteristics.

2.2 How to Capture Intelligence

There are many methods applied in AI to capture intelligence. This chapter deliberately focuses on a limited subset that is of particular interest to systems engineering and for which examples are given in other chapters of this books. Using the well known categories of Ackoff [2], we distinguish between data, information, knowledge, understanding, and wisdom. We understand *data* as a collection of facts. *Information* is data

in a context allowing answering questions like who, what, where, and when. *Knowledge* is applied information answering the question how. *Understanding* introduces an answer to the question why, and *wisdom* finally evaluates understanding and generalizes the findings, allowing application of understanding in other domains than the original source of gaining understanding.

In this chapter and this book, we apply semantic systems or use general ontological means to capture and model data and information. Applying these pieces of information on who, what, where, and when in the context of simulation introduces the aspects addressed by knowledge: *how*. Adding agents allows running not only one but many simulations and comparing alternative courses of action. To communicate between agents, ontology is needed to provide the basis for the communication language supporting the exchange of information. Figure 2 shows the three elements applied in this book.

Fig. 2. Components to Capture Intelligence

A recent book edited by Yilmaz and Ören [3] copes with the various aspects of agent-directed simulation and systems engineering. They also show the increasing importance modeling and simulation methods in general and agent-directed simulations in particular play for intelligence-based systems. Software agents expose many of the characteristic properties described earlier in this chapter.

Agents help designing communication and coordination protocols in the system and may even become a surrogate for a human user. Simulation helps answering questions about the achieved behavior, performance and robustness, giving first feedback about the quality of the design. In addition, simulation can be used for decision support by providing "what if" scenarios as well as for training and education purposes. In addition, agents are likely to replace, to a certain degree, objects that have traditionally been exploited in systems engineering. An interesting aspect evaluated is to replace the functions traditionally developed within the functional architecture of a

system as defined in [1] with agents. As this agent already possesses many characteristics of intelligence-based systems, the result is likely to be close to our objective. However, all three aspects shown in figure 2 are important.

Another example of interest described in [3] is autonomic computing, as it also shares many characteristic properties. Autonomic computing is a potential strategy and philosophy in systems design and management that aims to cope with increasing complexity in the presence of constant change addressing the area of systems of systems engineering which involves: (a) large scope and great complexity of integration efforts; (b) collaborative and dynamic engineering; (c) engineering under the condition of uncertainty; (d) continuing architectural reconfiguration; (e) simultaneous modeling and simulation of emergent behavior; and (f) stakeholders with competing goals and objectives.

Utilizing the characteristics of software agents, autonomic systems are based on architectures and mechanisms that facilitate self-configuration and adaptation through learning, anticipation, and robust designs to be able to adjust and fine tune system parameters to emerging situations in this environment. The main characteristics are self-configuration, self-healing, and self-optimization. The autonomic computing control loop moves from gathering data from resources in the system's environment (sensor) to registering to be notified as the sensors observe changes in the environment (monitor). Next, the status of the environment and operational components' ability to react to change is perceived, interpreted, and understood (analyze) while necessary information about the managed resources, data, and policies are being provided to the system (knowledge). If the analysis and knowledge cannot identify a proper reaction to unforeseen environmental conditions, the reasoning and planning components take control to generate a new plan and identify a sequence of actions to act on the system configurations. Then, those actions are translated into executable commands (execute). These key tenets of autonomic systems (sensor, monitor, analyze, knowledge, reason/plan, execute) provide a roadmap for building intelligence-based systems using all three components mentioned above.

However, the title of this book is not "systems engineering *of* intelligence-based systems," but "intelligence-based systems engineering," which also includes the application of these methods and technologies to improve the traditional systems engineering process and the emerging new field of system of systems engineering. The next section will describe the principles of systems engineering and identify where intelligence-based methods can be applied.

3 Systems Engineering

The genesis for systems engineering in particular in the United States has been attributed to complexity. Early pioneers in the systems engineering field emphasize increasing system complexity as the principal causative factor, although they recognize that this is far from a complete explanation [4] [5]. To explain this, some historical background is warranted.

In the late 1930s the fledgling radio, television, and telephone industries in the United States recognized the need for a systems approach in the development of modern telecommunications services. The Radio Corporation of America (RCA) and its

subsidiary, the National Broadcasting Company (NBC) were interested in the expansion of their television broadcast domain. At the same time, the Bell Telephone Company was interested in the expansion of their long-distance telephone network. Both companies initiated technical studies aimed at increasing their markets through the use of new broadband technologies that were beginning to emerge in the early 1940s. However, these exploratory studies and experimentation were interrupted by the Second World War.

During the Second World War, the American military used large numbers of scientists and engineers to help solve complex logistical and strategic bombing problems related to the war effort. Many of these efforts made significant contributions to the philosophy and techniques of what was then called *Operations Research*. At the same time, the need for many novel types of electronic gear for airborne use gave rise to a wide variety of component devices, popularly known as *black boxes*. These were ingenious devices, but their application in terms of the entire system of which they were merely parts was a matter of improvisation [4]. Inevitably, many of the engineers and scientists working on these *black boxes* were required, by necessity, to look ahead to the ultimate goal – the system. When the war ended a number of corporations (most notably the RAND Corporation, the Bell Telephone Laboratories and RCA) hired much of this pool of talented scientists and engineers to provide services to both the government and the telecommunications industry. These seasoned practitioners were able to capitalize upon the lessons from their war-time experiences in the development and implementation of the modern telecommunications and electrical power systems. The telecommunications system development efforts provide for much of the early literature on systems engineering. Schlager [6], in a nationwide survey found that the Bell Telephone Laboratories was probably the first organization to use the term systems engineering. If true, this places the start of what we call systems engineering, in the early 1940s.

3.1 Traditional Systems Engineering

The emergence of systems engineering in the 1940s was an outgrowth of the need to deal with large, expensive systems. The early textbooks [5] [7] [8] on systems engineering had an emphasis on topics such as decision making, problem solving, and analysis of alternatives. The texts relied heavily on the techniques and analytical methods from Operations Research [9] [10] [11]. A 1957 definition of systems engineering characterizes its early role [12].

> "*The design of systems in which the output is a set of specifications suitable for constructing a real system out of hardware*." (p. 1-4)

Over the next 30 years systems engineering assumed responsibility for not only the technical elements surrounding systems, but the life cycle management responsibilities as well. Systems engineering was, in-part, responsible for the delivery of large complicated projects of national importance that included the Polaris submarine, and the Mercury and Gemini space programs. By 1998 the definition of systems engineering had evolved to [13].

> *"An interdisciplinary collaborative approach to derive, evolve, and verify a life-cycle balanced system solution which satisfies customer expectations and meets public acceptability."* (p. 11)

In 30 years, systems engineering had evolved to include life-cycle management responsibilities, customers, and the public in its definition. Traditional Systems Engineering (TSE) has developed the frameworks and methodologies [13] [14] to successfully conceive, design, acquire, and field large multi-purpose systems. Three often used models developed in support of TSE most readers will recognize are (a) the waterfall model [15], (b) the *Vee*-model [16], and (c) the spiral model [17].

The waterfall model is characterized by the sequential evolution of phases in which as a rule only the two consecutive phases are connected with each other and the feedback is seen as the exception, not the rule. It starts with a set of requirements that are refined for the system and its component, followed by an analysis. The analysis is followed by the detailed design and the implementation of this design. Once the system is implemented, it is tested and afterwards operationally used. Some newer versions include maintenance and retirement as well. All versions of the waterfall model have the philosophy in common that if the engineer is doing a good job in all phases, he can successfully reach the project end. The *Vee*-model follows a slightly different philosophy by integrating the user into the engineering process. It starts with user requirements and ends with user acceptance. The two parts of the *Vee* are built by the phases comprised in the *decomposition and definition* of system components in the downward steps, and the *integration and verification* phases building the upward steps. On all levels, phases of the decomposition and definition are connected to respective integration and verification phases, such as verifying that the correct parts are built, verifying that configuration items are assembled correctly, verifying that the system performs as requested, and validating that the system fulfills all requirements. Overall, the feedback between the different phases and the possibility of corrections of earlier phases that are not necessarily mistakes of the systems engineer build the philosophy. The last model, the spiral model, is based on waterfall and *Vee* model ideas. It assumes that several iterations through the phases of these models will be needed resulting in a spiral in which each iteration leads to the next iterations objectives. Feedback is the rule and no longer the exception. The *spiral* model is an iterative model that combines elements of the classic *waterfall* model with the characteristic of prototyping and produces an evolutionary approach to engineering. The four major phases are (1) management planning, (2) engineering, (3) customer evaluation, and (4) risk analysis. The major distinguishing feature of the spiral is that by including a formal risk analysis phases, it introduced a risk-driven approach to the development process.

Systems Engineering therefore evolved well. However, the 21st century presented a new problem for systems engineering: the system of systems. The next section will exemplify that the traditional methods are insufficient to address these new challenges so that a new theory is needed that allows to derive methods and implement solutions.

3.2 System of Systems

Most 20th century systems were designed and implemented to satisfy specific functional objectives. The objectives were typically focused on the requirements in a single

functional area (i.e. accounting, inventory control, manufacturing, railroads, highways, etc.), resulting in a number of vertically independent, or stove-piped, systems within an organization or society. Few were designed to satisfy all of the functions required by the organization or society they were serving and as such are classified as monolithic in structure.

Today, large numbers of 20th century systems operate within these functional stovepipes, providing functionality inside but not across the stovepipes. Initial efforts to bridge the functional stovepipes have focused on integrating 20th century systems through a series of system-to-system interfaces. However, 21st century managers are no longer satisfied with disparate systems lashed together with complex interfaces and data validation routines. Enterprise Resource Planning (ERP) systems were supposed to be the panacea for the business world, replacing stove-piped legacy systems with a single system encompassing all of a company's functional requirements. In 1998 it was estimated that businesses around the world were spending $10 billion per year [18] on enterprise systems and that figure probably doubles when you add in associated consulting expenses.

By the turn of the century, a new type of system, beyond that envisioned by the late Russell Ackoff in his paper *The Systems Revolution* [19], began to emerge. It is the super-system, the metasystem, the system-of-systems which is made up of components which are large-scale systems themselves. If we are to understand system-of-systems we must be able to differentiate them from the more common monolithic systems.

Although the term system-of-systems has no widely accepted definition, Maier notes that the notion is widespread and generally recognized [20]. The following distinguishing characteristics have been proposed [20] [21].

1. *Operational Independence of the Individual Systems:* A system-of-systems is composed of systems that are independent and useful in their own right. If a system-of-systems is disassembled into the component systems, these component systems are capable of independently performing useful operations independently of one another.

2. *Managerial Independence of the Systems:* The component systems not only can operate independently, they generally do operate independently to achieve an intended purpose. The component systems are generally individually acquired and integrated and they maintain a continuing operational existence that is independent of the system of systems.

3. *Geographic Distribution:* Geographic dispersion of component systems is often large. Often, these systems can readily exchange only information and knowledge with one another, and not substantial quantities of physical mass or energy.

4. *Emergent Behavior:* The system-of-systems performs functions and carries out purposes that do not reside in any component system. These behaviors are emergent properties of the entire system-of-systems and not the behavior of any component system. The principal purposes supporting engineering of these systems are fulfilled by these emergent behaviors.

5. *Evolutionary Development:* A system-of-systems is never fully formed or complete. Development of these systems is evolutionary over time and with structure, function and purpose added, removed, and modified as experience with the system grows and evolves over time.

These distinguishing characteristics begin to place some degree of formality on the notion of system-of-systems, but something is missing. In order to go beyond the traditional perspective of a fully integrated system-of-systems which perfectly shares data in what we call hard interoperability, we must invoke a more systemic view. The ideal state for a system-of-systems requires what we will call *systemic interoperability*. Systemic interoperability is a holistic view of interoperability and requires compatibility in worldview and conceptual, contextual, and cultural interoperability, allowing the system-of-systems to act consistently with regard to purpose, function, and form. In other words, it is not sufficient to align the implementation details of the participating systems, but the underlying conceptualization and the assumptions and constraints need to be aligned as well. This is where System of Systems Engineering comes into play.

3.3 System of Systems Engineering

During the evolution of TSE, the educational texts [22] [23] [24] and curricula eliminated topics on the fundamental concepts and properties associated with systems and include few *soft* topics to encompass the rich context and human situations that real-world systems of systems engineering problems present.

Man-made systems of systems require a holistic, systemic understanding of both the technical problem and the contextual framework present in order to arrive at satisfactory solutions. A new set of methodologies and frameworks based upon formal systems principles are required. The new methodologies will also require new supporting methods, techniques, and tools.

The emerging discipline of System of Systems Engineering (SoSE) is attempting to address the problems associated with systems-of-systems. Because these problems are *messy* traditional methodologies of systems engineering are excluded from consideration in this context. Russell Ackoff coined the concept of a "mess" and "messes" [25]:

> *"Because messes are systems of problems, the sum of the optimal solutions to each component problem taken separately is not an optimal solution to the mess. The behavior of the mess depends more on how the solutions to its parts interact than on how they interact independently of each other. But the unit in OR is a problem, not a mess. Managers do not solve problems, they manage messes."* (p. 100)

Keating et al. [26] cite three important problems that TSE is not prepared to address when facing a complex metasystem problem:

1. Has not been developed to address high levels of ambiguity and uncertainty in complex systems problems . . . it strains to adequately respond to ill-structured problems with constantly shifting requirements.
2. Does not completely ignore contextual influences on system problem formulation, analysis, and resolution, but places context in the background.
3. Is not prepared to deliver incomplete or partial solutions that include iterative design and implementation after deployment.

Keating et al [26] provisionally define SoSE as:

> *"The design, deployment, operation, and transformation of metasystems that must function as an integrated complex system to produce desirable results. These metasystems are themselves comprised of multiple autonomous embedded complex systems that can be diverse in technology, context, operation, geography, and conceptual frame."* (p. 23)

Multiple, autonomous, embedded, complex systems function as a single meta-system, or system-of-systems. This is possibly the most daunting task ever presented to systems engineers. It exists within a unique new context and will require an entirely new methodological problem solving approach.

From a programmatic and enterprise viewpoint, TSE emphasized a system-centric view with individually designed, developed, implemented and optimized solutions, which necessarily incorporated the danger of stovepipes and fragmentation. What is envisioned, however, is an integrated system approach in which each system provides capabilities in an easy and composable way to support the rapid reconfiguration. To support this vision, a methodology is needed that guides systems engineers in the new system of systems problem domain.

3.4 System of Systems Engineering Methodology

Currently, there is no widely accepted approach to conducting System of Systems Engineering (SoSE) efforts. However, there is recognition that approaches must address challenges of increasingly complex systems that must be conceived, built, operated, and evolved in a changed landscape marked by: (1) an exponential rise in the demand, accessibility, and proliferation of information, (2) increasing interdependence and demands for interoperability between systems that have previously been developed, tested, operated, and maintained in isolation, (3) missions and flow down requirements that are subject to rapid and potentially radical shifts due to policy, organizational, funding, or other factors beyond the technical aspects of the system, and (4) demands for the accelerated fielding of systems that are technically incomplete, but offer an improved alternative to what is presently available [27].

The SoSE Methodology [28] is a rigorous engineering analysis that invests heavily in the understanding and framing of the problem under study. By conducting a rigorous engineering analysis of the problem and its associated context, the SoSE Methodology minimizes the chance that a Type III error or solving the wrong problem precisely and efficiently [29] [30] may be committed early on in a SoSE analysis. It is important that the SoSE Methodology is not taken as a prescriptive approach to addressing complex SoSE problems. Instead, the SoSE Methodology must be taken as a guide, to be adapted to the particular circumstances that define its application. Otherwise, it will not serve its intended purpose: to provide a high level adaptable structure to guide rigorous exploration of complex systems problem situations.

The SoSE Methodology is intended to provoke rigorous analysis – resulting in the potential for alternative decision, action, and interpretations for evolving complex system of systems solutions. The SoSE Methodology is based in facilitating inquiry that is as much about thinking and framing of problems, their context, and managing emergent conditions as it is about taking decisive action. The SoSE Methodology was

purposefully built and seeks to provoke higher levels of inquiry, systemic analysis, and advance understanding of seemingly intractable problems enroute to more robust solutions.

We position the SoSE Methodology to be consistent with Checkland's [31] perspective of a methodology, which suggests that a methodology provides a framework, more specific than philosophy, but more general than a detailed method or tool. Therefore, a systems-based methodology must provide a framework that can be elaborated to effectively guide action. There are several critical attributes for a methodology and these are consistent with the current state of evolution for the SoSE Methodology. These critical attributes are discussed in the next section.

There are several critical attributes in the SoSE Methodology that are consistent with the current state of evolution for SoSE. Although the listing is certainly not intended to be exhaustive, we offer these as insight to our thinking with respect to the characteristics that make the SoSE Methodology sustainable. The nine (9) critical attributes and how the SoSE Methodology satisfies these are presented in Table 1.

Table 1. Critical Attributes of the SoSE Methodology

Attribute	Explanation
Transportable	Must be capable of application across a spectrum of complex systems engineering problems and contexts. The appropriateness (applicability) of the methodology to a range of circumstances and system problem types must be clearly established as the central characteristic of transportability.
Theoretically and Philosophically Grounded	Must have a linkage to a theoretical body of knowledge as well as philosophical underpinnings that form the basis for the methodology and its application. The theoretical body of knowledge for the SoSE Methodology is systems theory.
Guide to Action	Must provide sufficient detail to frame appropriate actions and guide direction of efforts to implement the methodology. While not prescriptively defining how execution must be accomplished, the methodology must establish the high level what's that must be performed.
Significance	Must exhibit the holistic capacity to address multiple problem system domains, minimally including contextual, human, organizational, managerial, policy, technical, and political aspects of a SOSE problem.
Consistency	Must be capable of providing replicability of approach and results interpretation based on deployment of the methodology in similar contexts. The methodology is transparent, clearly delineating the details of the approach for design, analysis, and transformation of the SOS.
Adaptable	Must be capable of flexing and modifying the approach, configuration, execution, or expectations based on changing conditions or circumstances – remaining within the framework of the guidance provided by the methodology, but adapting as necessary to facilitate systemic inquiry.
Neutrality	Attempts to minimize and account for external influences in application and interpretation. A methodology must provide sufficient transparency in approach, execution, and interpretation such that biases, assumptions, and limitations are capable of being made explicit and challenged within the methodology application.
Multiple Utility	Supports a variety of applications with respect to complex SOS, including, new system design, existing system transformation, and assessment of existing complex SOS initiatives.
Rigorous	Must be capable of withstanding scrutiny with respect to: (1) identified linkage/basis in a body of theory and knowledge, (2) sufficient depth to demonstrate detailed grounding in relationship to systemic underpinnings, including the systems engineering discipline, and (3) capable of providing transparent results that are replicable with respect to results achieved and accountability for explicit logic used to draw conclusions/interpretations.

The foundations of the SoSE Methodology are found in two primary aspects, namely (a) the theoretical and philosophical foundations for systems and (b) the seven perspectives of an enabling methodology shown in figure 3.

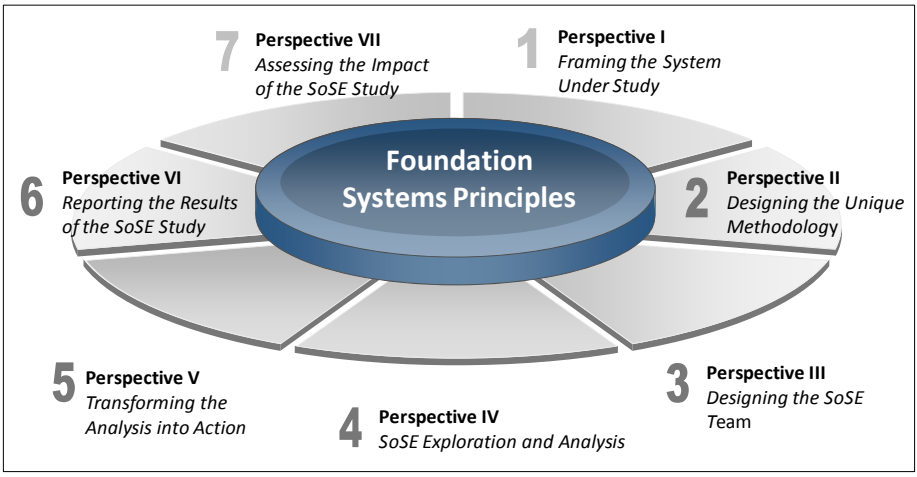


Fig. 3. The SoSE Methodology

First, the underlying theoretical and philosophical grounding are derived from systems theory. The principles, laws, and concepts central to the SoSE Methodology are from systems theory [32]. These principles, laws, and concepts are central to everything that follows in application of the SoSE Methodology to a specific problem domain. In effect, they define the thinking that supports following decision, action, and interpretation essential to effective SoSE. This sets the stage for a consistent approach to deployment of the SoSE Methodology by participants.

The second aspect of the SoSE Methodology is found in the seven perspectives that exist throughout a SoSE effort. Each perspective is:

- Essential to a holistic SoSE treatment of a problem area,
- Applied in iterative fashion throughout a SoSE project effort,
- Exists in relationship to all other perspectives, informing and informed by other perspectives,
- Can have a different priority at different times during an effort,
- Flexible in application, requiring tailoring depending on the context and problem domain, and
- Consists of detailed elements (that will vary in application) that serve to structure the application of the perspectives.

Each of the seven perspectives is briefly presented in the following paragraphs.

Perspective I: Framing the System under Study. This perspective is designed to rigorously structure the system problem, the contextual setting and environment within which the problem system exists. Key execution elements in this perspective include:

- Generalize the Wide Context for the System under Study – establish the circumstances, factors, conditions, and patterns that are characteristic of the situation surrounding the system of systems (SoS).
- Characterize the System under Study – understand the basic structure and characteristics of the system of systems under study, including the SoS's objectives, functions, environment, resources, components, and management.
- Characterize the Complex Nature of the System Domain under Study – establish the complex nature of the SoS and problem domain.
- Present the System Domain as Characteristically Complex - present the SoS under study as a complex systems problem.
- Frame the SoSE Problem - depict the problem situation by expressing the structure, elements of processes and the situation.
- Define Study Purpose, Reformulated Problem Statements and Objectives - clearly explain the nature, purpose, high-level approach, and objectives for the effort.
- Conduct Stakeholder Analysis - explicitly account for and address the multiple interests (rational and irrational, inside & outside) which can impact achievement of system objectives.
- Conduct Contextual Analysis - account for the set of circumstances, factors, conditions, values and/or patterns that are influential in constraining and enabling the SoS engineering process, the SoS solution/recommendation design, SoS solution/recommendation deployment considerations, and interpretation of outputs/outcomes stemming from the effort.

Perspective II: Designing the Unique Methodology. This perspective designs a unique methodology based on the problem and the problem context.

- Construct High-Level Design for the Study - construct a unique high-level methodology that will adequately support the study objectives and the SoS context. This must be compatible with the problem and problem context.
- Develop the Analytic Strategy - create the design for quantitative and qualitative exploration (data collection and analysis) necessary to understand and make decisions concerning the SoS under study.
- Develop Assessment Criteria and Plan - construct a set of measurable performance criteria that can be used during and after the problem study to ensure continued fit of problem, context, methodology and capability to meet study objectives.

Perspective III: Designing the SoSE Team. This perspective designs the team to undertake the SoSE study, taking into account the nature of the SoS problem and the team resources, skills, and knowledge that can be brought to bear for the problem.

- Assess Team Knowledge, Skills, and Abilities (KSA) - develop an inventory of team knowledge, skills, and abilities (KSA) that may be used in the study.
- Match Team KSA to the Analytic Strategy and Unique Methodology - based on the KSAs; establish assignments, roles, and expectations for the team in performing the study. Team expectations and selection of task leaders established.
- Establish Team Expectation Performance Assessment - construct a set of measurable performance criteria that can be used during and after the SoS problem study to evaluate the performance of the team.

Perspective IV: SoSE Exploration and Analysis. This perspective is designed to explore and conduct the emergent analysis by executing the analytic strategy and SoSE Management Plan (SoSEMP).

- Build the SoSE Management Plan (SoSEMP) - The SoSEMP defines how the SoS study will be organized, the structure of the team, and how the SoSE process will be designed to provide products that directly support the study goals and objectives requirements.
- SoSE Exploration and Analysis - conduct exploration and analysis for each of the study objectives by executing elements of the analytic strategy.

Perspective V: Transforming the Analysis into Action. This perspective is designed to transform the results of the emergent analysis by guiding implementation of the analysis recommendations.

- Define Implementation Goals, Objectives and Activities - clearly explain the nature of the implementation, purpose, high-level approach, and objectives necessary to support the desired SoS outputs and outcomes.
- Modify the SoSE Management Plan (SoSEMP) - add activities to the integrated schedule that ensures that the tasks from the implementation objectives tree are properly resourced to support the implementation goals, objectives, and activities.
- Implementation of the Exploration and Analysis Recommendations - change, modify, or construct processes for the SoS under study to implement recommendations.

Perspective VI: Reporting the Results of the SoSE Study. This perspective reports the results of the SoSE effort to capture the transformation of the analysis into action.

- Developing the Engineering Report - develop a coherent set of artifacts (data, analyses, correlations, etc) that can provide specific findings and recommendations that directly impact the SoS problem under study.
- Internal Evaluation of the Engineering Report - evaluate the study report using the set of measurable performance criteria previously developed.

Perspective VII: Assessing the Impact of the SoSE Study. This perspective is designed to assess the impact of the report on the real-world SoS problem under study.

- Evaluating the Initial Impact of the Engineering Report - evaluate the impact that the SoSE study report had on the real world system problem and its environment.
- Plan for Follow-up and Follow-through - evaluate the impact analysis and develop a set of actions to follow-up and follow-through on the analysis.

The SoSE Methodology provides a high level framework, philosophically based in systems theory, which offers a systemic, non-prescriptive guide to practitioners in SoSE. It is important to note three particular aspects concerning the SoSE Methodology.

1. The core of the methodology resides in the underlying foundation system principles. This establishes the systemic worldview that permits execution and interpretation of everything else that follows in the SoSE analysis. If this worldview is not correct, it is doubtful that the ensuing analysis will have the appropriate emphasis or effectiveness in execution.
2. The perspectives in the methodology are not intended to be approached as a linear stepwise set of perspectives to be accomplished independent or mutually exclusive of one another. On the contrary, there should be continual reframing and revisiting of perspectives and their execution elements as the SoSE analysis progresses. In fact, emphasis on particular perspectives or sequencing may, and probably will vary depending on: (1) the nature of the problem, (2) the sophistication of the participants in systemic SoSE experience/expertise, and (3) the nature of the context within which the problem is embedded.
3. It would be naïve to think that the methodology will have effective results if it is applied by those without sufficient grounding in the system fundamentals and/or approached as a prescriptive sequential set of steps that, if performed in a rote fashion, will generate successful SoSE outcomes. Only through appreciation of these limitations and considerations will the approach be capable of deployment as it has been intended.

In summary, the SoSE Methodology is a holistic framework that contains the theoretical foundation in systems theory that substantiates the use of perspectives and execution elements for addressing complex systems problems. It addresses not only the system internal challenges, but it addresses the context and external systems as well and changes the introspective processes into processes that take extrospective viewpoints into account as well.

3.5 Intelligence-Based Systems Engineering

How can intelligence-based systems help to realize intelligence-based systems engineering? Based on current research, the following aspects can help migrating TSE towards SoSE and ultimately intelligence-based systems engineering:

- <u>Increased used of semantic systems technology and ontological means within systems engineering</u>: In order to capture requirements in an unambiguous way and to better communicate them within teams, the increased use of semantic systems technology is needed. An example is transforming

requirements into logical expressions of a system's ontology. This allows not only to check their consistency, it also allows to directly communicate them with intelligent software components, as they understand logic, and also facilitates the validation and verification, as logical expressions are easier to evaluate than volumes of prosaic text. Controlled vocabularies that are distributed within the enterprise by Enterprise Lexical Services as described in [33] can be the first step. Such efforts need to be followed by defining thesauri and taxonomies that will support the lexical analysis between systems to detect redundancies and gaps regarding required versus available system capability. As shown in [34], for a full support of interoperable and composable system of systems higher levels of ontological support are necessary in support of data, process, and constraint engineering.

- Increased use of simulation technology in support of systems engineering: As stated by van Dam during his lecture at Stanford [35]: *"If a picture is worth a 1000 words, a moving picture is worth a 1000 static ones, and a truly interactive, user-controlled dynamic picture is worth 1000 ones that you watch passively."* That makes simulation very interesting not only for managers and decision makers, it also encourages the use of decision support simulation systems for systems engineering. While traditional decision support systems are used to compile useful information from raw data and documents that are distributed within the potentially very heterogeneous enterprise infrastructure, decision support simulation systems can be used to obtain, display and evaluate operationally relevant data in agile contexts by executing models using operational data exploiting the full potential of M&S and producing numerical insight into the behavior of complex systems. The idea of executable system architecture is already exploited. If simulation is integrated appropriately into the TSE and SoSE processes, communication will be increased and mistakes in the design can be identified earlier.

- Increased use of intelligent agent technology in support of systems engineering: As mentioned earlier, Yilmaz and Ören [3] dedicated a whole book to the synergisms of agent-directed simulation and systems engineering. With their ability to support the development of robust, fault tolerant, adaptive, self-optimizing, learning, social capable, autonomous, and agile solutions, they are a good match to support intelligence-based systems engineering. Replacing functions delivering the system capability by intelligent software agents delivering the system capability opens interesting new opportunities. Furthermore, in particular if combined with the rigorous use of ontological means to capture requirements and results within all phases, agents can support or even take over many routine jobs, freeing system engineers to focus on the more challenging phases of the process.

Although already powerful when applied stand alone, the support can be significantly increased when all methods are applied orchestrated in a distributed enterprise infrastructure. New information integration methods, based on semantic systems and derived standards, allow homogeneous support even in heterogeneous environments. Furthermore, new paradigms are exploited, like contribution of agents to the theoretical

and philosophical foundations for systems, validation of systems based on semantic means, and more. A research agenda for this domain is still an open requirement within the community that must comprise system engineers, computer scientists, modeling and simulation experts, and engineering managers. Contributions of all these domains are necessary to address the new challenges and enable new options. This discussion needs to be completed by integrating all project management challenges as well, such as extending the Body of Knowledge for Project Management [36] for intelligent-based systems engineering with special focus on applicability for risk management, value and cost management, and related tool support for a new generation of system architecture frameworks. Finally, these new insights need to be integrated into the education of future experts at colleges and universities. We are only at the beginning of this set of challenges.

However, the contributions to this book are a good start into this endeavor and represent a variety of common approaches towards intelligence-based systems engineering. They all have in common that the systems engineering community in general and the contributing authors in particular understand the need for the application of semantic systems, simulation technology and intelligent agents to overcome the introspective blocks and move towards agile and robust SoSE to enable intelligence-based systems engineering.

4 Contributions to These Topics within This Volume

This volume comprises a selection of state-of-the-art contributions to the topics discussed above, focusing on but not limited to semantic systems, simulation technology, and intelligent agents. The common theme is systems engineering and its relation to intelligence-based solutions. All authors were invited by the editors to submit their work based on their recognition in the field and the applicability of their findings to improve intelligence-based systems engineering, complex systems development, knowledge-based engineering, etc.

The work on *Semantic Systems for Intelligence-based Systems Engineering* by John Sowa describes the foundations for the use of ontology-based solutions. This chapter describes the foundations for expressing, sharing, and using knowledge and as such describing the essence of intelligence-based communication needs.

As one of the focal points of intelligence-based systems engineering is the support by intelligent machines, understanding within such machines is absolutely essential to be effective and efficient and to avoid significant errors by misinterpretations. The work on *Defining and Validating Semantic Machine to Machine Interoperability* by Claudia Szabo and Saikou Diallo is based on awarded academic research in this domain helps to better understand the challenges and provides first solutions of interest to the scholars and practitioners in the field.

An even more practical approach using a particular problem domain is given by Maria Vargas-Vera, Miklos Nagy, and Dietmar Jannach. They describe *An Approach to Knowledge Integration applied to a Configuration Problem* in which several of the theoretic results of the earlier chapters are applied.

With the chapter on *Multiple Worlds, a Framework for Modeling and Simulation Based Design* by Michele Fumarola, Mamadou Seck, and Alexander Verbraeck the

domain of simulation and the support of systems engineering is entered. Many engineering applications are philosophically dominated by a positivistic world view of physics-driven independent entities that interact under Newton's laws. The work on SoSE should that such a limited view is no longer sufficient, but multiple views need to be aligned, and simulations can help to do so.

A more technical aspects, namely that of how to integrate such simulation knowledge into the potentially heterogeneous enterprise infrastructure, is dealt with in the chapter *Distributed Simulation Using RESTful Interoperability Simulation Environment (RISE) Middleware* by Khaldoon Al-Zoubi and Gabriel Wainer. Web services have been identified in may papers as a potential universal integration tool, but the overhead is often the negative characteristic speaking against them. RESTful services avoid the overhead, allowing the integration of simulation capability.

The Discrete Event System Specification (DEVS) formalism for simulation systems emerged from systems engineering efforts two decades ago. Using the results of the simulation community to improve DEVS and merging various trends into the DEVS Unified Process (DUNIP) allows now the development of *Agile Net-centric Systems using DUNIP-Based Event Driven Architectures* described by Saurabh Mittal. As intelligent agents can be part of this simulation-based systems engineering approach, this chapter connects both worlds.

Starting with a practical view on *Systems Engineering and Conversational Agents,* James D. O'Shea introduced the intelligent agent topic explicitly. Conversational agents are known for their contributions to improving man-machine interfaces, including allowing for multiple representations in multiple modalities. The chapter also gives practical application examples for the semantic means introduced in the earlier chapter by Sowa.

The use of agents to enable innovative internal capabilities is demonstrated in Levent Yilmaz and C. Anthony Hunt's chapter *Toward Advanced Concepts and Generative Simulation Formalisms for Creative and Robust Discovery Systems*. This chapter shows how many of the characteristic properties identified above are implemented by agent-like structures and support intelligence-based systems engineering in new ways.

Jose J. Padilla, Saikou Y. Diallo, and Andres A. Sousa-Poza even go one step beyond. In their chapter *Establishing a Theoretical Baseline: Using Agent-Based Modeling to Create Knowledge* the idea to use intelligent agents as "research partners" is not only motivated, they report on successful applications of this idea to drive simulation beyond the scope of being pure computational activity, but becoming a knowledge generating activity.

Bringing agents out of the laboratory and to users of systems is described in *Artificial Intelligence Support for "The User around the Marketplace": Automatic Engineering of Interactive E-commerce Applications* by Martin Lopez-Nores, Yolanda Blanco-Fernandez, and Jose J. Pazos-Arias. This is an e-commerce application that can be used as an example who to embed systems into a broader, agile, and highly complex environment.

Another application example is given in the chapter on *Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies* by Raja Jurdak, X. Rosalind Wang, Oliver Obst, and Philip Valencia. As wireless sensor networks are tightly coupled regarding their development and operational environment. Each development phase has

immediate reactions in the use of the networks, making them emerging systems within a system of systems. Without the application of intelligence-based methods, the systems engineer could easily become overwhelmed.

The book ends with the chapter on *Enterprise Ontologies – Better Models of Business* by Ian Bailey, which is a critical review of claims, success, and perception thereof from a practitioner in the field of ontological systems and their application. With several years of experience from conducting projects, this chapter provides insights of interest to enthusiasts and critics in this field. He also applies a unique style and practical criticism of a domain that is often perceived wrongly as too academic and with no practical relevance.

In summary, this book should provide interesting material for scholars and practitioners. It comprises theoretic contributions as well as practical applications. It shows how the main contributing domains can and need to be combined in order to support intelligent-based systems engineering.

Moreover, the book in general and the chapters in particular also show that many niches are still open and gaps in the body of knowledge are waiting to be filled. In particular doctoral students should be able to use this book in search of valuable topics that need to be evaluated in detail and closed to promote the discipline of intelligence-based systems engineering.

By combining papers from industry experts with those of leading scholars and presenting them side by side, this book should be a valuable contribution to everyone being interested in the field of intelligence-based systems engineering and contribute to fruitful discussions on research agendas as well as applications in the field.

References

[1] Buede, D.M.: The Engineering Design of Systems, Models and Methods, 2nd edn. John Wiley and Sons, Inc., New York (2009)
[2] Ackoff, R.L.: From Data to Wisdom. Journal of Applied Systems Analysis 16, 3–9 (1989)
[3] Yilmaz, Ören (eds.): Agent-Directed Simulation and Systems Engineering. Wiley, Berlin (2009)
[4] Engstrom, E.: System Engineering – A Growing Concept. Electrical Engineering 76, 113–116 (1957)
[5] Goode, H., Machol, R.: Systems Engineering: An Introduction to the Design of Large-Scale Systems. McGraw-Hill Book Company, New York (1957)
[6] Schlager, K.: Systems Engineering – Key to Modern Development. IRE Trans., Prof. Gp. Eng. Management 3, 64–66 (1956)
[7] Hall, A.: A Methodology for Systems Engineering. D. Van Nostrand Company, Inc., Princeton (1962)
[8] Flagle, C., Huggins, W., Roy, R.: Operations Research and Systems Engineering. The Johns Hopkins Press, Baltimore (1960)
[9] Morse, P., Kimball, G. (eds.): Methods of Operations Research. John Wiley & Sons, New York (1951)
[10] McCloskey, J., Trefethen, F.: Operations Research for Management. Johns Hopkins Press, Baltimore (1954)

[11] Churchman, C., Ackoff, R., Arnoff, E. (eds.): Introduction to Operations Research. John Wiley & Sons, New York (1957)
[12] Machol, R. (ed.): Systems Engineering Handbook. McGraw-Hill Book Company, New York (1965)
[13] IEEE 1220. IEEE Standard 1220: Application and Management of the Systems Engineering Process. The Institute of Electrical and Electronics Engineers, New York (1998)
[14] EIA 632, Electronics Industries Alliance Standard 632: Processes for Engineering a System.Electronics Industries Alliance, Arlington (1999)
[15] Royce, W.W.: Managing the Development of Large Software Systems. In: IEEE West Conference, pp. 328–338. IEEE, Los Alamitos (1970)
[16] Forsberg, K., Mooz, H.: The Relationship of System Engineering to the Project Cycle. In: Proceedings of the 1st Annual Conference of the National Council on Systems Engineering, Chattanooga, TN (1991)
[17] Boehm, B.: A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes 11(4), 14–24 (1986)
[18] Davenport, T.: Putting the Enterprise into the Enterprise System. Harvard Business Review 76(4), 121–131 (1998)
[19] Ackoff, R.: The Systems Revolution. Long Range Planning 7(6), 2–20 (1974)
[20] Maier, M.: Architecting Principles for Systems-of-Systems. Systems Engineering 1(4), 267–284 (1998)
[21] Sage, A., Cuppan, C.: On the Systems Engineering and Management of Systems of Systems and Federations of Systems. Information, Knowledge, Systems Management 2(4), 325–345 (2001)
[22] Blanchard, B., Fabrycky, W.: Systems Engineering and Analysis, 5th edn. Prentice Hall, Upper Saddle River (2010)
[23] Kosiakoff, A., Sweet, W.: Systems Engineering Principles and Practice. John Wiley and Sons, Hoboken (2003)
[24] Sage, A., Armstrong, J.: Introduction to Systems Engineering. Wiley-Interscience, New York (2000)
[25] Ackoff, R.: The Future of Operational Research Is Past. Journal of the Operational Research Society 30(2), 93–104 (1979)
[26] Keating, C., Rogers, R., Unal, D., Dryer, D., Sousa-Poza, A., Safford, R., Peterson, W., Rabadi, G.: System of Systems Engineering. Engineering Management Journal 15(3), 35–44 (2003)
[27] Keating, C.B.: Emergence in System of Systems. In: Jamshidi, M. (ed.) Systems of Systems Engineering: Innovations for the 21st Century, pp. 169–190. John Wiley and Sons, Hoboken (2009)
[28] Adams, K. MacG., Keating, C.B.: SoSE Methodology Rev 0.2. NCSoSE Technical Report 009-2009.National Centers for System of Systems Engineering, Norfolk, VA (2009)
[29] Mitroff, I., Featheringham, T.R.: On Systematic Problem Solving and the Error of the Third Kind. Behavioral Science 19(6), 383–393 (1974)
[30] Mosteller, F.: A K-sample Slippage Test for an Extreme Population. Annals of Mathematical Statistics 19(1), 58–65 (1948)
[31] Checkland, P.: Systems Thinking, Systems Practice. John Wiley & Sons, New York (1993)
[32] Adams, K. MacG.: Systems principles: Foundation for the SoSE Methodology. International Journal for System of Systems Engineeing (in press, 2010)

[33] Durham, J.: Enterprise Lexicon Services. In: Proc. of 11th International Business Rules Forum, Orlando, FL (2008)
[34] Tolk, A., Diallo, S.D., King, R.D., Turnitsa, C.D.: A Layered Approach to Composition and Interoperation in Complex Systems. In: Complex Systems in Knowledge based Environments: Theory, Models and Applications. SCI, vol. 168, pp. 41–74 (2009)
[35] van Dam, A.: Education: the unfinished revolution. ACM ComputingSurveys(CSUR) 31(4), Article No. 36 (1999)
[36] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 4th edn. Project Management Institute Series on Project Management (PM) and PM Standards, Newton Square (2008)

Chapter 2
Future Directions for Semantic Systems

John F. Sowa

Abstract. For over thirty years, the complexity of knowledge acquisition has been the greatest obstacle to widespread use of semantic systems. The task of translating information from a textbook to a computable semantic form requires the combined skills of a linguist, logician, computer scientist, and subject-matter expert. Any system that requires its users to have all those skills will have few, if any, users. The challenge is to design automated tools that can combine the contributions from multiple experts with different kinds of skills. This article surveys systems with different levels of semantics: lightweight, middleweight, and heavyweight. Linked data systems with lightweight semantics are easy to develop, but they can't interpret the data they link. The heavyweight systems of traditional AI can perform deep reasoning, but they place too many demands on the knowledge engineers. No one can predict what innovations will be discovered in the future, but commercially successful systems must satisfy two criteria: first, they must solve problems for which a large number of people need solutions; second, they must have automated and semi-automated methods for acquiring, analyzing, and organizing the required knowledge.

1 The Knowledge Acquisition Bottleneck

Computers can process numbers, data structures, and even axioms in logic much faster than people can. But people take advantage of background knowledge that computers don't have. Hao Wang (1960), for example, wrote a program that proved all 378 theorems in propositional and first-order logic from the *Principia Mathematica*. On a slow vacuum-tube computer, Wang's program took an average of 1.1 seconds per theorem — far less time than Whitehead and Russell, the two brilliant logicians who wrote the book. But the theorems in the *Principia* require a negligible amount of built-in knowledge — just five axioms and a few rules of inference. The computer Wang used had only 144K bytes of RAM, but that was sufficient to store the rules and axioms and manipulate them faster than professional logicians.

During the 1970s and '80s, rule-based expert systems and programs for processing natural languages became quite sophisticated. But most applications required an enormous amount of background knowledge to produce useful results. Knowledge engineers and subject-matter experts (SMEs) had to encode that knowledge in formal logic or some informal rules, frames, or diagrams. The experts were usually highly paid professionals, such as physicians or geologists, and the knowledge engineers required long years of training in logic, ontology, conceptual analysis, systems design, and methods for interviewing the experts. For critical applications, the investment in knowledge acquisition produced significant results. For other applications, the cost of defining the knowledge might be justified, but the AI tools were not integrated with

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 23–47.
springerlink.com

commercial software. Furthermore, most programmers did not know how to use AI languages and tools, and the cost of training people and adapting tools was too high for mainstream commercial applications.

During the 1990s, vast amounts of data on the World Wide Web provided raw data for statistical methods. Machine learning, data mining, and knowledge discovery found patterns more cheaply and often more accurately than rules written by experts. The more challenging goal of language understanding was largely abandoned in favor of statistical methods for information retrieval and information extraction. Although statistical methods are useful, they don't generate a semantic representation suitable for further reasoning or for explanations in ordinary language.

At the beginning of the 21st century, the Semantic Web adapted the AI technologies of the 1980s to the vast resources of the World Wide Web. But the mainstream commercial software, which had never been integrated with AI technology, was just as isolated from the Semantic Web. For most programmers and web masters, the languages and tools of the Semantic Web were unfamiliar, there was no migration path from conventional software to the new technology, and the task of knowledge acquisition was just as difficult as ever.

The complexity of knowledge acquisition increases with the complexity of the semantics, the amount of detail that must be specified, and the interdependencies among different aspects of the knowledge base. To relate different methods, this article uses a three-way distinction: *heavyweight* semantics is represented in a formal logic with detailed axioms that can support extended reasoning; *middleweight* semantics is based on formal or informal notations that support a modest amount of reasoning, but with less complexity than heavyweight semantics; *lightweight* semantics uses tags to classify information, to check simple constraints on types and connections, but not to perform extended reasoning.

Many systems use variations of these three kinds of semantics. Section 2 of this article uses the distinction to compare systems for natural language processing (NLP). Section 3 applies it to systems for reasoning and problem solving. Section 4 analyzes the Semantic Web technologies in these terms. Section 5 shows how the VivoMind Language Processor (VLP) uses all three kinds of semantics for language analysis and reasoning. The concluding Section 6 discusses the requirements for commercially successful systems and the ways of using AI technology to design and implement them.

2 Natural Language Processing

Documents that people write to communicate with other people are rarely as precise as logic. Yet people can read those documents and relate them to formal notations for science, mathematics, and computer programs. They can derive whatever information they need, reason about it, and apply it at an appropriate level of precision. That flexibility is essential for a system of knowledge acquisition — automated, semi-automated, or at least computer assisted. For the past half century, AI researchers and computational linguists have tried to achieve that goal.

Some of the most successful NLP systems use lightweight semantics. One of the first was the Georgetown Automatic Translator (GAT), for which research was terminated in 1963. Under the name Systran, it became the most widely used machine-translation

system in the 20th century. A version is still available on the web under the name Babelfish. For each pair of languages to be translated, Systran uses a large dictionary of equivalent words and phrases. The computer processing consists of a limited amount of movement and adjustment to accommodate the syntactic differences between each language pair (Hutchins 1995). Constructing those dictionaries by hand requires many person-years of effort. With the large volumes of documents available on the web, statistical methods for detecting and aligning equivalent pairs have become more widely used. Although these techniques are useful for MT, they don't produce a semantic representation that can be used for reasoning. Hybrid systems that combine statistics with shallow parsing and templates are widely used for information extraction, but Hobbs and Riloff (2010) noted that such systems have reached a barrier of about 60% accuracy in recall and precision.

The most sophisticated NLP systems use heavyweight semantics based on some version of logic. Typical systems have two distinct levels: syntactic analysis to generate a parse tree and semantic interpretation to map the parse tree to a logical form. But after forty years of research, no system based on that approach can read one page of a high-school textbook and use the results to solve the problems as well as a B student. Even pioneers in logic-based methods have begun to doubt their adequacy. Kamp (2001), for example, admitted that "the basic concepts of linguistics — and especially those of semantics — have to be thought through anew" and "many more distinctions have to be drawn than are dreamt of in current semantic theory."

To understand the issues, consider the combination of syntax, semantics, and database structure necessary to analyze a question and answer it. As an example, the Transformational Question Answering system (Petrick 1981) analyzed English questions and used middleweight semantics about the subject matter to map English to and from logic. TQA also used heavyweight semantics to map logic to and from the SQL query language, which has the expressive power of first-order logic. The parser evolved from a research project that Petrick (1965) designed for his PhD dissertation under Chomsky's supervision. After joining IBM, Petrick collaborated with other researchers to develop TQA as an English front-end to a relational database.

To evaluate TQA's potential, IBM management wanted to test it on actual users. The nearby city of White Plains served as a test case. During the 1974 gasoline shortage, city officials had to search land-use records by hand to find the locations of all gas stations so that police could go there to direct traffic. Later, the records were stored on a computer, but somebody had to write a new program and print a new report for every question. Every follow-on question required another program. In 1978, the IBM researchers loaded the land-use files on a relational database at Yorktown, customized TQA to access the database, and connected it to a dedicated terminal in the city hall.

For a full year, the White Plains officials and land-use planners could type English questions to TQA and get immediate answers. Of 788 questions typed during the year, TQA answered 65% correctly and failed to parse 35%. For most parsing failures, the users rephrased the sentence in a way that TQA could answer. Occasionally, they called the IBM developers for help. Overall, the users loved it. They were unhappy when the trial period ended, and IBM unplugged the terminal (Damerau 1981).

Following are some questions that TQA answered correctly:

```
What is the total area of the parcels in ward 6 block 72?
How many two family houses are there in the Oak Ridge
Residents Assn?
Where are the apartment dwellings which have more than
50 units
which are more than 6 stories high on Lake St?
```

The TQA test showed that subject-matter experts, who had no training in programming or database software, could effectively use an English front-end to conventional software. It also showed the kind of syntax and semantics that was needed to customize a language processor for each application. The syntax of the phrase *ward 6 block 72* is familiar to the SMEs, but it is rare in ordinary English. The TQA developers added grammar rules for many such phrases before the test period. During the test, they analyzed the questions that TQA failed to parse correctly and revised the grammar to accommodate them. The TQA users also learned to adjust their grammar to accommodate the parser.

The test version of TQA also generated a rudimentary *echo* that showed how each question was parsed. Unfortunately, some echos used syntax that the parser failed to recognize when the users typed them back. Mueckstein (1983) later designed Q-TRANS to generate an echo that TQA could always parse. Following is a question processed by TQA:

```
What parcels in the R5 zone on Stevens St. have
greater than 5000 sq. ft.?
```

TQA translated that question to the following SQL:

```
SELECT UNIQUE A.JACCN, B.PARAREA
FROM ZONEF A, PARCFL B
WHERE A.JACCN = B.JACCN
AND B.STN = 'STEVENS ST'
AND B.PARAREA > 5000
AND A.ZONE = R5;
```

Q-TRANS translated that SQL to the following echo:

```
Find the account numbers and parcel areas for lots that have
the street name STEVENS ST, a parcel area of greater than
5000 sq. ft., and zoning code R5.
```

These examples show the kind of customization required by any processor that maps natural language queries to and from a computer system: first, an ontology of the entities, relations, and constraints in the subject matter; second, a lexicon that maps words and phrases to and from the ontology; third, specialized syntax for patterns that are rare in ordinary language; and finally, mappings of the ontology to computer formats and interfaces. To simplify the task, Damerau (1988) designed a tool to

enable "database administrators to generate robust English interfaces to particular databases without help from linguistic experts." IBM management, however, decided that it was too complex for most customers and the potential market was too small to be profitable. Therefore, they canceled the TQA project.

TQA was one of many NLP systems that demonstrated usefulness for some applications, but were not commercially successful. Systems with lightweight semantics, such as Systran, have been more successful. Some of the most successful are search engines that index documents by the words they contain without using any explicit semantics. Google improved the search with statistical methods for deriving some implicit semantics from the patterns of cross references. In general, systems based on lightweight semantics depend on the readers to use their background knowledge to fill in the gaps, but no human army could process the huge volumes of data on the web. For some applications, statistical methods can filter out much of the irrelevant data, but even a thousand-to-one reduction in petabytes still leaves terabytes. NLP systems with heavyweight semantics are necessary to interpret the details.

3 Reasoning and Problem Solving

Since the 1950s, research in AI explored a wide range of techniques from neural networks to formal logic. But the classical AI paradigm combines some knowledge representation language with some formal or informal methods of reasoning. Two classical system of radically different sizes illustrate the problems and the range of possible solutions: the very large Cyc system, which shows the power of a general-purpose, heavyweight semantics; and a simpler system designed for online sales, which shows the ease of use of middleweight semantics combined with semi-automated methods for knowledge acquisition.

The expert systems of the 1980s showed that the level of expertise increased as more rules and facts were added. Some AI experts estimated that a human level of intelligence could be achieved with less than a million concepts encoded in some computable form. Lenat and Feigenbaum (1987) summarized the arguments:

- Lenat estimated that encyclopedic coverage of the common knowledge of typical high-school graduates would require 30,000 articles with about 30 concepts per article. That justified the Cyc Project, whose name comes from the stressed syllable of *encyclopedia*.
- The Japanese Electronic Dictionary Research Project (EDR) estimated that the knowledge of an educated speaker of several languages would require about 200K concepts represented in each language.
- Marvin Minsky noted that less than 200,000 hours elapses between birth and age 21. If each person adds four new concepts per hour, the total would be less than a million.

For the Cyc Project, they concluded that a knowledge base "of under a million frames" could be constructed in one decade with $50 million and less than two person-centuries of work.

The original version of Cyc was an informal system of frames with heuristic procedures for processing them (Lenat & Guha 1990). But as the knowledge base grew, the dangers of contradictions, spurious inferences, and incompatibilities became critical. As a result, the frames had to be more highly structured, and the procedures became more systematic and tightly controlled. Eventually, the CycL language and its inference engines were rewritten as a superset of first-order logic with extensions to support defaults, modality, metalanguage, and higher-order logic. The most significant innovation was a context mechanism for partitioning the knowledge base into a basic core and an open-ended collection of independently developed *microtheories* (Guha 1991).

After the first 25 years, Cyc grew far beyond its original goals: 100 million dollars had been invested in 10 person-centuries of work to define 600,000 concepts by 5 million axioms organized in 6,000 microtheories. Cyc can also access relational databases and the Semantic Web to supplement its own knowledge base. For some kinds of reasoning, Cyc is faster and more thorough than most humans. Yet Cyc is not as flexible as a child, and it can't read, write, or speak as well as a child. It has not yet achieved the goals of the "sweeping three-stage research programme" outlined by Lenat and Feigenbaum in 1987:

1. "Slowly hand-code a large, broad knowledge base."
2. "When enough knowledge is present, it will be faster to acquire more through reading, assimilating data bases, etc."
3. "To go beyond the frontier of human knowledge, the system will have to rely on learning by discovery, carrying out research and development projects to expand its KB."

The first goal has been achieved. The second goal was far more difficult than expected. Cyc cannot yet read a textbook and map the knowledge to CycL, and it can only access external databases whose metadata or ontology has been mapped to CycL concepts. The third goal is still a dream.

Even though Cyc did not achieve all the original goals, it remains the world's largest body of knowledge represented in logic and suitable for detailed deduction. For any given problem, Cyc automatically selects the required axioms and an inference method that is suitable for that problem. The Cyc tools can also be used as a development platform for defining axioms that can drive other inference engines. As an example, Peterson et al. (1998) designed a knowledge compiler to translate a subset of axioms from CycL to more restricted logics that drive a deductive database: Horn-clause rules for the inference engine, and database constraints stated in SQL WHERE-clauses. For a sample problem, they extracted 5532 axioms (about 1% of the five million axioms in the Cyc knowledge base). Of those axioms, 84% could be translated directly to Horn-clause rules for performing inferences. The remaining 16%, which required full first-order logic, were translated to update constraints in SQL to ensure that the database is always consistent with the axioms.

For the first dozen years, the Cyc Project focused on research, but the academic research was not easy to commercialize. Later, they gradually increased the time devoted to applications. As a result, Cyc earned more money from applications in the years 2008 to 2010 than in the previous 24 years. Some of the fastest growing applications are in medical informatics. At the Cleveland Clinic, about 1700 axioms

from the general Cyc ontology are used to understand and respond to a typical query. The applications show considerable promise, but most application programmers find it difficult to adapt their software and databases to the Cyc knowledge base. Although Cyc is primarily a reasoning system, it also supports an English interface, which requires customization similar to TQA.

In contrast with Cyc, which has been in continuous development for over 25 years, smaller AI systems can be implemented much faster. As an example, Tesco, a large UK retailer, sells a variety of goods, ranging from groceries to electronic equipment. For their online branch, Tesco.com, they wanted a flexible system that employees could update dynamically. One software vendor designed a system based on RDF and OWL, but Tesco employees could not modify it. Calling an OWL expert for every update would be too slow, and hiring one for every store would cost too much. They needed a simpler system that current employees could modify without lengthy and costly training.

As an alternative, Gerard Ellis, an employee of the vendor, designed and implemented a prototype of a more flexible system in just a few weeks. Tesco liked it, and the complete system was delivered to them in a few months (Sarraf & Ellis 2006). Unlike the heavyweight semantics of Cyc, which requires professional knowledge engineers to update and modify, the Tesco system had middleweight semantics that could be updated by Tesco employees who had no training in AI, logic, or ontology. Automated tools could also check that the knowledge base is consistent and help Tesco employees correct any errors. The reason why Ellis could implement the new system so quickly is that he had spent a dozen years in developing a toolkit of AI software and related technologies (Ellis et al. 1994). To replace the system that used RDF+OWL, he put together the following components:

- Conceptual graphs (CGs) as the internal knowledge representation with basic tools for storing, retrieving, and manipulating CGs. Communication with other components was based on the Conceptual Graph Interchange Format (CGIF).
- A version of controlled English (CE) as the notation for subject-matter experts (SMEs) with tools to map CE to and from CGIF.
- Ripple-down rules (RDR) as the technology for learning, reasoning, and maintaining the knowledge base with a mapping to and from CGIF.

The SMEs were Tesco employees, who used controlled English to edit the rules, get an explanation of how a conclusion was derived, and correct any errors by typing the conclusion that should have been derived. This application was designed for selling groceries and later adapted for the electrical and wine departments.

Ripple-down rules are derived from a decision tree that is compiled to a nest of if-then-else statements (Quinlan 1993; Compton et al. 2006). The raw data for deriving a decision tree is a set of *cases*, each of which is described by one or more conditions and one or more conclusions. Each link of the tree is labeled with one condition, and each leaf (end point) shows one or more conclusions implied by the conjunction of all the conditions leading to that leaf. To derive a complete and consistent tree, the algorithms detect possible conflicts, show the conflicting cases, and request additional information to resolve the conflicts. For major updates, the algorithms can derive a new tree from the raw data, but for minor editing, they can make local changes to the

tree. For the Tesco application, SMEs describe the cases by CE statements, and the system generates the rules. Following are some rules derived for the grocery application:

- **If a television product description contains "28-inch screen", add a screen_size attribute_inches with a value of 28.**
- **If a recipe ingredient contains butter, suggest "Gold Butter" as an ingredient to add to the basket.**
- **If a customer buys 2 boxes of biscuits, the customer gets one free.**
- **If the basket value is over £100, delivery is free.**
- **If the customer is a family with children, suggest "Buy one family sized pizza and get one free".**

The RDR rule format has proved to be convenient for SMEs from a wide range of backgrounds, especially medical informatics. Compton et al. (2006) describe an application developed by pathologists who used RDR tools to derive a knowledge base of 16,000 rules from a set of 6 million cases. But RDR is just one of a large class of tools for *case-based reasoning*, which overlap methods of machine learning. Some of them, like RDR, draw sharp distinctions that can be expressed in a subset of logic. Others use statistics, clustering algorithms, neural networks, and fuzzy logic for learning and reasoning from cases without sharp boundaries. Still others use analogies, which can derive sharp or fuzzy distinctions under varying conditions.

In summary, a large ontology such as Cyc does not, by itself, lead to successful commercial applications. A great deal of work on customization and knowledge acquisition is necessary to adapt Cyc to more conventional software. The Tesco.com application shows how systems with middleweight semantics can often simplify the task of knowledge acquisition. But the people who develop systems that SMEs find easy to use require advanced education and a toolkit of sophisticated software. With appropriate tools and methodologies, a convenient front-end could make any system easier to use. A challenging research goal is to develop an integrated knowledge acquisition system that could support both AI and conventional software.

4 Semantic Web

The Semantic Web was inspired by Tim Berners-Lee, but it was designed by a committee. It evolved from a keynote speech at the First World Wide Web Conference (Berners-Lee 1994):

> Adding semantics to the Web involves two things: allowing documents which have information in machine readable forms, and allowing links to be created with relationship values.

At this level of detail, nobody could object. But the speech didn't describe the machine readable formats, the kinds of relationship values, the logical operations on

those values, or any influence from the 40 years of research on semantics in artificial intelligence, computational linguistics, and software engineering. The W3 Consortium, which met for the first time at that conference, took charge of the design. Although every branch and nearly every aspect of computer science was represented by one or more members of the W3C, a design by committee is a compromise of good people pulling in different directions for good, but often conflicting reasons. By 2001, some components of the Semantic Web had been specified by W3C recommendations, but the only consensus on the overall architecture was the so-called "layer cake" at the left of Figure 1.

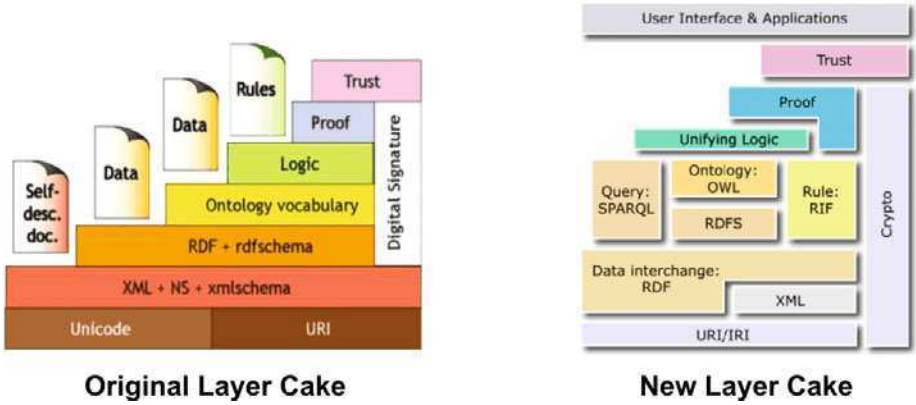


Fig. 1. The architectural layer cakes for the Semantic Web

The new layer cake on the right of Figure 1 developed toward the end of the decade. The differences between the two cakes show significant changes in the evolution of the Semantic Web:

1. The original cake had clear layers that built on one another. The new cake adds more boxes, as one might expect, but it also lets some layers dip beneath their earlier foundations.
2. The Resource Description Framework (RDF), which was defined by XML Schema, now extends below XML. One extension allows RDFa, which consists of single attribute tags, to be placed in any XML area. But other variations have been used or proposed.
3. The digital signature pipe, which was supposed to be based on XML, is replaced by a cryptography pipe that goes beneath all layers, since XML and the browsers that process it cannot guarantee security.
4. The ontology vocabulary layer has been replaced by four loosely related boxes. The Web Ontology Language (OWL) could be used with the other boxes, but applications that use them are more likely to avoid OWL.
5. The logic layer has shrunk to a smaller box for a unifying logic, since the components beneath it use some sort of logic, but each of them has its own independently defined semantics.

6. Proof rests on top of the unifying logic, but it also dips beneath it to the Rule Interface Format (RIF).
7. Trust is the only layer that has not changed, primarily because nobody really knows how to achieve it.

The evolution of these components can clarify their interrelationships and suggest future directions. RDF began with a diagram by Tim Berners-Lee that showed how the links between documents formed a Giant Global Graph. The detailed specification evolved from an internal dispute in the Cyc Project. The director, Doug Lenat, wanted a single unified CycL language, but the associate director, R. V. Guha, considered CycL too complex for most users. Guha wanted a simpler subset of logic that would allow SMEs to read, write, and edit at least some of the knowledge base. Whatever the issues may be, Guha left Cyc to join Apple, where he designed a language called the Meta Content Framework (MCF). He later collaborated with Tim Bray to represent MCF in XML terms (Guha & Bray 1997). MCF was renamed RDF when it became a W3C recommendation, and Bray promoted it enthusiastically. But he later expressed serious concerns about the way it had developed (Bray 2003):

> Conceptually, nothing could be simpler than RDF. You have Resources, which by definition are identified by URIs. The resources have Properties, which by convention are identified by URIs. The properties have Values, which can be strings or numbers or Resources. Everything's a triple: (Resource, Property, Value)...
>
> Speaking only for myself, I have never actually managed to write down a chunk of RDF/XML correctly, even when I had the triples laid out quite clearly in my head. Furthermore — once again speaking for myself — I find most existing RDF/XML entirely unreadable. I think the Semantic Web people have taken on a job that's already tough, and are adding difficulty, and increasing the probability of failure, by sticking to the currently broken RDF/XML syntax.

Various tools provide more readable notations for triples, which are translated to and from the XML format. A popular alternative is the JavaScript Object Notation (JSON), which can represent an RDF triple as **`[R, P, V]`**. A collection of property-value pairs for the same resource could be written more compactly as **`{P1:V1, P2:V2, ..., Pn:Vn}`**. JSON is a humanly readable notation that is directly processed by JavaScript.

Although MCF had only a modest amount of logic, Guha and Bray noted that it could be used to describe its own structure and datatypes: "This self-description allows MCF to be its own schema definition language. This in turn allows MCF to be dynamically extended by an author or application." That principle was carried over to RDF: the base RDF notation has no built-in ontology, and RDF Schema (RDFS) contains a metalevel ontology for stating constraints on types and relations. The logic base (LBase) of RDF is simple, but quirky (Guha & Hayes 2002; Hayes 2003). A triple with three names (URIs or literals) represents a relation R applied to two arguments A and B: R(A,B). A collection of triples represents a conjunction:

$$R1(A1,B1) \wedge R2(A2,B2) \wedge R3(A3,B3).$$

But any argument slot could contain a URI that specifies a relation. In fact, a relation could even be applied to itself:

R1(A1,R2) ∧ R2(R1,B2) ∧ R3(R2,R3).

Furthermore, RDF allows "blank nodes" that represent anonymous entities: a triple of the form R(A,_) would say that something A is related by the relation R to some unspecified resource. In effect, a blank node represents an existentially quantified variable: $(\exists x)$R(A,x). If a blank node happens to occur in the relation slot, then the quantifier ranges over relations: $(\exists r)r$(A,B). That triple would say that there exists an unspecified relation r between A and B. The LBase semantics shows that this logic is consistent, but it allows combinations that go beyond the usual first-order logic.

To support reasoning, some version of logic with suitable rules of inference is required. For 2400 years, the most widely used version for representing and reasoning about ontology has been Aristotle's syllogisms. Description logics (DLs) are a family of formalisms that extend Aristotle's logic with features such as cardinality constraints and Boolean combinations of categories. McGuinness et al. (2002) showed how two description logics, DAML and OIL, could be adapted to the RDF notation to form OWL. But the combination of DLs with RDF exacerbated old controversies and created new ones.

For thirty years, the DL community has been divided between practitioners who use highly expressive languages to implement applications and theoreticians who prove theorems about computational complexity. The Loom and PowerLoom systems, for example, have been widely used for practical applications (MacGregor 1991; Chalupsky et al. 2006). But the theoreticians ignored PowerLoom because it's too expressive: it's possible to state undecidable problems. Yet every major programming language is undecidable, and programmers want more expressiveness, not less. For any language, reducing the expressive power does not make the easy problems easier to define or faster to solve. It just makes the hard problems impossible to express. The PowerLoom language became undecidable because the users asked for more expressive power; none of them asked for decidability.

CycL is an extremely expressive language, but undecidability has never been an obstacle. On the contrary, OWL has created more obstacles by its draconian measures to enforce decidability: the constraints on OWL cause all models to be tree structured. A benzene molecule, for example, has a ring of six carbon atoms. In OWL, it's not possible to state or imply that they form a ring, because a ring is not a tree. In Cyc, a knowledge engineer can choose tree models when appropriate and thereby guarantee decidability. Chemists, architects, and airplane designers, however, require graphs with cycles. They have developed highly efficient ways of representing them in both procedural and logic-based languages. They can represent them in Cyc, but not in OWL.

Computational complexity is a critical issue, and software engineers have developed ways of addressing it. Structured programming, design patterns, and their associated methodologies help programmers in several ways: provide a toolkit of useful, repeatable techniques; guide a design toward structures that are known to be safe, decidable, and efficient; and support tests for detecting problematical aspects. Yet all these methods are optional. They don't stop creative programmers from exploring innovative ways of using their highly expressive languages to invent new

patterns. Software engineers also observe a time-honored principle: "Premature optimization is the root of all evil." Fine tuning one component is irrelevant and even counterproductive before its relationships to all other components are thoroughly understood.

The fragmentation of the ontology layer in Figure 1 is the result of developing the components independently without considering their roles in an integrated system. In the 1980s, description logics were used as one part of a hybrid system: a DL would define concept types in the terminology component (T-Box) while a more general logic used those types in the assertional component (A-Box). By giving priority to definitions in the T-Box, the hybrid had a modal effect of making T=box statements necessarily true with respect to the statements in the A-Box. The hybrid structure also allowed tradeoffs that improved efficiency while simplifying the task of knowledge acquisition. Some systems had three levels: a T-Box for defining concept and relation types, and an A-Box that was split between a rule-based reasoning system and a database for storing ground-level facts. Cyc has similar levels internally, but all levels use different subsets of the very expressive CycL notation.

That point about databases raises another issue: commercial web sites usually include relational DBs, which are as important as any component in the layer cakes. Some people claim that RDBs are obsolescent, but they still run the world economy. Furthermore, vendors of RDBs provide SPARQL interfaces to the tables, and vendors of triplestores provide SQL interfaces to theirs. Back in the 1980s, query systems like TQA might have been successful if their semantics could be derived as a byproduct of the database design methodology. In fact, Figure 2 shows a proposal from that era that could have and should have supported such systems (Tsichritzis & Klug 1978).

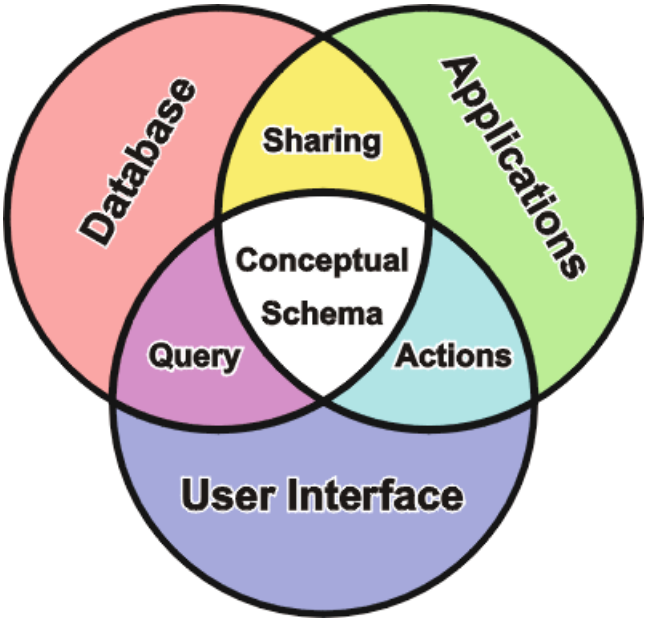


Fig. 2. The ANSI/SPARC conceptual schema

The conceptual schema at the center of Figure 2 represents semantics, which includes logic and ontology. Unlike the layer cakes, Figure 2 makes semantics the foundation and relegates syntactic formats to the periphery. The same conceptual schema could represent the semantics of data organized in tables, networks, or hierarchies. And the meaning would remain constant under mappings to different application programs or user interfaces. Instead of leaving semantics for the comments, tools based on Figure 2 could begin with words and phrases the SMEs understand and create lexicons for systems like TQA. Unfortunately, the final C of SPARC represented a committee with conflicting experts pulling in different directions. The conceptual schema remained a technical report, and not a standard.

In summary, the goals of the Semantic Web were good, but the emphasis on syntax was a distraction. The strategy must begin with semantics: knowledge representation, reasoning methods, and knowledge acquisition. Guha had hoped to design a simpler notation than CycL, but the syntactic details of the components — RDF, RDFS, OWL, RIF, and SPARQL — dwarf the CycL manual in size and complexity. For special purposes, the semantics of a notation like SKOS (Simple Knowledge Organization System) is defined by a mapping to a larger component like OWL. That mapping enables OWL applications to use knowledge entered through SKOS. But there is no unified semantics that can define, relate, and share knowledge among all the components. Cyc, for example, allows each item of knowledge to be entered once and be reused in as many different ways as necessary. But the components of the layer cake have overlapping semantics, they tend to compete with one another, and any sharing that might occur is on an ad hoc basis. A coherent strategy should build on a unified semantic foundation, simplify knowledge acquisition, and promote the original goals of sharing and reusing knowledge among all systems connected through the WWW.

5 Language Analysis and Reasoning

The Holy Grail of knowledge acquisition is to design computer systems that can read a textbook and map it to logic. But that task is difficult, even for logicians. Hans Kamp, for example, was a graduate student at UCLA when he got a summer job at the RAND Corporation to translate an article from the *Scientific American* to logic. His thesis advisor, Richard Montague (1970), had claimed that

> There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed, I consider it possible to comprehend the syntax and semantics of both kinds of languages within a single natural and mathematically precise theory.

But when Kamp tried to translate the article from English to logic, he found that the mapping was far more difficult than anyone had thought. Some factual statements were fairly straightforward. But most sentences required new ontological assumptions, new translation rules, and sometimes *ad hoc* decisions about word senses and anaphoric

references. That experience led him to develop *discourse representation theory* (DRT) on formal principles that went beyond Montague's (Kamp & Reyle 1993). Other linguists and logicians added more details and variations. But by the early 21st century, most of them agreed with Kamp that the basic principles "have to be thought through anew."

Even though a direct translation from language to logic or other computable form is not always possible, the opposite mapping from a formal language to a natural language is much easier and more systematic. It is also possible to relate formal programs to the documents that describe them, but the task requires a looser kind of analogy rather than a direct translation. The VivoMind Analogy Engine (VAE), for example, was used in legacy re-engineering: analyze and compare the programs and documentation of software in daily use that was up to forty years old (LeClerc & Majumdar 2002; Sowa & Majumdar 2003). Although the documents specified how the programs were supposed to work, nobody knew what errors, discrepancies, and obsolete business procedures might be buried in the code. Following is an excerpt from one of them:

> The input file that is used to create this piece of the Billing Interface for the General Ledger is an extract from the 61 byte file that is created by the COBOL program BILLCRUA in the Billing History production run. This file is used instead of the history file for time efficiency. This file contains the billing transaction codes (types of records) that are to be interfaced to General Ledger for the given month.
>
> For this process the following transaction codes are used: 32 — loss on unbilled, 72 — gain on uncollected, and 85 — loss on uncollected. Any of these records that are actually taxes are bypassed. Only client types 01 — Mar, 05 — Internal Non/Billable, 06 — Internal Billable, and 08 — BAS are selected. This is determined by a GETBDATA call to the client file.

No computer program could translate that text to an executable program, and even professional programmers would need much more analysis before deciding how to design the system. The problem of comparing previously written programs to the documents that describe them is much easier, but not trivial. Note that the text contains a large amount of jargon, and it mixes English words with the names of programs, files, and variables. Instead of references by name, some files are mentioned by descriptions such as "the 61 byte file that is created by the COBOL program BILLCRUA." The text also uses ad hoc syntax, such as "32 — loss on unbilled."

The project required an analysis of 100 megabytes of English, 1.5 million lines of COBOL programs, and several hundred JCL scripts, which called the programs and specified the data files and formats. Over time, the English terminology, computer formats, and file names had changed. Some of the format changes were caused by new computer systems and business practices, and others were mandated by different versions of federal regulations. The goal was to generate an English glossary of all processes and data, to note and generate cross references for all changes of terminology and definitions over time, to define the specifications for a data dictionary, to create dataflow diagrams

of all processes, and to detect inconsistencies between the documentation and the implementation. Off-the-shelf software was available for analyzing COBOL programs, but not for analyzing the documentation and relating it to the programs. A major consulting firm estimated that the project would require 40 people for two years to read all the documentation, relate it to the software, create all the cross references, and produce the desired results.

For this project, Arun Majumdar and André LeClerc produced those results in 15 person weeks instead of 80 person years. To do that, they used VAE combined with a language analyzer called Intellitex, which translated English to conceptual graphs. The elapsed time was 8 weeks: 4 weeks for design, ontology, and additional programming for I/O formats; 3 weeks to run Intellitex, VAE, and the new programs on all the data; and 1 week to produce a CD-ROM with the results, which were exactly what the company had asked the consulting firm to produce.

During the computation, the combination of VAE and Intellitex analyzed the English documentation in terms of the semantic patterns specified by the COBOL programs and JCL scripts. Key to that analysis was a common knowledge representation in conceptual graphs (Sowa 2008). Even more important were the strategy and tools for using CGs:

- The first step is to use off-the-shelf grammars to analyze COBOL and JCL and add a back-end for generating conceptual graphs instead of executable instructions. That analysis also generates a lexicon of all the names of programs, files, and variables with cross references to the text sources and the CG translations.
- The next step uses VAE to index the CGs from COBOL and JCL to make them accessible while Intellitex is analyzing English. For N graphs, the indexing time is proportional to (N log N), but the time for VAE to find all graphs within a given semantic distance of a particular graph is proportional to (log N).
- Since English sentences are frequently ambiguous, many different CGs can be derived from the same sentence. During the analysis, VAE checks each option against the previously generated CGs to determine which ones are the most likely. Any CGs that match something derived from the programs are saved and indexed. The others are discarded as irrelevant.
- Pronouns and other anaphoric references are resolved by matching the newly generated CGs to other CGs derived from the same document. Names and vague references like “the 61 byte file” can be matched to any CGs derived from the entire corpus. Context surrounding the coreferent nodes can also be used to resolve ambiguities.

When VAE compares a CG derived from the current sentence to CGs derived from the programs or other documents, an exact match confirms the accuracy. If one CG has more or less detail than the others, there is no contradiction. Perhaps some program didn’t need all the detail, or some document didn’t mention it. But sometimes two CGs might

represent different pathways through the background knowledge. Figure 3 shows different paths through the company's database from market to location.

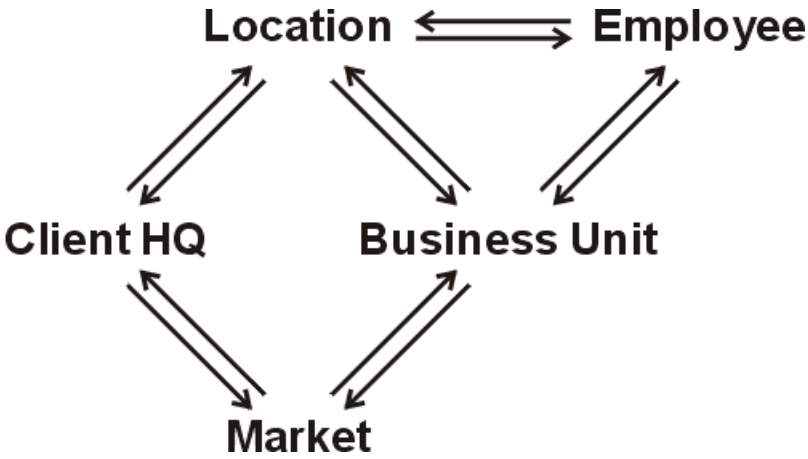


Fig. 3. Different paths for relating market to location

VAE discovered that the CG derived from the documentation showed that the company's market regions are determined by the location of its business units. Sony Pictures, for example, would be in the California market, where the company has a business unit. But the CG derived from the COBOL programs shows that the market region is computed from the location of the client headquarters. Sony Pictures would therefore be in the Japan market. Some programmer had made a mistake, and management was making decisions based on incorrect assumptions. Nobody noticed the discrepancy until VAE discovered it.

As another example, the ontology implied that every employee is a human being and no human being is a computer. But CGs derived from COBOL showed that some employees were computers. The trail of pointers from those CGs led to a comment buried in a COBOL program that described an ad hoc patch. Back in 1979, two computers were used to assist human consultants. The company had standard procedures to bill customers for time spent by their employees, but there was no provision to bill for computer time. Therefore, the programmer named the computers Bob and Sally and assigned employee IDs to them. This was a "temporary" patch that would be removed when the project was finished. But few people clean up after finished projects. As a result, Bob and Sally remained employees for over 20 years before VAE discovered them.

Intellitex has a simple grammar, but it always produces some CG for any sentence. If a word is not in its lexicon, Intellitex capitalizes the word as a starting hypothesis about the associated concept type. If no parse is found for some string of words, it

uses the completely unspecified relation **(Link)** to connect adjacent words. As an example, Intellitex would translate the phrase "32 — loss on unbilled." to a conceptual graph of the following form:

```
[Integer: 32]→(Link)→[Punctuation: "—"]→(Link)→
[Loss]→(On)→[Entity]←(Thme)←[Unbilled]
```

The first line contains a concept of the integer 32 linked to some punctuation linked to a CG for the phrase *loss on unbilled*. The concept **[Entity]** in the second line is derived from a *canonical graph* for the participle *unbilled*, which by default would have some unspecified entity as its theme **(Thme)**. Then VAE would compare this graph to the previously generated graphs to find anything similar. For this example, VAE found comments in the data division of a COBOL program that mentioned "transaction code" and other comments that related 32 to the phrase "loss on unbilled." Similar phrases, such as "72 — loss on uncollected," used the same punctuation for the same semantics. But VAE also found that the syntactically similar phrase "06 — Internal Billable" was related to client types rather than transaction codes. To derive generalizations, detect exceptions, and refine hypotheses, some learning algorithms were later combined with VAE.

More recently, a new VivoMind Language Processor (VLP) has replaced the old Intellitex (Majumdar et al. 2008, 2009). One of the first applications was for analyzing documents about oil and gas fields, and answering extended queries by geologists who wanted to evaluate the potential for exploring new regions. Two geologists visited the VivoMind offices on a Monday morning and brought a collection of 79 documents in the geosciences domain. They ranged in size from 1 to 50 pages, some described sites of interest for oil and gas exploration, and others were chapters from a textbook on geology that VLP could use to extract background knowledge. The documents were not tagged or annotated in any way, except for the usual formatting tags intended for human readability.

The first test was to run the documents through VLP without adding any domain ontology and let the geologists ask questions. The answers were not bad, but they weren't much better than a typical search engine applied to the same documents. As a result of the analysis, VLP also produced a list of all terms that were not found in its lexicon. For the next four days, the geologists worked with the VivoMind staff to generate a domain ontology with lightweight and middleweight semantics. The first task was to classify the unknown words in several categories, such as Rock, RockFormation, Hydrocarbon, and GeologicalAge. Another task was to add domain-dependent word senses for common words, such as *basin*, *cap*, *corridor*, *fan*, *feeder*, *field*, and *reservoir*. The third task was to add a modest amount of background knowledge for resolving some of the ambiguities. The first task, which used lightweight semantics, was completed in about two days. It made a major improvement in the quality of the answers.

The next two tasks used Common Logic Controlled English (CLCE) to state some middleweight semantics. Instead of formal definitions, new word senses for the common words were introduced by stating simple CLCE sentences that use them in the new sense:

```
A cap on a well is a barrier.
A field that contains a hydrocarbon is under
ground.
A reservoir that contains a hydrocarbon is under
ground.
```

The background knowledge was also stated in CLCE:

```
Some ground is under water.
No city is under water.
Every reservoir that contains a hydrocarbon is
in a field that contains a hydrocarbon.
```

The geologists learned to write such statements during their visit, and they continued to add more background knowledge after they left. CLCE is general enough to represent full first-order logic (Sowa 2004), but this level of detail was sufficient for the task. Following is a sample sentence:

> The Diana field is situated in the western Gulf of Mexico
> 260 km (160 mi) south of Galveston
> in approximately 1430 m (4700 ft) of water.

If the sentence had ended with the word *Mexico*, the syntax would be unambiguous. But the measures in the next two lines, the parenthetical expressions, and the points for attaching prepositional phrases create ambiguities. Is Diana field or the Gulf of Mexico south of Galveston? What is in the water? Diana field, the Gulf of Mexico, or Galveston? After a devastating hurricane, Galveston was under water, but background knowledge should imply that cities are usually not under water. Majumdar et al. (2008) describe the VLP organization and how it uses background knowledge to resolve such ambiguities.

After they had added sufficient semantics to the domain ontology, the geologists who developed it invited another geologist from an oil company to test the system. He brought a small file that described a prospective site. He wanted VLP to compare all the sites it had analyzed to the following description, rank the sites that were most similar, and determine both the similarities and the differences for each site:

> Turbiditic sandstones and mudstones deposited as a passive margin lowstand fan in an intraslope basin setting. Hydrocarbons are trapped by a combination of structural and stratigraphic onlap with a large gas cap. Low relief basin consists of two narrow feeder corridors that open into a large low-relief basin approximately 32 km wide and 32 km long.

From the 79 documents it had analyzed, VLP found 17 sites that had some similarity. The most similar was in the Vautreuil region of France. It based that evaluation on three of the 79 documents. The report that described the Vautreuil site was essential, but VLP also extracted information from two chapters of the geology textbook in order to relate the geologist's query to that report. The screen shot in Figure 4 shows how those documents are related to the query.

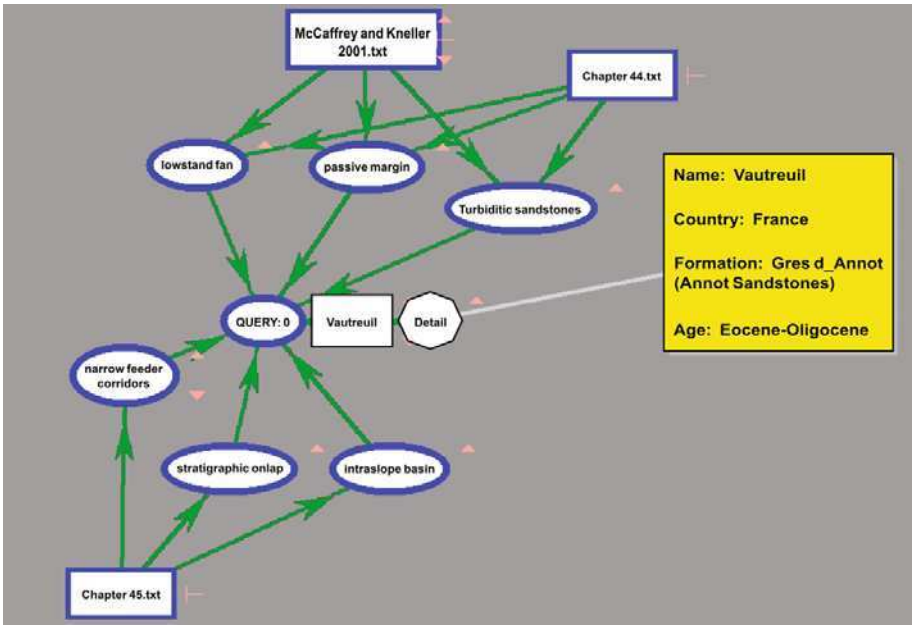


Fig. 4. Relating documents to a query

The oval at the center of Figure 4 represents the query. At the right is a short description of the Vautreuil region. That summary was extracted from a site report about the Vautreuil region written by McCaffrey and Kneller, represented by the box at the top. The six ovals surrounding the query oval contain English phrases, whose translations to conceptual graphs led to the documents used to answer the query. Three of those phrases were not found in the site report, but they led to the box for Chapter 45, from which VLP extracted CGs with background information that it used to interpret the site report. The three phrases above the query oval occurred in both the site report and Chapter 44 of the textbook. The result is a network of conceptual graphs that connect the geologist's query to the Vautreuil report via background graphs derived from chapters 44 and 45. The little red triangles in Figure 4 are links to windows that display relevant paragraphs from the source documents. Clicking on detail leads to a side-by-side comparison of the similarities and differences between the Vautreuil site and the site described by the geologist's query.

The VLP analysis goes into greater depth and precision than current systems for information retrieval (IR) and information extraction (IE). For each of the 17 sites related to the query, VLP found multiple documents that contained CGs derived from the query, CGs from the site report, and CGs from the textbook or other reports that contained background information. In a sense, VLP "learns" new information by reading a book. But for each query, it focuses only on those parts of the book that are useful for relating the query to the answer. This method is very different from current IR, IE, and DB systems:

- IR systems typically use a "bag of words" method to measure the similarity of a query to a document that might contain an answer to that query. But they don't extract the information and summarize it in a table or paragraph. It's possible to apply IR methods to individual paragraphs, but that technique would miss documents in which the significant words are scattered in different paragraphs. And no IR systems connect partial information from multiple documents.
- IE systems extract particular pieces of information, and some can link multiple pieces from different documents. Typical IE systems use predefined templates that specify expected syntactic and semantic patterns, but they have stagnated at about 60% accuracy. Hobbs and Riloff (2010) noted "it is not clear what we can do to overcome [that barrier], short of solving the general natural language problem in a way that exploits the implicit relations among the elements of a text." As Figure 4 shows, VLP doesn't need predefined templates. CGs derived from the query enable it to find implicit relations in a textbook and exploit them to generate precise answers.
- DB systems can relate and combine information from multiple sources, but they use query languages like SQL and SPARQL. Some support English-like front ends, but they face the same customization problems as TQA and Cyc. Furthermore, all the information they access must be predigested and translated to whatever format the database system requires.

Although conceptual graphs are defined as a formal logic, precise logic cannot be derived from a vague sentence. The CG that represents a sentence is actually derived by combining CGs from previously acquired knowledge. The precision of the result is determined by the precision of the original CGs. This method violates Frege's principle of *compositionality*, which says that the meaning of a sentence is derived from the meaning of the words it contains and the grammar rules for combining words. Montague was a strict adherent: each word is defined by one or more logical expressions, and each grammar rule has an associated semantic rule for combining those expressions. Montague allowed some words to have multiple meanings, but the grammar rules check semantic constraints to determine the correct option in each case. To support context-dependent references, Kamp's DRT uses information outside the sentence to determine interconnections. Both neat and scruffy systems make tradeoffs between the amount of meaning stored in the lexicon and the amount derived from context or general background knowledge. The high-speed analogy engine enables VLP to find and use much more background knowledge than most NLP systems.

In summary, VLP uses a combination of lightweight, middleweight, and heavyweight semantics. For any text, the broad outline of meaning comes from lightweight resources such as WordNet combined with middleweight ontologies with few axioms and definitions. The detail comes from background knowledge represented in conceptual graphs. At the heavyweight extreme, those CGs may be derived from formal logics, programming languages, and highly structured databases. The Common Logic standard (ISO/IEC 2007) specifies a model-theoretic semantics for CGs. But CGs have extensions beyond the CL standard (Sowa 2006, 2008), and they can also be used with "scruffy" heuristic methods.

6 Integrating Semantic Systems

Semantic systems have different interfaces for people with different requirements and skills. People with no training in programming or artificial intelligence, either casual users or subject-matter experts, should have interfaces that take advantage of their knowledge of the subject and their knowledge of their native language. Conventional programming tools and AI languages require different kinds of skills, and developers with experience in either or both should be able to collaborate. Automated and semi-automated tools should assist all developers in the stages of design, implementation, testing, and integration with other systems. The examples of Systran, TQA, Tesco, Cyc, and VLP illustrate the issues:

- Systran required highly trained linguists to design the dictionaries, but the resulting translations could be read by large numbers of people with no special training. TQA required DB administrators with special skills to customize the system for every application, even though the number of users of an application might be small. The cost of customization per TQA user was much higher than the cost per Systran user.
- Tesco wanted a more flexible system that could provide helpful suggestions to customers who visited their website. The software vendor designed a reasoning system based on OWL, but Tesco employees could not modify it. Ellis designed a new system with the same kind of interface for Tesco customers, but with a simpler interface for Tesco employees. Ellis's design reduced the cost for updates by Tesco, but many software vendors don't have employees with PhDs in computer science.
- Cyc was developed by 10 person-centuries of programmers and PhDs in several different fields. The cost of customizing Cyc is similar to the cost of customizing TQA, and success depends on the number of users per application. Each client must pay Cyc experts to customize the AI technology. That is a source of revenue for Cycorp, but it limits their market to large clients that can afford to pay.
- The Tesco system used a version of controlled English to enable employees with no AI training to read and update the knowledge base. The language for TQA users was called English, and it had greater expressive power than Tesco English, but it was just as tightly controlled. VLP was designed to process unrestricted natural languages, but it also supports controlled English for SMEs who update, supplement, and correct its knowledge base. All three systems show that users who know the subject matter can adapt to controlled NLs and use them effectively.
- People with every level of skills find diagrams helpful as a supplement to both natural and artificial languages. For the legacy re-engineering project, the VivoMind system translated the internal conceptual graphs to the more conventional dataflow diagrams and system structure diagrams used by programmers and systems analysts. The screen shot in Figure 4 is one of several kinds of diagrams that VLP generates from the internal CGs. They enable a geologist or other SMEs to explore the network of inferences and associations at different levels of detail. At each step, the system can follow

> the pointers from any CG to display the paragraph or paragraphs from which it was derived.

All five systems connect AI technology with conventional software, but the esoteric theories, languages, and methodologies of AI limit their use by most programmers and webmasters. With relational databases, Codd (1970, 1979) introduced first-order logic as the semantic foundation for database query languages. The conceptual schema illustrated in Figure 2 was an attempt to introduce even more semantics into database systems. It stimulated thirty years of research and collaboration between the AI and database communities, but most of that technology remains isolated from mainstream IT. To be successful, AI developers must find ways to simplify their development tools and integrate them with commercial software.

In contrast with the slow transfer of AI research to applications, the original World Wide Web addressed a specific problem with a well-defined goal: combine hypertext with the Internet, and let physicists get any report by clicking on a citation. It worked so well that everybody wanted to use it. The Semantic Web, however, began with the vague idea of adding semantics to the links. AI researchers, who were eager to promote their tools and theories, proposed them to the W3C. Those proposals led to the boxes in the layer cakes of Figure 1. Meanwhile, skeptics like Shirky (2005) claimed that "ontology is overrated." Like the conceptual schema, ontology is a fertile field for research, but most programmers who use XML for data interchange ignore OWL.

Yet the Semantic Web has been developing a valuable set of tools, and they should be better integrated with both AI technology and more conventional software. One way to begin is to fill the box for "unifying logic" in Figure 1 with Common Logic (ISO/IEC 2007). The semantic foundation for Common Logic is based on a proposal by Hayes and Menzel (2001). Guha and Hayes (2002) adopted that semantics for RDF, and it is compatible with every logic in the layer cakes. Common Logic has also been adopted as the unifying logic for the UML diagrams, which are widely used to specify conventional software (OMG 2010). A unifying logic can support development tools with precisely defined mappings between components from different communities with different notations and methodologies.

A promising opportunity for semantic applications is the movement toward Linked Open Data (LOD), and the largest single collection of data is held by the US Government. Vivek Kundra, the chief information officer, summarized the issues:

> Just consider the huge experience gap that Americans have when they go online to make a hotel reservation or buy a book through Amazon versus how they interact with the public sector — whether it's paying taxes, applying for student aid or applying for Social Security benefits. (Quoted by Moyer 2010)

Yet Kendra's examples are far more complex than buying a book or reserving a hotel room. Commercial websites handle those routine transactions very well, just by using conventional programs and databases. For complex reservations, travelers prefer to talk with a human agent, even if they have to pay a service charge. For most commercial sites, the help facilities are notoriously poor, and it's unlikely that they could answer the kinds of questions people would ask about taxes, student aid, or

social security. Search engines are popular because they're easy to use for finding documents, but some knowledgeable person or computer would have to read and understand them to answer such questions.

In the interview, Kendra used the term *data sets*, not *documents*, and most of those data sets are stored in databases. The initial goals are to make the data accessible by interfaces to the web, but those interfaces need not use AI technology. For many of them, tags that mark the datatypes are the only semantics. Such tags are useful for both conventional software and AI technology, but detailed reasoning requires more semantics. Fortunately, AI technology can also derive the semantics. The US Government is the world's largest publisher, and every data set is described or mentioned in many documents. A user-friendly interface should relate that data to the documents and answer questions by extracting related paragraphs. But most LOD projects aren't using NLP methods to process the documents and relate them to the data sets.

The VivoMind examples in Section 5 show how an integrated system of semantic tools can process both structured data and unstructured texts. The legacy re-engineering project shows that formal semantics derived from a subject can be used to interpret reports and manuals about the subject. The method of answering a geologist's query shows how NLP systems can integrate semantics derived from multiple sources to analyze documents and answer questions. These methods are at the cutting edge of applied research, but they are likely to evolve rapidly during the coming decade. That evolution is inevitable, and better tools can facilitate the transition.

References

Berners-Lee, T.: W3 future directions, Keynote speech. In: First International Conference on the World-Wide Web, May 25-27. CERN, Geneva (1994), `http://www.w3.org/Talks/WWW94Tim/`

Bray, T.: The RDF.net challenge (2003), `http://www.tbray.org/ongoing/When/200x/2003/05/21/RDFNet`

Chalupsky, H., MacGregor, R.M., Russ, T.: PowerLoom Manual. ISI, Marina Del Rey, CA (2006)

Codd, E.F.: A relational model of data for large shared data banks. Comm. ACM 13(6), 377–387 (1970)

Codd, E.F.: Extending the relational model to capture more meaning. ACM Transactions on Database Systems 4(4), 397–434 (1979)

Compton, P., Peters, L., Edwards, G., Lavers, T.G.: Experience with ripple-down rules. Knowledge Based Systems 19(5), 356–362 (2006)

Damerau, F.J.: Operating statistics for the Transformational Question Answering System. American Journal of Computational Linguistics 7(1), 30–42 (1981)

Damerau, F.J.: Prospects for knowledge-based customization of natural language query systems. Information Processing and Management 24(6), 651–664 (1988)

Ellis, G., Levinson, R.A., Robinson, P.J.: Managing complex objects in Peirce. International J. of Human-Computer Studies 41, 109–148 (1994)

Feigenbaum, E.A., McCorduck, P.: The Fifth Generation. Addison-Wesley, Reading (1983)

Guha, R.V.: Contexts: A Formalization and Some Applications, PhD dissertation, Stanford, and Technical Report ACT-CYC-423-91, MCC, Austin, TX (1991)

Guha, R.V., Bray, T.: Meta Content Framework using XML, W3C (1997), http://www.w3.org/TR/NOTE-MCF-XML-970624

Guha, R.V., Hayes, P.J.: LBase: Semantics for languages of the semantic web (2003), http://www.w3.org/TR/2003/NOTE-lbase-20031010/

Hayes, P.J., Menzel, C.: A semantics for the Knowledge Interchange Format. In: Workshop on the IEEE Standard Upper Ontology, IJCAI 2001 (2001), http://reliant.teknowledge.com/IJCAI01/HayesMenzel-SKIF-IJCAI2001.pdf

Hayes, P. (ed.): RDF Semantics. W3C Recommendation (2004), http://www.w3.org/TR/rdf-mt/

Hobbs, J.R., Riloff, E.: Information extraction. In: Indurkhya, N., Damerau, F.J. (eds.) Handbook of Natural Language Processing, 2nd edn. CRC Press, Boca Raton (2010)

Hutchins, W.J.: Machine translation: a brief history. In: Koerner, E.F.K., Asher, R.E. (eds.) Concise History of the Language Sciences: from the Sumerians to the Cognitivist, pp. 431–445. Pergamon Press, Oxford (1995)

ISO/IEC, Common Logic (CL) — A Framework for a family of Logic-Based Languages, IS 24707, International Organisation for Standardisation, Geneva (2007)

Kamp, H., Reyle, U.: From Discourse to Logic. Kluwer, Dordrecht (1993)

Kamp, H.: Levels of linguistic meaning and the logic of natural language (2001), http://www.illc.uva.nl/lia/farewell_kamp.html

Kassoff, M., Genesereth, M.R.: PrediCalc: a logical spreadsheet management system. Knowledge Engineering Review 22(3), 281–295 (2007)

Moyer, M.: Digitizer in chief, an interview with Vivek Kundra. Scientific American 303(4), 90–94 (2010)

LeClerc, A., Majumdar, A.: Legacy revaluation and the making of LegacyWorks. Distributed Enterprise Architecture 5(9) (2002)

Lenat, D.B., Feigenbaum, E.A.: On the thresholds of knowledge. In: Proc. IJCAI 1987, pp. 1173–1182 (1987)

Lenat, D.B., Guha, R.V.: Building Large Knowledge-Based Systems. Addison-Wesley, Reading (1990)

Majumdar, A.K., Sowa, J.F., Stewart, J.: Pursuing the goal of language understanding. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS (LNAI), vol. 5113, pp. 21–42. Springer, Heidelberg (2008), http://www.jfsowa.com/pubs/pursuing.pdf

Majumdar, A.K., Sowa, J.F.: Two paradigms are better than one and multiple paradigms are even better. In: Rudolph, S., Dau, F., Kuznetsov, S.O. (eds.) (2009), http://www.jfsowa.com/pubs/pursuing.pdf

MacGregor, R.A.: The evolving technology of classification-based knowledge representation systems. In: Sowa, J.F. (ed.) Principles of Semantic Networks, pp. 385–400. Morgan Kauffmann, San Mateo (1991)

McGuinness, D.L., Fikes, R., Hendler, J., Stein, L.A.: DAML+OIL: An ontology language for the Semantic Web. IEEE Intelligent Systems 17(5) (2002)

Montague, R.: Universal grammar (1970); reprinted in Montague, R.: Formal Philosophy, pp. 222–246. Yale University Press, New Haven

Mueckstein, E.-M.M.: Q-Trans: Query translation into English. In: Proc. IJCAI 1983, pp. 660–662 (1983)

OMG, Semantics of a Foundational Subset for Executable UML Models, Object Management Group (2010), http://www.omg.org/spec/FUML/1.0

Peterson, B.J., Andersen, W.A., Engel, J.: Knowledge bus: generating application-focused databases from large ontologies. In: Proc. 5th KRDB Workshop, Seattle, WA (1998), http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-10/

Petrick, S.R.: A recognition procedure for transformational grammars, PhD dissertation. MIT, Cambridge (1965)

Petrick, S.R.: Field testing the TQA System. In: Proc. 19th Annual Meeting of the ACL, pp. 35–36 (1981)

Quinlan, J.R. (ed.): C4.5: Programs for Machine Learning. Morgan-Kaufmann, San Mateo (1993)

Sarraf, Q., Ellis, G.: Business rules in retail: the Tesco.com story. Business Rules Journal 7(6) (2006)

Shirky, C.: Ontology is overrated. Talk presented at the O'Reilly Emerging Technology Conference, San Diego (2005), http://www.shirky.com/writings/ontology_overrated.html

Sowa, J.F.: Graphics and languages for the flexible modular framework. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 31–51. Springer, Heidelberg (2004)

Sowa, J.F.: Worlds, models, and descriptions. Studia Logica, Special Issue Ways of Worlds II 84(2), 323–360 (2006)

Sowa, J.F.: Conceptual graphs. In: van Harmelen, F., Lifschitz, V., Porter, B. (eds.) Handbook of Knowledge Representation, pp. 213–237. Elsevier, Amsterdam (2008)

Sowa, J.F., Majumdar, A.K.: Analogical reasoning. In: de Moor, A., Lex, W., Ganter, B. (eds.) ICCS 2003. LNCS (LNAI), vol. 2746, pp. 16–36. Springer, Heidelberg (2003), http://www.jfsowa.com/pubs/analog.htm

Tsichritzis, D.C., Klug, A.(eds.): The ANSI/X3/SPARC DBMS framework. Information Systems 3, 173–191 (1978)

Wang, H.: Toward mechanical mathematics. IBM Journal of Research and Development 4, 2–22 (1960)

Whitehead, A.N., Russell, B.: Principia Mathematica, 2nd edn. Cambridge University Press, Cambridge (1910/1925)

Chapter 3
Defining and Validating Semantic Machine to Machine Interoperability

Claudia Szabo[1] and Saikou Y. Diallo[2]

[1] Department of Computer Science National University of Singapore Computing 1, 13 Computing Drive Singapore 117417
Claudias@comp.nus.edu.sg
[2] Virginia Modeling Analysis and Simulation Center, 1030 University Blvd Suffolk, VA, 23435
Sdiallo@odu.edu

Abstract. Current approaches to interoperability focus on the technical aspects related to connecting systems through the development of technical standards and frameworks and the semantic aspects of exchanging data through the development of common reference models (CRM), ontology or federated schemas. This chapter will show that those approaches are computationally equivalent and will formally define what pure machine to machine semantic interoperability is as opposed to human in the loop interoperability where a human is needed to assess semantic equivalence. A formal method for validating federations under certain conditions is also provided. The validation process looks at general model properties of the composed artifact, such as ensuring that the federations execute correctly without hanging. Next, the aggregated execution of the federation is formally compared with the execution of a reference federation. Comparisons consider time-based orderings and semantic closeness according to an ontology that describes federations and exchanged data.

Keywords: Modeling, Simulation, Interoperability, Composability, Validation.

1 Introduction

The term interoperability[1] is used to mean different but closely related things depending on the application area. Nations and companies are ready and willing to invest in "interoperable solutions" even though it is not clear what that really means or what an interoperable solution represents. It is however clear that just like composition, and validation[2], interoperability is more a practice than a science. As a result it is difficult to come to a consensus on what the problems inherent to interoperability are, how they are categorized and whether they can be solved. Issues such as multi resolution, multi scope and multi structure that are not central in software interoperation take on a new dimension in model based interoperation because each system is a purposeful abstraction of reality [1]. Consequently solutions

[1] The work on the formal aspects of interoperability is drawn from [38].

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 49–74.
springerlink.com

dealing with technical issues such as connectivity or syntactical alignment while sufficient for software and hardware interoperability fall short when it comes to model based systems [2].

One of the main roadblocks to interoperability for model based system is the issue of semantic inaccessibility. As stated in [3]:

"*The semantic rules of the component simulation tools and the semantic intentions of the component designers are not advertised or in any way accessible to other components in the federation. This makes it difficult, even impossible, for a given simulation tool to determine the semantic content of the other tools and databases in the federation, termed the problem of semantic inaccessibility. This problem manifests itself superficially in the forms of unresolved ambiguity and unidentified redundancy. But, these are just symptoms; the real problem is how to determine the presence of ambiguity, redundancy, and their type in the first place. That is, more generally, how is it possible to access the semantics of simulation data across different contexts? How is it possible to fix their semantics objectively in a way that permits the accurate interpretation by agents outside the immediate context of this data? Without this ability—semantic information flow and interoperability—an integrated simulation is impossible"*.

The challenge is therefore to make data in systems semantically accessible to other systems so that they make use of it. The idea that systems can access the semantics of data produced by other systems without a human in the loop making decisions about semantic equivalence is interesting but is it possible? In this chapter, we will attempt to answer this question by presenting what semantic interoperability means for a machine and how to validate a federation of interoperating machines. The first part of the chapter will focus on defining interoperability and examining how it is currently understood. We will also formally define what interoperability is for machines at the data level. The second part will focus on validating a federation of interoperating machines and ascertaining that the federation is behaving in accordance with the desired expectations. We will conclude with a series of observations and recommendations for agent supported machine to machine interoperability at the semantic level.

2 State of the Art in Interoperability

In this section, we examine definitions of interoperability in order to derive a shared understanding of the basic expectations when we use the term "interoperability". Webster's online dictionary defines interoperability as

"the ability to exchange and use information (usually in a large heterogeneous network made up of several local area networks)" [4]

The Department of Defense (DoD) defines interoperability as

"the condition achieved among communications-electronics systems or items of communications-electronics equipment when information or services can be exchanged directly and satisfactorily between them and/or their users. The degree of interoperability should be defined when referring to specific cases."[5]

The Institute of Electrical and Electronics Engineers (IEEE) defines interoperability as

"the ability of two or more systems or components to exchange information and to use the information that has been exchanged"[6].

Another definition of interoperability is proposed by the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) as

"The capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units."[7]

While these definitions are not formal, they emphasize two points that are recurrent in all of the working definitions of interoperability found in the literature:

Information Exchange: Interoperable systems are characterized by their ability to exchange information. It is also clear from these definitions that interoperability is at the system level and more precisely systems that are implemented on a computer. This definition also takes the position that interoperability is a condition that must be achieved which implies that systems are interoperable when they are interoperable. The IEEE definition defines interoperability as inherent to a system (its ability to exchange information) which implies that systems are interoperable if they are interoperable. These two definitions are both tautological and contradict one another which show that the understanding of what interoperability is at the basic level is not consensual. The ISO definition focuses solely on the technical side of interoperability and ignores the semantics and pragmatic aspects of data i.e. its usefulness. However, it is still an applicable definition because it talks about data instead of information.

Usability of Information: The other aspect stressed in these definitions is the notion of usability or usefulness of the information exchanged. The natural question that arises immediately is who determines what is useful and is this determination done before, during or after the information exchange. In the case of the IEEE, the use of information is determined by the receiving system as the wording indicates, which implies that not only the receiving system is able to process information it can also determine which information it can use and which it should throw out. It also points to the fact that there is a direction of information flow and it is important to identify it during interoperation. The difficulty for machine to machine interoperability is in this second part because machines do not have the ability to ascertain meaning as humans do and typically treat data as bits and bytes. Meaning has to be imposed from outside the machine through the use of models (in this case a protocol or standard is considered a model).

The distinction between data and information is important because all information is data but the converse is not true. According to [8], data are simply a set of symbols, facts or figures, while information is data that are processed to be useful and provides answers to questions such as "who", "what" etc…. The differentiation between data and information is essential because typically computers deal with data while humans deal with higher levels of abstraction such as information, knowledge and wisdom [9]. Nonetheless, it is possible to represent information in a computer by relating data in a structure. The term interoperability as described in the earlier definitions point to not only information but useful information. In Ackoff's categorization, useful information is knowledge and gaining understanding through knowledge is the ultimate goal of interoperability. However, there must be a transition between data, information and knowledge that allows a formulation of a useful formal definition of interoperability.

The next important question is the determination of what is useful. In the current state of the art, the determination of what is useful is completely dependent on the system receiving the data and the sending system is not able to determine whether what is exchanged is useful. The current approach is to introduce an interface that will ensure usefulness by codifying all relevant transactions and translations between machines. The interface qualifies as a model whose role is to broker information between models. If we want a machine to determine what is useful such a model is required and the machine must be able to generate it. However, the introduction of an interface does not answer the question as to what is useful and actually leads to a paradox if we were to use a machine (read software agent) to generate the interface:

Paradox 1: *Let's consider two models A and B which are interoperable through an interface I. If I is a model then A and I are also interoperable and therefore an interface* I_1 *is required between A and I.* I_1 *is also a model and therefore and interface* I_2 *is required. To generalize, interface* I_n *and A are interoperable requires and interface* I_{n+1} *which leads to an infinite sequence.*

The same logic can be applied to *I* and *B*. Consequently, the introduction of an interface is an outcome of interoperability not interoperability itself. For machines, this paradox resolves itself at the bit and bytes level as there is usually an agreement on hardware protocols to directly connect machines. However, the problem remains intact in that we still do not know what interoperability means for a machine let alone what semantic interoperability means. From the definitions above, we do know that interoperability involves the exchange of data and the use of data once it is exchanged. Consequently, it is safe to conclude that understanding what the semantics of data are for a machine is a good starting point for understating what semantic interoperability means for a machine.

The next section examines data models and Data Modeling Theory (DMT) in order to explain the semantics of data. DMT has focused on defining data models and conveying meaning between machines over the past forty years. Current approaches to data modeling and databases such as the Entity-Relationship (ER) model [10] and the Relational Model (RM) [11] are some of two of the well known models of DMT. In the case of machine to machine interoperability, each machine has an internal representation of data, and in this part of the chapter, we focus on data solely without

any direct concerns for the functions that produce or consume data. These functions are examined when we discuss federation validation as they capture part of the behavior of the federation.

2.1 Semantics of Data for a Machine

A data model in Data Modeling Theory is usually a means to store data that is relevant to an activity. Typically, this model is caputred as a database which is different from a model that is developped to answer a modeling question. In this chapter, we are interested in models in the Modeling and Simulation (M&S) sense of the term i.e. a purposeful abstraction of a referent (real or imagined things). Consequently, semanctic machine to machine interoperability is similat to M&S interoperability and the problem we are addressing can be reduced to identfying how agents can help enable semantic machine to machine interoperability. The model in DMT is an abstraction of reality but not a purposeful one in the M&S sense. The model of data represented in machine must be a purposeful one because it captures the meaning intended by the designer of the model. In [12] the author argued the need to differentiate between the real world and the model of the real world and showed that if a question that is relevant to the real world is posed to a model of this world that did not consider the question in the first place, it leads to the problem of incomplete information. This problem simply does not exist in M&S because of the purposeful nature of the model. Further, different modelers might model the same thing differently, one as an object the other as an attibute. Objects, attributes and value domains are possible representations of a referent. The selection of objects, attibutes and value domains depends on the purpose of the model or the modeling question one is trying to answer.

DMT has focused on capturing and communicating more semantics through two main forms. The first is the introduction of hierarchical structures to represent how things are related and the second is the reliance on terms to carry the meaning of what is represented. While the reliance on structure is adequate for machines as they can easily represent and capture structures, the reliance on terms to carry meaning presents a central problem. Aside from the issue that machines cannot understand the meaning of terms, the problem is that a term is a model because just like any model it is an abstraction of reality. This observation leads to another paradox that is stated as follows:

Paradox 2*: Given a term that has some meaning, it takes a least one term to describe it. The describing term needs at least one term describe it, etc…, which leads to an infinite recursion.*

This paradox is similar to the "symbol-grounding problem" found in Artificial Intelligence [13]. In order to avoid this paradox, a starting point must be provided in the form of either an initial set of terms that all agree upon or a description of terms that all can refer to. This description of terms can be in a dictionary, a taxonomy or an ontology. The problem in agreeing to an initital set of terms is that each machine can have an internal representation of data using its own set of terms. Therefore, Paradox 2 is avoided not by agreing on a common set of terms to be used universally but by

establishing equivalence between sets of terms. The first condition for establishing equivalence is to guarantee that each term corresponds to a unique string within a model and that the term has one and only one meaning (imposed from the outside). This notion is formally captured in the next section as a domain.

The structure of models as reviewed in this section are assumed to be unique for a given referent. For machine to machine interoperability, there exist multile possibly equivalent structures. Further the structure addresses how things are related and not how they change with respect to one another. DMT needs the strucure for the purpose of data integrity meaning data that belongs together is always provided as whole or not at all. However, in the current understanding of interoperability, the idea of integrity is not explicitely supported due to the independence assumption which is described as follows:

Independence assumption*: For a given model, each element within the model exists independently from any other element.*

The independence assumption is the driving force behind the definition of interoperability and the inteoprability approaches reviewed in the previous section. However, this assumption leads to multiple versions of the truth in a federation of models, that is to say that if two identical elements participate in different structures they might change differently for the same input. As a simple example, let's consider model one where a tank has a crew (tank and crew are objects that are related) and model two where a crew is part of a tank (crew is an attribute of the tank). If the models interoperate over tank or exchange information about the tank, the destruction of a tank results in the destruction of the crew in model two while the crew might still be alive in model one if they were not in the tank at the time it was destroyed. In this federation, it is possible for the crew to be both dead and alive at the same time. Consequently in defining machine to machine interoperability, it is essential to avoid the independence assumption and treat it as a special case.

This section motivates the need to separate the structure of models from the meaming of terms. The structure of models in Data Modeling Theory is applicable to M&S and brings the additional element of data integrity to avoid the independence assumption at work in the approaches to interoperability. The reliance on terms to carry meaning leads to an infinite recursion which can be avoided by creating a domain of validity for the terms used in a model in the form of a set and allowing equivalence relations to be established between sets. The section shows that the definition of a data model in M&S should take into account the modeling question and allow all possible representations of the referent to be captured. The separation between the referent and the model of the referent are shown to be essential in interoperability. The next section introduces a formal specification of data in machines based on the discussion presented in this section. This formal specification takes into account the aspects of data from Data Modeling Theory and conceptual modeling as well as additional aspects essential to M&S.

2.2 Formal Representation of Data for a Machine

A formal definition of a model has to take into account not only the model but also its relation to the referent on one hand, and its relation to the simulation on the other hand. In order for a machine to account for the semantics of the model, it is important to decouple the description of the model from its representation and its representation from its implementation. The three terms-entities, properties, value domains-constitute the core of a data model in Data Modeling Theory. For agents, this chapter adds the notion of a domain which is similar to the traditional view of the value domain (a collection of values that an attribute can take) but is generalized to encompass the domain of discourse. Finally, the term element is used to mean anything real or imagined that can be described and/or observed. The five terms are defined as follows:

Elements are real or imaginary things.
Entities are abstractions of elements. It is worth noting that by this definition any abstraction including processes and relationships are considered entities.
Properties are the characteristics of an element.
Symbols are the representations of elements. Symbols can be numbers, strings, images, text or a combination of symbols.
A *domain* is a collection of unique symbols. The domain is the set of elements that belong to a context. Every element is uniquely identifiable within a given domain.

It is important to note that elements are used to separate the notions of entities and properties and how they are related. Elements can be modeled as entities or properties and then related to form the entity-property relationship. This separation of elements and how they are represented reflects the general case and therefore subsumes the entity-property-value triple introduced by the ER and RM.

Given these definitions, let us first formally capture a conceptualization of the referent:

Definition 1. *Let S be the set of elements,* Ω *the set of entities,* Π *the set of properties, V the set of symbols. A conceptualization S is formally captured by the categorization of elements into entities, properties or symbols. Formally a conceptualization S is a partial function F defined such that:*

$$F(S) = \begin{cases} \Omega \; if \; S \; is \; an \; entity \\ \Pi \; if \; S \; is \; a \; property \\ V \; if \; S \; is \; a \; symbol \end{cases}$$

A function is a binary relation between sets in which every ordered pair has a different first member. A partial function is a function for which not every member participates in the relation [14]. The definition of a conceptualization as a partial function accounts for the complexity of elements and the fact that it is impossible to capture them completely for non-trivial cases. Staying with functions, a bijection, surjection and injection are defined as follows [14]:

Injection: For every element in S, $x,\ x = F(S)$.
Bijection: A function is bijective if it is injective and surjective.

There is no requirement that reality be injective (distinct elements of S map to entities, properties or symbols) or surjective (every entity, property and symbol in their respective set must refer to an element) during the conceptualization process. However, once a commitment is made, a representation (capture) of *F(S)* must be bijective and consequently always have an inverse. An inverse is a function G such that for an element s and an entity, property, or symbol $x,\ G(x) = s\ if\ F(s) = x$.

Definition 2 captures the need to express the conceptualization process as a bijective function.

Definition 2. *An element within S is an entity, a property or a symbol, otherwise stated the three sets are mutually disjoint.*

The introduction of the domain as a collection of symbol provides one way to avoid paradox 2. To illustrate, let's consider the following problem absent the notion of a domain:

Given two conceptualizations, is it possible to determine whether they are equal?

This question is central in interoperability as it determines whether the information exchange is meaningful. Since conceptualizations are partial functions and two functions F and G are said to be equal if $Fs{=}Gs$ for every element s, this question can be reduced to a problem known as the *Equivalence Problem* and has been shown to be unsolvable in the general case (Hein, 2002). For machine to machine interoperability it means that, in general, given two representations we cannot determine whether they refer to the same referent. This finding is not surprising considering that even for humans this is a difficult endeavor. Consequently, semantic interoperability for machines is limited to comparing sets of terms and structures without regard to what it is they represent in the real (or imaginary) world.

Having established that fact, let's now consider the domain and the following definition:

Definition 3. *Given a set S of elements and a non-empty set of domains Δ, every element in S is associated with a domain.*

Mathematically, we define the tuples:

- *α is a subset of* Ω X*Δ, the Cartesian product of entities and domains*
- *β is a subset of* Π X Δ*, the Cartesian product of properties and domains*
- *γ is a subset of* V X Δ*, the Cartesian product of symbols and domains*

For every element s belonging to S, s belongs to α, β or γ. In addition α, β and γ are disjoint as a consequence of definition 2. The domain reduces terms to their assigned symbol. Terms have the meaning assigned to it its domain which might or might not be the same as other meanings it has in other domains. Assigning meaning to a term is a modeling decision captured by definition 3. Determining equality between conceptualizations is reduced to determining equality between domains by definition 3.

A direct consequence of the introduction of the domain is the shift in the role of a referent which is now undistinguishable from the conceptualization at least in terms of its description. The distinction while existing in reality disappears once conceptualizations are captured.

Having captured a model description, we need to capture the structure of a model as a means to carry semantics. A model of a conceptualization is one of the many possibly equivalent representations of the conceptualization which itself is one of the many possibly equivalent conceptualizations of a referent. Conceptualizations might or might not be related through a relationship relation. The following definition is a generalization of all possible relationships including relationships between relationships that are expressible within this formalism:

Definition 4. *Given Δ the set of domains, we define the relation ρ as the subset of Δ X Δ the Cartesian product of domains.*

The relation ρ captures the relationship between entities and entities, entities and properties, entities and symbols, properties, properties and symbols and symbols and symbols. In addition ρ captures relationships between relationships if one considers that all the relationships in definition 1.3 are elements that have as domain a subset of Δ X Δ and therefore abide by the previous definitions. The relation ρ is a graph with vertices Δ.

Having defined the conceptualization, let us now define a model of the conceptualization:

Definition 5. *A model M of a conceptualization S denoted M_S is the relation $(\alpha, \beta, \gamma, \rho)$.*

By definition 5 a model is also a representation of a conceptualization. If M is countable, M is computable and further if M is finite and countable it can implemented on a digital computer. However, results derived from these definitions are not limited to computable functions but should apply in general. The model avoids the paradoxes by separating the referent, a conceptualization of the referent and a model of the conceptualization. The semantics are captured by a collection of groupings of symbols and how they are related. Several additional observations should be pointed out with these definitions. The first observation is that the referent and a model of the referent are not required to be finite and/or countable and therefore are not required to be computable. The second observation is that the definitions do not make any assertions about inherent semantic relationship between the sets. In this sense any model of the referent is captured under these definitions.

Definition 1 does not take a position on which set to capture first and in fact does not require any set to be non-empty except for the domain. It is perfectly acceptable to view the world in terms of properties and symbols or entities and domains. Consequently, the definition does not espouse any predetermined description of the referent. Most importantly, the semantics of the referent and the model are explicitly captured by the quadruple. The introduction of the domain as part of the specification in definition 3 plays the same role as the use of labels to carry meaning but in a

formal way. Definition 4 accounts for the semantic relationships between entities. The existence of a relationship between entities implies a relationship between the domains of these entities. Definition 4 implies that a new domain is created by relating entities and the domain thus created is the context of the relationship. Definition 5 is a generalization of the traditional view of a model in which entities have properties which have values and those values can in turn be grouped into a value domain. The traditional view does not cover a model in which entities interact and are affected by their environment. In this case, entities might or might not have properties explicitly modeled; nonetheless there are properties of the environment that are affecting their interactions and properties of their behavior (the exchange of Protocol Data Unit between entities is a simple example). This is the case for example in multi-agent models. Another simple example is a grouping of properties of several entities to form a context or the occurrence of events in event-based simulations or the modeling of structures in System Dynamics. Definition 5 covers all these models within its specification and additionally allows relationships to be related within its specification.

Having defined a model and shown that interoperability for machines reduces to comparing domains and relationships between domains in this section, the section will define and discuss machine to machine interoperability.

2.3 Semantic Machine to Machine Interoperability

In general, while there are guidelines and best practices for modeling, a normalization process as practiced in Data Modeling Theory might result in a model that does not fit the scope and resolution originally intended. At this juncture, it is more important for the model to be separated from the business rules that dictate the interactions with it. This separation is also very useful in augmenting the ability of models to interoperate as it distinguishes between the semantics of the data model and the semantics of the interactions with the data model. As a reminder from the previous section, a model of a referent M_S is the relation (Ω, Π, V, ρ) which is a composition of the subset of the Cartesian product of entities and domains, properties and domains, symbols and domains, and domains with themselves. We distinguish between existential dependency and transformational dependency.

Definition 6. *Let X, Y be sets of entities, properties or symbols with respective domains Δ_X and Δ_Y, Y is existential dependant on X denoted $X\Phi Y$ or $\Phi(X, Y)$ if the existence of Y implies the existence of X.*

Every element is existentially dependent on itself and it is worth noting that the set thus defined is a subset of ρ the Cartesian product of domains which is nonempty implying that Φ is also nonempty. By this definition, multi valued dependency is the particular case where Φ is a function and X, Y are sets of properties where Y has cardinality one. Existential dependency is a generalization of traditional conceptual modeling relationships (is-a, part-of, has-a, etc…) that is able to capture the idea that a designated grouping of elements (entities, properties, symbols) has some meaning

[15]. Traditionally the meaning of these groupings is carried by a semantic label assigned to Φ (is-a, parent-of, child-of), but in this case the meaning is carried by the grouping of the two domains. In practical terms, semantic labels are meaningful to human consumers, but for computers it translates into an association between elements in one namespace with others in the same or a different namespace. The label is then another term used to capture existential dependency. As a simple example, let's take the statement "*son-of (parents, child)*" to mean "a child is the son of its parents". It can be easily verified that the existence of a child depends on the existence of its parents. This statement allows the identification of the entities that are members of son-of but it falls short of capturing all the characteristics of the relation i.e. what does it mean to be the son-of an entity and how to automatically identify those entities. Let's assume that this is an inheritance relationship as the label suggests, then *son-of* means that a child has at least all of the properties of the parent in addition to its own. This is another existential dependency but this time between the properties of the entities. We write son-of $\rightarrow(\Pi_Y \subseteq \Pi_X)$ to capture this relationship. Let us now assume, that son-of also means that a son has to have certain properties assigned a constant value. The easiest example is to require that all sons be male. We write son-of $\rightarrow$ (Π_Y = Sex$\rightarrow$Sex≡Male) to capture that relationship. It is worth noting that because this is an existential dependency between symbols, we use equivalent instead of equal to capture the requirement that a function translating the symbol of Y into "Male" must exist for the property "Sex". This example illustrates the need to express the meaning of a label in terms of dependencies between entities, properties and symbols. It also shows that dependencies exists between domains for humans but must be expressed between terms to have meaning for machines.

Existential dependencies capture the fact that a set of elements (entities, properties, symbols) cannot exist without another. Transformational dependencies exist when in the process of interacting with the model; an update to an element (entity, property, symbol) implies an update to another which often will be the case when users or other systems are interacting with the system.

Definition 7. *Let X, Y be sets of entities, properties or symbols with respective domains Δ_X and Δ_Y, Y is transformational dependant on X denoted XΘY or Θ(X, Y) if a change to Y implies a change to X.*

It is trivial to show that every element is transformational dependent on itself and Θ is a subset of ρ which means that Θ is nonempty. Change could be the creation, deletion or update of an element similar to the specification presented in [16]. The nature of the change can be captured similarly to existential dependency. As a simple example, a symbol y of Y is transformational dependant on a symbol x of X thusly
$\Theta(X, Y) \rightarrow y = x + 3$.

Similar to existential dependency, a transformational dependency between domains can translate into dependencies at the entity, property and symbol level. Continuing with the previous example, in *son-of*, child is not transformational dependant (transformational independent) of parents because a change in parents does not imply a change in child. It is important to note that contrary to intuition, existential dependency does not imply transformational dependency.

Interoperability and current approaches to interoperability have been reviewed in the previous section. While the focus is on interoperability, it is obvious that interoperation is subsumed within that concept. Interoperation is perceived to be necessary but not sufficient for interoperability. The definitions and models of interoperability as well as industry standards for interoperability are evidence of this common understanding with an accent on semantic interoperability. The review also shows that semantic interoperability can be enhanced by agreeing on the meaning of labels and models either through standardization or a CRM. Interactions with the model are subsumed within this agreement whatever form it takes. Based on the formalism developed in this chapter, it has been shown that instead of interoperability, current approaches address interoperation. Without using any particular definition, the review done in section two shows that key characteristics of interoperability are the exchange of information and the use of the information thus exchanged. The exchange of information is interoperation and the evaluation of its usefulness determines the degree of interoperability between the systems. Using the formalism defined previously we can formally exami*ne interoperability* and *interoperation.*

Interoperation informally captures information exchange between systems. Interoperation is formally defined along with its characteristics and requirements as follows:

Definition 8. (Interoperation): *Let M_S be an arbitrary model of a referent S, Φ the set of existential dependencies within M and Θ the set of transformational dependencies within M, a model A is said to interoperate with M if there is a subset of Φ in A or A and M interoperate, denoted A Φ M if $\Phi(A) \cap \Phi(M) \neq \emptyset$.*

A and M are said to interoperate over the subset of Φ which represents the intersection of the sets of existence dependencies between the models. The subset of Φ over which A and M interoperate is the set of elements that A can produce and M can process. By definition this subset is the Common Reference Model (CRM) of A and M. The degree of interoperation between A and M is the cardinality of the CRM.

Definition 9. (Existence condition)*: Given two models A and M, A and M interoperate implies the existence of a CRM.*

Interoperability informally captures the notion of the use of information once it is exchanged. The fact that A and M interoperate does not mean does that they are interoperable. Interoperability requires the ability of M to use what it receives from A or conversely the ability of A to interact with M following the rules of interaction of M. To illustrate, the following proposition is stated:

Definition 10. (Interoperability): *Let M_S be an arbitrary model of a referent S, Φ the set of existential dependencies within M and Θ the set of transformational dependencies within M,* a model *A and M are interoperable denoted $A\Theta M$, if A and M can interoperate and* $\Theta(A) \cap \Theta(M) \neq \emptyset$.

Having defined interoperation (the exchange of information) and interoperability (the exchange and use of information), we can now use Graph Theory to study the

implications of these definitions for machine to machine interoperability. Graph theory is an area of mathematics focused on the study of connections between pairs of objects or collections of objects [14]. This area of mathematics is relevant to a theory of interoperability because as discussed in the previous section, a model is a composition of relations between elements which results in a structure. The representation of a model as graph is equivalent to its representation as a relation. The representation of models as graphs allows us to inherit all the findings of Graph Theory and apply them to interoperation and interoperability. The determination of a CRM is similar to finding the similarity between two or more graphs. We will examine the complexity of finding a CRM and formally show that current approaches to interoperability are equivalent. This finding motivates the need for heuristics that enables interoperation and interoperability.

There are different types of graphs but in this chapter the term *graph* is used to mean *multigraph*. A *multigraph* is formally defined as the triple $G = (V, E, \mathrm{A})$ where

- V is the finite set of vertices or nodes
- E is the finite set of edges
- $A: E \rightarrow E$, the identity map

From the definitions of a model provided earlier, many graphs can be specified. However, the most general case is to define a finite set of elements as the set of vertices V, $E = (\alpha \cup \beta \cup \gamma \cup \rho)$ the union of all possible relationships between elements and $A: E \rightarrow E$ the function that relates entities and their relationships.

A model is simply the graph $G = (V, E, \mathrm{A})$ as formulated above. The formulation of G subsumes the existence condition; that is to say that a formulation of a model as a graph contains the formulation of all of its existential dependencies.

Two graphs G and H *isomorphic* if there is an edge preserving morphism between G and H. Formally, two graphs are isomorphic if there exist a function:

$$f: G \rightarrow H \text{ and } f \text{ is bijective.}$$

The definition of isomorphism leads to the following definition:

Definition 11. *Two models G and H interoperate if and only if they are isomorphic.*

This definition is consistent with previous conditions of interoperation as the existence of an isomorphic between the two models is equivalent to stating that they intersect. Interoperation between *N* models is defined as follows:

Definition 12. *A collection of models $G_1, G_2, \ldots, G_n$ interoperate if and only if they are isomorphic.*

Definition 4.2 is also consistent with the notion that interoperation is bounded. Simply stated, the degree of interoperation is the cardinality of the isomorphic class which is finite and bounded by the smallest and largest isomorphic set. Previously in this chapter, interoperability has been defined as the intersection of the transformational dependencies between models. In term of a graph, if the set V of vertices is defined as the set of elements and the set E of edges as the set of transformation dependencies, definition 4.2 can be reformulated as follows:

Definition 13. *A collection of models G_1, G_2,…,G_n are interoperable if and only if they are isomorphic.*

Interoperability and interoperation are similar (intersection of dependencies) with the difference that interoperation is a necessary condition for interoperability. Interoperation and interoperability are simply the specification of a CRM and a set of rules governing interactions with the CRM. Based on these findings, the fundamental question of interest in studying interoperation and interoperability is:

The fundamental question of interoperability*: Given a modeling question and a set of models can a CRM be identified?*

This question is important because of the implications carried by the answer. If the answer is yes it means there is potentially an algorithm that could take the modeling question and models as inputs and provide the corresponding CRM. If this algorithm is efficient is terms of time and/or memory space, it would mean that interoperability can be solved in general at a reasonable cost. Conversely, if the answer is no and a CRM cannot be identified it would mean that the best that can be done is to engineer a solution that would be the closest to answering the modeling question. The degree of closeness is determined by the modeler by inspection or by a metric such as time, space or correctness. Current approaches to interoperability (common framework and common standard) assume that the fundamental question of interoperability is decidable and solutions can be constructed.

The questions as posed can be mapped to *decision problems* that are well known and have been studied in *Computational complexity theory.* Complexity theory is an area of mathematics and computer science that is focused on studying and classifying computational problems based on criteria such as time and resources required to provide a solution (Hein, 2002). A decision problem is a type of computational problem in which a yes-no answer is provided based on a given input (Hein, 2002). In terms of classifications a problem is said to be:

Polynomial(P) if the answer to the question can be provided in polynomial time by a deterministic Turing Machine (computer).
Non deterministic polynomial (NP) if the answer to the question can be verified in polynomial time.
NP-Complete if the problem is in NP but there is no known efficient algorithm to solve it.
NP-Hard if the problem is at least as hard as NP-Complete problems.

The question of determining whether a CRM exist can be formulated as a decision problem in which the inputs are two or more models and the answer is *yes* they are isomorphic or *no* they are not. The determination of the existence of a CRM can be formulated as follows:

The fundamental question of interoperability reformulated*: Are two or more graphs isomorphic?*

This problem is known as the *graph isomorphism problem* for which it is not known whether it is in P or NP. A generalization of this problem known as the *subgraph*

isomorphism problem in which the input is two graphs and the question is whether a subgraph in one is isomorphic to a subgraph in the other. The *subgraph isomorphic problem* is NP-Complete. As a result, for the general case, there is no known efficient algorithm to find the answer to the first question. That is to say, regardless of the approach taken to generate or identify a CRM, there is no known way to efficiently find the CRM. Consequently with respect to a computer, all current approaches are equivalent in that they are providing heuristics to obtain a CRM. In terms of interoperation and interoperability, it is worth noting that the decision problems addressed so far only focus on structure and as such are a subset of the general problem of interoperability. The question as to whether two or more elements are identical still needs to be answered. This question as discussed in the previous section is reduced to comparing two or more sets of strings as there is no algorithm that can determine whether two conceptualizations are the same. As a result, interoperation and interoperability are at least as complex as NP-Complex problems, which lead to the formulation of the following observations:

Observation 1: *The determination of whether models interoperate is at least NP-Complete.*

Observation 2: *The determination of whether models are interoperable is at least NP-Complete.*

The overall complexity of interoperation and interoperability is possibly NP-Hard if one assumes an oracle that can identify equivalent elements instantaneously and an algorithm that resolves redundancies in polynomial time.

From this observation, all approaches to interoperability are equivalent in that they represent heuristics that approximate the CRM. In this section, we have shown that it is important to distinguish interoperation (the exchange of information) and interoperability (the ability to use the information once it is exchanged). Interoperation is a necessary condition for interoperability and both are only possible over the intersection of the existential and transformational dependencies. This observation means that the interoperability space diminishes as more models join a federation, contrary to the state of the art that points to an increase in the interoperability space as more models join a federation. The observations on the complexity of interoperability do not mean that interoperability is not possible, and in fact federations are developed constantly in practice. It just means that only a semi-automated approach will work and in the next section we show how to validate a federation of interoperating models assuming they can be composed (interoperable at the model level).

3 Formal Validation of Interoperable Federations

The previous sections presented what semantic interoperability means for machines and showed that it can only be accomplished through heuristics. This section focuses

on the validation of composed models as another way[2] to determine whether interoperating models are indeed representative of the modeling question one wants to have answered. In this chapter we talk about composition of models and interoperability of simulations in keeping with [17].

The validation of the composed artifact is of paramount importance to increase model credibility and user confidence in adopting composable models [18],[19]. A recent finding of the World Technology Evaluation Center states that "*without validation, computational data are not credible, and hence, are useless*" [20]. This is because simulation models are widely used to support critical decision-making [21], [22]. However, the validation process is often a lengthy, manual process that mandates the presence of at least one system expert. For example, the process of Verification, Validation, and Accreditation (VV&A) for modeling and simulation in the US Department of the Navy defines seven user roles and thirteen important steps grouped in five categories, namely, conceptual model validation, design verification, implementation verification, and results validation [23]. Furthermore, the costs of VV&A constitute a large part of the model development cost [21]. The benefits of an automated validation process of a composed model are significant.

The validation of interoperable federations is a non-trivial problem [19], [24]. This is because composition is not a closed operation with respect to validation since semantically valid components do not necessary form valid compositions [18]. Moreover, reused components are developed for different purposes and when composed may result in emergent properties [25]. Similarly, the context in which a reused component was developed and validated might differ from the new context of the composed model [24],[26]. Next, different validation perspectives must be considered, such as *logical* aspects of deadlock, safety, and liveness, *temporal* aspects such as the behavior of components and compositions over time, and *formal* aspects such as the need to provide a formal measure of the validity of compositions, also called "figure of merit"[27].

We define semantic validity as follows:

A composition of federations is semantically valid and its federations are said to be semantically composable if and only if:

(a) federations to be integrated behave correctly to form a valid composition both externally, with respect to their neighbors, and internally when safety and liveness properties are preserved over time, and

(b) the resulting composition produces valid output.

Studies of semantic composability validation show that the validity of a model is not a fixed point and there are many valid models but with different degrees of validity [28]. Current approaches to validate composed models are theoretically elegant but are not practical to implement [19] or are computationally expensive and thus do not

[2] The work on semantic composability validation is drawn from [39].

scale well [29]. Other approaches focus on the experimental model validation at the cost of reasoning about composition at conceptual levels [40].

We propose a dual-step elimination strategy for the validation of semantic composability, as shown in Figure 1.

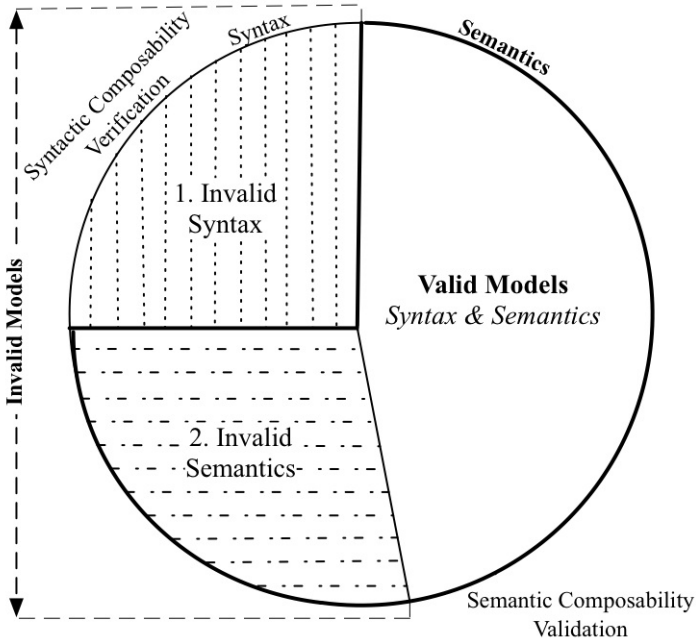


Fig. 1. Population of Composable Models

Our validation approach first discards invalid models through the validation of general model properties, such as safety and liveness for instantaneous and timed transitions [28][34]. We cover various perspectives on the definition of model properties, such as formal, practical, timeless, and timed, among others. Moreover, we propose a composability index as a measure of the degree of data alignment in the composition. Models that have passed the first validation step might still be invalid. Furthermore, to increase model credibility, formal guarantees and measures are required. Towards a formal guarantee of the composed model validity, we perform formal validation with respect to a reference model using a novel time-based formalism [28]. As a certificate of quality of the validity of the composed model, we introduce the semantic metric relation V_{ε}, which quantifies state similarities based on semantically-sugared components defined in our component-based ontology.

Our dual-step semantic validation process is shown in Figure 2.

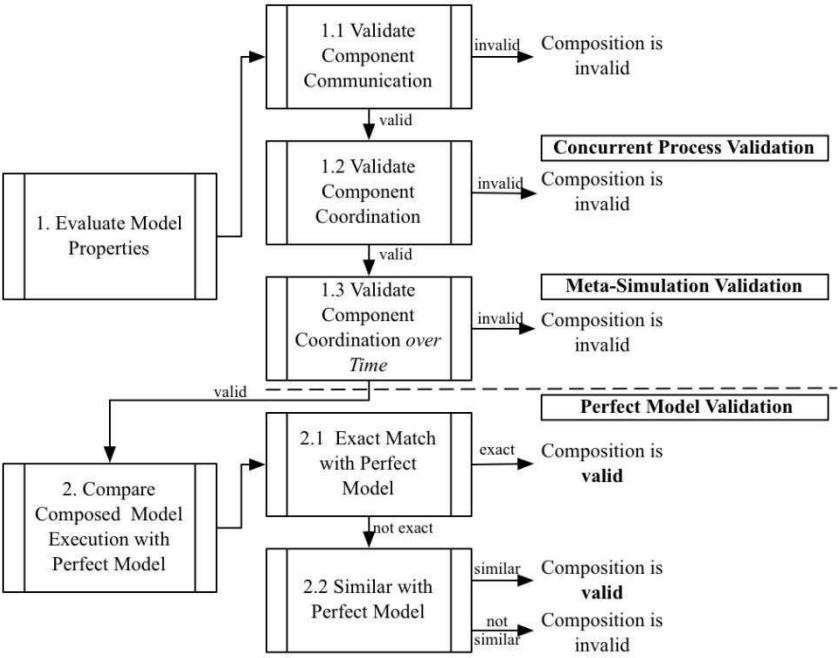


Fig. 2. Layered Validation Process

This section discussed formal validation of interoperable federations in the context of the CRM discussed above. The formal validation process calculates the closeness of the composed federation's execution to that of a reference model. The main assumption in the formal validation of semantic composability is that there exists interoperability between components, which is guaranteed by the CRM discussed above and in parts by the first layer, e.g 1.1. and 1.2., in the validation process presented above.

The proposed validation process employs semantic knowledge about the federation and its application domain, which is captured using a component-based ontology described in the next section.

3.1 Knowledge Representation

An important issue in addressing composability, in particular semantic composability, is expressing *domain or component knowledge* in an unambiguous, standardized format. An *ontology* is an organized knowledge representation to capture object information in a particular domain [30] in formats readable by humans and computers alike. Ontologies are conceptual models that capture and explain the vocabulary used in semantic applications guaranteeing communication free of ambiguities [31]. When

applied to the modeling and simulation domain, ontologies facilitate model discovery and integration and the development of formal methods for simulation and modeling [32]. Ontologies can be used to express syntax and semantics to facilitate communication and allow for automated semantic checking. Furthermore, they are employed to express the resource discovery request and determine whether the discovered model is reusable. An ontology should ideally focus on the description of a simulation component to facilitate semantic validation of compositions, as well as to support component discovery and reuse.

COSMO (COmponent Simulation and Modeling Ontology) [33] is an ontology for describing component-oriented simulation within and across application domains. COSMO semantically enriches the description of model components to support model discovery, model reuse as well as semantic composability validation of the discovered models. The ontology consists of sets of classes to describe simulation components and their compositions. The hierarchies in the COSMO ontology span two main directions, as shown in Figure 3. To achieve generality across application domains and at the same time support specific application domain requirements, we include first a simulation oriented component hierarchy, followed by an application domain oriented component hierarchy. The second set of classes describes components with respect to their *attributes* and *behavior* expressed as a state machine. We assume that irrespective of the simulation component's implementation and worldview, its behavior can be represented as a finite state machine initially provided by the component creator. Transitions in the state machine from an initial to a final state are triggered by an arrival event or an elapse in a time interval. The final state can be determined by some conditions on the component's attributes and the transition may produce output. The classes for attribute, behavior, worldview, transition, state, data, condition as well as simulation concepts such as time, distributions, etc. are defined in the ontology.

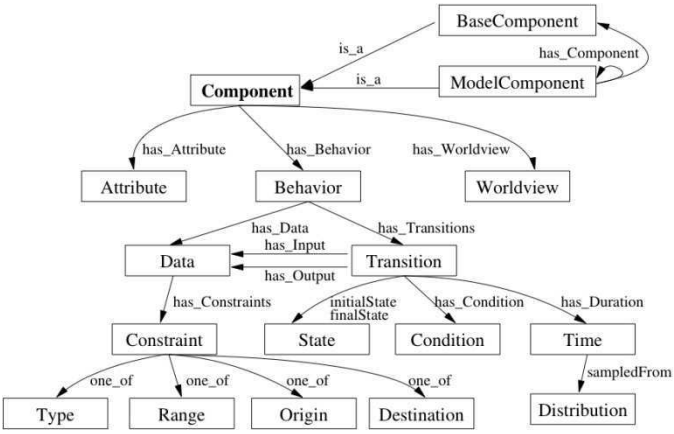


Fig. 3. Ontology for Component-based Simulation Development

In the construction of COSMO, we assume that the problem space is divided into several application domains. For each application domain, we further assume that

there exist several base components or base federations, which represent fundamental entities specific to the application domain. The base components form the atomic building blocks for each simulator in the application domain. For example, for the *Queueing Networks* application domain, we can assume the base components *Source*, which creates jobs, *Server*, which services the jobs created by the *Source*, and *Sink*, which collects all jobs from the *Server* components. Similarly, for a *Military Training Simulation* application domain, base components could be *Tank* and *SoldierTroop*. This separation into application domains facilitates reasoning about composition because application domain specific knowledge can be captured using ontologies (as shown in Figure 3) and using compositional grammars to describe syntactic composability. Furthermore, the validation process is meaningful and more accurate as we will see in the following.

3.2 Formal Validation of Model Execution

Assuming that a composed federation interoperates correctly, we validate the execution of the composed federation by comparing it to the execution of a reference model [28]. For this, we first propose a time-based formalism to represent federations as functions of states over time. Next, the formalism representing the composed federation is executed and compared with the execution of a reference model, considering semantic knowledge captured in the ontology described above.

We first present a discussion about the reference model and how it is obtained. An overview of our formal definitions of components, simulation, and validity in the context of the proposed five-step validation process follows.

3.3 Reference Model

The proposed validation process aims to provide a formal measure of composition validity by comparing the composed model with a reference composition made up of reference components or federations. We consider that for each type of base component or base federation there exist a reference model in the repository, initially provided by domain experts. The reference base component models describe what the domain experts consider to be the ideal component behavior. The generic descriptions lack specific attributes (e.g. sampling distributions for time attributes) and are without an implementation. We assume that for each base component type (e.g. *Source* in Queueing Networks Application domain) there exist different base component implementations in the repository (e.g. *SourceOpen* - a *Source* component for open queueing network systems). A reference component is a generic, desirable representation of a base component ideally provided by domain experts when the new application domain is added to the framework. Ideally, the reference components should describe what the system experts consider to be the desirable base component behavior. It should be generic in the sense that their description lacks any real data values. It follows that the reference model composed from the generic reference components is only a description of the desired simulation, without an attached implementation. Throughout the validation process, the generic reference components attributes will be instantiated using the same attribute values used by the

corresponding components in the composed model. The base component implementations may differ widely from the reference base component models.

3.4 Formal Validation Process

Figure 4 presents our five-step validation process [28].

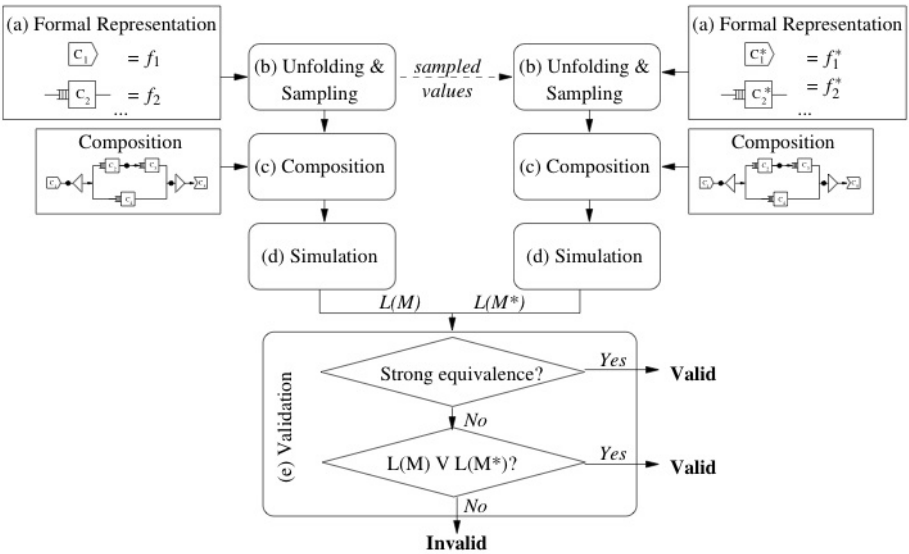


Fig. 4. Formal Validation Process

The first four steps of the validation process, namely *Formal Component Representation*, *Unfolding and Sampling*, *Composition*, and *Simulation* are applied separately to the components and reference components. Components and reference components (annotated with a star symbol (*)) from the composition and reference composition respectively, are formally represented as functions of their states over time. The formal component representations are input to the *Unfolding and Sampling* step, in which the component representation is adjusted to fit our validation process. Based on the composed model topology, the unfolded representations obtained from the *Unfolding and Sampling* step are composed as mathematical functions in the *Composition* step. The *Simulation* step applied to the composition and reference composition results in a composition simulation, *L(M)*, and reference composition simulation, *L(M)*. The Composition step formally composes the functional representations based on our mathematical composability definition, which considers the time moments when the functions are activated. As such, *L(M)* and $L(M^*)$ consist of time-ordered simulation schedules of the function executions. Lastly, in the *Validation* step, we first attempt to determine whether *L(M)* and $L(M^*)$ are exact matches. This is done by determining strong equivalence between *L(M)* and $L(M^*)$. If strong equivalence is not possible, we introduce the semantic relation V_ε to determine

weak equivalence only between related states, i.e. the parts in the two executions that are semantically related. If V_ε is not a weak bisimulation relation between *L(M)* and *L(M*)*, then the model is invalid. Figure 5 illustrates the approach using a single-server queue example.

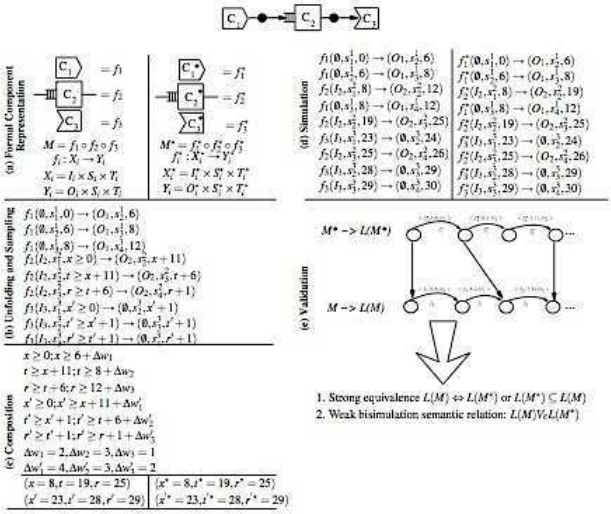


Fig. 5. Formal Model Execution Validation

The five steps in our validation process are as follows:

a) *Formal Component Representation* – In this step, federations are formally represented as mathematical functions of states over time as below.

The formal representation of a simulation component F_i is a function $f_i: X_i \rightarrow Y_i$, where $X_i = I_i \times S_i \times T_i$, and $Y_i = O_i \times S_i \times T_i$. I_i and O_i are the set of input/output messages, S_i is the set of states and T_i is the set of simulation time intervals at which the component changes state.

b) *Unfolding and Sampling* – This step unfolds the function definition from above over the simulation time using sampled values. Normally, the function definition in Step a) is a cyclic expression which gets executed many times through the simulation execution time *T*. For example, for a *Source* component, the formal representation would be: $f(\emptyset, s_i, t) \rightarrow (O_i, s_i, t + \Delta t)$, where Δt is sampled from an exponential distribution. This expression is not useful because *f* will get called many times throughout the simulation execution time *T*. As such, sampling for values of Δt is performed and the function execution is unfolded. We perform this operation for an unfolding degree τ number of times. Details are presented in [28] and [34]. Step a) and

b) are applied for the reference components in a similar manner, using the same sampled values used for the composed federations.

c) *Composition* – The Composition step validates that the functions are mathematically composable. We obtain constraints on the time variables obtained in the unfolding step considering a simple observation derived from the connection of the federations, namely, that if F_i is connected with F_j with F_j requiring input from F_i, then F_j cannot execute its transitions that require input until has F_i produced output. For federations that require input to proceed, we also consider the average time spent by messages from the senders to the receiver. The equations obtained for the time attributes are solved using an open source constraint solver such as Choco [35]. The equations are solved both for the composed federations and the composed reference model. The function calls are then ordered based on the solutions for the time attributes.

d) *Simulation* – Based on the time values obtained in the Composition step, an interleaved simulation run is obtained for the composed federation simulation and for the reference model. The simulation runs are represented as Labeled Transition Systems (LTS) [36], $L(M)$ and $L(M^*)$, as follows. Given a composed model M and its simulation $S(M)$. The simulation run $\boldsymbol{S}$ is represented as a LTS where nodes represent the entire composition state as a reunion of the individual component states, and edges are labeled to facilitate the validation process. To facilitate accurate comparison between $L(M)$ and the reference LTS $L(M^*)$, the edge labels contain the name of the function called to exit the node, its duration, and its output: $<function\_name(f_{out}), duration(fout), output(f_{out})>$. We consider the *duration* rather than the *time* moment when f_{out} begins to execute, because the time moments at which the functions f_{out} start to execute are already ordered through the directed nature of simulation $\boldsymbol{S}$.

e) *Validation* – This step validates the composed federation against the composed reference model. We consider two possible relations between the simulation of the composed model and the simulation of the reference model, $L(M)$ and $L(M^*)$ respectively: strong equivalence relation [37] and our proposed semantic parametric metric relation, V_ε. Informally, strong equivalence between $L(M)$ and $L(M^*)$ validates that $L(M)$ is exactly the same or included in $L(M^*)$, including the sequence of the function calls and the edge labels. If this is not possible, we propose the semantic parametric relation V_ε as a weak bisimulation relation. V_ε considers only parts of $L(M)$ and $L(M^*)$ that are semantically close and validates that they appear in the same sequence in $L(M)$ and $L(M^*)$. Semantic closeness considers attributes that are close or similar in COSMO ontology. For such attributes, we assign a higher weight to those that have the same value in the model and the reference model or have been showing the same modification trend.

4 Summary and Recommendations

In this chapter, we presented and defined semantic interoperability and discussed what it means for machines. Interoperability based on the definitions provided cannot be fully automated so heuristics are required to fully support federations. Another approach to interoperability and composability is to assume the existence of a perfect model, develop a federation and validate it against the perfect model. The chapter shows how to define and validate a federation of models using ontology and semantic similarity metrics. The authors identify two main directions for future research. The first area focuses on the development of formal theory of interoperability that includes not only data interoperability but also an algebra that captures the existential and transformational dependencies. Such algebra will move the community a step closer to understanding interoperability beyond data. One of the main observations is that interoperability being the intersection of models is a limiting factors and therefore it is better to have models (and machines) collaborate to answer a given question rather than have them interoperate and produce the least common denominator. This algebra will help define criteria for collaboration and orchestration of models.

The other area of research is how to develop and validate the next generation of models. So far, the community assumes that a model is necessarily a model of reality and models derived from the same referent are equivalent. However, this is not true in the general case, especially in a post-positivist world in which we are interested in more than physical objects. The idea of validation for instance rests on the existence of a perfect model usually a physical one. What if we do not have access to that model? Further, what if the goal is to define the main characteristics of the perfect model in order to generate sub models of interest (this is particularly true when one uses simulation to develop theory). The authors encourage the reader to reflect on the use of agents and M&S in support of training and experimentation which is prevalent today and to also help prepare a paradigm shift towards the use of agents and M&S as a mean to gain new insights into physical and non-physical entities (Human Social Cultural Behavior for instance). The authors, as many others believe that this new paradigm is the way of the future and it is one that presents the most challenges for the scientific community.

References

1. Davis, P.K., Anderson, R.H.: Improving the Composability of Department of Defense Model and Simulations. RAND National Defense Research Institute (2004)
2. Hofmann, M.A.: Challenges of Model Interoperation in Military Simulations. Simulation 80, 659–667 (2004)
3. Benjamin, P., Akella, K., Verna, A.: Using ontologies for simulation integration. In: Winter Simulation Conference, pp. 1081–1089 (2007)
4. Webster's Online Dictionary. Interoperability (2008)
5. Department of Defense Website, `http://www.dtic.mil/doctrine/jel/doddict/data/i/02802.html`
6. IEEE: A Compilation of IEEE Standard Computer Glossaries, New York (1990)
7. ISO/IEC: International Technology for Learning, Education, and Training, Geneva (2003)
8. Ackoff, R.: From Data to Wisdom. J. of Applied Syst. Analysis 16, 3–9 (1989)

9. Rowley, J.: The Wisdom Hierarchy: Representations of the DIKW Hierarchy. J. of Information Science, 163–180 (2007)
10. Chen, P.P.: The Entity-Relationship Model – Toward a Unified Data View. ACM Transactions on Database Systems 10(3), 9–36 (1976)
11. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. Communications of the ACM 13(6), 377–387 (1970)
12. Lipski, W.: On Semantic Issues Connected with Incomplete Information Databases. ACM Transactions on Database Systems (TODS) 4(3), 262–296 (1979)
13. Turing, A.: Computing Machinery and Intelligence. Mind, 433–460 (1950)
14. Hein, J.: Discrete Structures, Logic, and Computability, 2nd edn. Jones and Bartlett Publishers, Inc., USA (2002)
15. Sowa, J.: Top Levels of the KR Ontollogy, `http://www.jfsowa.com/ontology/toplevel.htm`
16. Dori, D.: Object-Process Methodology – A Holistic Systems Paradigm. Springer, New York (2002)
17. Page, E.H., Briggs, R., Tufarolo, J.A.: Toward a Family of Maturity Models for the Simulation Interconnection Problem. In: Spring Simulation Interoperability Workshop (2004)
18. Balci, O.: Verification, Validation and Accreditation of Simulation Models. In: Winter Simulation Conference, pp. 135–141 (1997)
19. Petty, M., Weisel, E.: A Composability Lexicon. In: Simulation Interoperability Workshop (2003)
20. Glotzer, S., Kim, S., Cummings, P., Deshmukh, A., Head-Gordon, M., Karniadakis, G., Petzold, L., Sagui, C., Shinozuka, M.: WTEC Panel on International Assessment of Research and Development in Simulation-based Engineering and Science. World Technology Evaluation Center, Maryland (2009)
21. Hartley, D.S.: Verification, Validation in Military Simulations. In: Winter Simulation Conference, pp. 925–931 (1997)
22. Min, F., Ma, F., Yang, M.: A Knowledge-based Method for the Validation of Military Simulation. In: Winter Simulation Conference, pp. 1395–1402 (2007)
23. NAVMSMO: Depatment of the Navy, Modeling and Simulation Verification, Validation, and Accreditation Implementation Handbook (2004)
24. Tolk, A., Muguira, J.: The Levels of Conceptual Interoperability Model (LCIM). In: IEEE Fall Simulation Interoperability Workshop (2003)
25. Gore, R., Reynolds, P.F.: Applying Causal Inference to Understand Emergent Behavior. In: Winter Simulation Conference, pp. 712–721 (2008)
26. Bartholet, R.G., Brogan, D.D., Reynolds, P.F., Carnahan, J.C.: In Search of the Philosopher's Stone: Simulation Composability Versus Component-Based Software Design. In: Fall Simulation Interoperability Workshop (2004)
27. Kasputis, S., Ng, H.G.: Composable Simulations. In: Winter Simulation Conference, pp. 1577–1584 (2000)
28. Szabo, C., Teo, Y.M.: A Time-based Formalism for the Validation of Semantic Composability. In: Winter Simulation Conference, pp. 1411–1422 (2009)
29. Traore, M.K.: Analyzing Static and Temporal Properties of Simulation Models. In: Winter Simulation Conference, pp. 897–904 (2006)
30. Sowa, J.F.: Knowledge Representation: Logical, Philosophical and Computational Foundations. Brooks/Cole, Pacific Grove, CA (1999)
31. Breitman, K.K., Casanova, M.A., Truszkowski, W.: Semantic Web: Concepts, Technologies and Applications. Springer, London (2007)

32. Miller, J.A., Fishwick, P.A.: Ontologies for Modelling and Simulation: Issues and Approaches. In: Winter Simulation Conference, pp. 259–264 (2004)
33. Teo, Y.M., Szabo, C.: An Integrated Approach to Composable Modeling and Simulation. In: Annual Simulation Symposium, pp. 103–110 (2008)
34. Szabo, C., Teo, Y.M.: On the Validation of Semantic Composability in Data-Driven Simulations. In: ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation, pp. 73–80 (2010)
35. Sourceforge.: Choco: A Constraint Programming System (2010), `http://sourceforge.net/projects/choco/`
36. Srba, J.: On the Power of Labels in Transition Systems. In: International Conference on Concurrency Theory, pp. 277–291 (2001)
37. Park, D.: Concurrency and Automata on Infinite Sequences. In: GI-Conference on Theoretical Computer Science, pp. 167–183
38. Diallo, S.Y.: Towards a Formal Theory of Interoperability. Old Dominion University, Norfolk (2010)
39. Szabo, C.: Composable Models and Their Formal Validation. National University of Singapore (2010)
40. Leye, S., Uhrmacher, A.M.: A Flexible and Extensible Architecture for Experimental Model Validation. In: Proceedings of SIMUTools (2010)

Chapter 4
An Approach to Knowledge Integration Applied to a Configuration Problem

Maria Vargas-Vera[1], Miklos Nagy[2], and Dietmar Jannach[3]

[1] Computing Department
The Open University, UK
mvargasvera@gmail.com
[2] Knowledge Media Institute(Kmi)
The Open University, UK
M.Nagy@open.ac.uk
[3] Computer Science Department
Technical University of Dortmund, Germany
Dietmar.Jannach@tu-dortmund.de

Abstract. This chapter presents a framework for knowledge integration based on mappings between similar concepts in constraint graphs associated to a configuration problem. In particular, the chapter is devoted to one of the problems which could arise when performing collaborative knowledge integration, namely detecting knowledge overlaps. The solution to the overlapping problem relies on the use of matching algorithms. To illustrate our approach we present as a case study a computer configuration problem. This problem is important as it has the promise to become an alternative approach for the current knowledge integration solutions. Through our approach the real cost of integration can be reduced as it is not necessary to invest a great amount of resources beforehand a truly integrated system can be operational.

Keywords: Knowledge integration, Semantic Web, Constraint programming.

1 Introduction

An important challenge in several fields ranging from design of expert systems to collaborative design construction in engineering is to integrate several sources of knowledge created by different stakeholders. This is not a new problem as it has been around for several decades i.e. CAD (Computer-Assisted Design) systems. In our proposal, we attempt to design a new solution approach, which amalgamates research outputs from Ontology Alignment (in particular distributed solutions) and Fuzzy Logic in order to provide an alternative solution to the problem of knowledge integration.

There exists no unique definition of knowledge integration. Therefore, we need to first clarify the different meanings of the term Knowledge Integration. The definitions found in the fields of Artificial Intelligence, Ontologies, Databases and Knowledge Management vary strongly. For example, in the Artificial Intelligence

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 75–105.
springerlink.com

community, Knowledge Integration is seen as the process of integrating knowledge into an existent body of knowledge [24]. Knowledge integration refers to the identification of how new and prior knowledge interact while incorporating new information into a knowledge base. In contrast, the view taken from Knowledge Management is that knowledge integration is a fundamental management practice. According to Grant [15], the organization's primary concern is the integration of its dispersed knowledge resources in order to apply them to a "production of a new artifact" as a mean of creating new knowledge out of novel combinations of existing knowledge [15].

In the context of Artificial Intelligence, the problem of knowledge integration can also be seen as a scenario of building distributed knowledge bases in a collaborative way. These knowledge models need to be integrated into a single model. However, while integrating knowledge several problems could arise. One of them is the problem of overlaps and conflicts in the knowledge as subtle and unexpected interactions of knowledge could appear with the newly added knowledge [24]. However, in our opinion knowledge integration is the process of creating an unified knowledge model by means of integrating individual models made by different knowledge engineers. This integration is basically a reconciliation of the terms and relations used by each knowledge engineer while building their own model.

Furthermore, the basic argument is that knowledge cannot be viewed as a simple conceptualization of the world, but it has to represent some degree of interpretation. Such interpretation depends on the context of the entities involved in the process. This idea is rooted in the fact the different entities' interpretations are always subjective, since they occur according to an individual schema, which is than communicated to other individuals by a particular language. These schemas called mental spaces, contexts, or mental models have been investigated in the past [10] [14] [20].

The motivating scenario of our work is that we assume that large knowledge bases are typically constructed by different knowledge engineers or domain experts in an incremental and collaborative way. However, as the individual parts of the knowledge bases may also be developed independently by different teams or organizational units, one or more knowledge integration phase are required in the overall process in order to detect and resolve conflicts and overlaps. The development of a knowledge base for a product configuration system [30] is a typical example as different organizational units contribute technical and process- or marketing-related constraints on legal product constellations. The problem is even harder, when the configurable product is delivered by multiple providers in a supply-chain [4], and requires the cross-company integration of knowledge bases and interfaces. Another example, the creation of a 3D-virtual campus where several departments of the Open University (OU) collaborated in the whole design of the campus i.e. Library, Research School, Computing Department, among others [29].

Our suggested approach to knowledge integration deals, in a first instance, with the overlapping problem. The detection of the overlapping problem is performed by mappings between the knowledge models. Therefore, the main contribution of this chapter is to propose as a solution to the overlapping problem

based on matching algorithms which use Dempster-Shafer and Fuzzy Voting Model. An scenario to illustrate the knowledge integration using DSSim best methods is outlined in our case of study.

The chapter is organized as follows. Section 2 provides an overview of related work. Section 3 presents a case scenario that illustrates the overlapping problem when performing knowledge integration. Section 4 describe details of the mapping process. Section 5 shows our framework to knowledge integration. Section 6 presents an evaluation of our methodology to knowledge integration. Finally, in Section 7 we present our conclusions and describe our future work.

2 Related Work

Several research communities have investigated the information integration problem. This lead to numerous different approaches in a way in which different information sources can be integrated. After an analysis of the literature we have identified four perspectives on our literature review. These perspectives are Knowledge Based Systems, Ontologies, Databases and Knowledge Management. The first perspective deals with problems in knowledge modeling in particular in expert systems. The second perspective is the work in the Ontologies field ranging from ontology merging to alignment. The third perspective, Databases, is more related to data integration which consists of providing an unified view on the data stored in different databases with different models. The Knowledge Management perspective is not explored in too much detail as is not the main focus of this chapter.

2.1 Expert Systems - Knowledge Bases

Murray [23] presents an approach to knowledge integration as a machine learning task. He implemented a system called REACT which is a computational model that identifies three activities. (1) "Elaboration": describes, how new and prior knowledge interact, although this feature is restricted to focus only on selected segments of prior knowledge. (2) "Recognition", which identifies the consequences of new information for relevant prior knowledge and (3) "Adaptation", which exploits the learning opportunities by modifying the new and prior knowledge. A learning opportunity occurs when a property of a particular object in the learning context can be generalized into a property for every instance of a class of objects. Empirical evidence indicates that indeed knowledge integration helps knowledge engineers to integrate new information into a large knowledge base.

Knowledge Integration has became an essential element in the Semantic Web Community. For example, knowledge integrations allows to access services which offer knowledge contained in various distributed databases associated with semantically described web portals. In this context Zygmunt et al., propose a

framework for knowledge integration supported by using an agent-based architecture [35]. The approach relies very much on the integration of ontologies by the gradeAgent which estimates the similarity between classes and properties in the ontology. The approach uses algorithms of lexical and structural comparison. The checking of similarity between larger parts of a graph is performed with the use of Similarity Flooding algorithm. The approach also applied additional techniques based on a thesaurus when looking for synonyms and on the use of high level ontology to adjust concepts from the ontology to a given set of concepts which identify important notions. The framework does not handle uncertainty in the similarity metrics. In principle, it seemed as a good solution but in real scenarios the notion of uncertainty limited to a crisp mappings 0 or 1 made a strong limitation in a proper identification of matching concepts and properties.

2.2 Ontologies View

The knowledge engineering community uses ontologies as the main approach for resolving semantic differences in heterogeneous data sources. Based on this approach several categories can be identified to Data Integration. One of them is to create a global ontology. In this way all the different sources share the same ontology in order to make the information integration possible. These solutions fit well when the number of sources is limited and a consensus can be achieved between partners. However, for real life scenarios, this solution is inflexible in nature and is not considered as a viable alternative in the context of knowledge integration.

Ontology merging aims to achieve semantic integration through merging different source ontologies into a consistent union of the source ontologies. These systems make use of the fact that different ontologies have overlapping fragments that is the basis of the merging process. FCAMERGE [13] offers a global structural approach to the merging process. It takes the source ontologies and extracts instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances the system apply formal concept analysis techniques to derive a lattice of concepts as a structural result of merge process. The produced result is explored and transformed to the merged ontology by the ontology engineer. PROMPT [11] is a semi-automatic ontology merging tool that makes initial suggestions based on linguistic similarity between class names then performs automatic updates, finds new conflicts and makes new suggestions.

Ontology mapping aims to achieve semantic integration through the creation of mappings between concepts, attributes etc. between two ontology entities. Based on database schema integration solutions a wide range of techniques has been proposed from manually defined rules to semi automatic approaches that make use of machine learning, heuristics, natural language processing and graph matching algorithms. MAFRA [1], a mapping framework for distributed ontologies supports in interactive, incremental and dynamic ontology mapping process

in the Semantic Web context. The main contribution of this approach is that it creates a true distributed ontology mapping framework that is different from mediator based approach. GLUE [2] evolved from a mediator based LSD [9] data source schema matching, applies machine learning techniques and similarity measures based on joint probabilistic distributions.

Since ontology mapping problem is one of the first steps in the direction of Semantic Web based data and information integration it has become an active research topic recently. As a consequence numerous ontology mapping systems have been proposed but only a handful of them have participated in the Ontology Alignment Initiative (OAEI)[1], which serves as a comparison benchmark for such systems. ASMOV) [19] is an automatic ontology mapping approach, which carries out the mapping in two phases. In the first phase different similarity measures are calculated and combined in order to establish preliminary mapping pairs. In the second phase the system carries out a semantic verification, in order to detect semantically inconsistent mappings and their causes. RiMOM[34] is an ontology mapping approach that uses the combination of different strategies in order to achieve the good results. The different strategies are selected based on the characteristics of the source ontologies and the pre-defined rules. Anchor-Flood [31] is an ontology mapping system, which has been developed in the context of International Patent Classification (IPC) in order to exploit the available taxonomy of related terms in an abstract and align it with the taxonomy of IPC ontology. The mapping is done in two phases. First part is the ontology mapping, where the concepts and properties in the different ontologies are aligned. The second part of the mapping process is the mapping of the instances of the ontologies. Anchor-Flood approach assumes that neither ontology concepts nor an instance comprises the full specification in its name or URI alone. TaxoMap [17] is an ontology mapping tool, which was designed to support information integration between different sources. The mapping process is oriented from ontologies that describe external resources (named source ontology) to the ontology (named target ontology) of different web portals. However the system design assumes that target ontology is supposed to be well-structured, whereas source ontology can be a flat list of concepts. Therefore TaxoMap heavily relies on the labels it uses a moropho-syntactic analysis for tagging text with part-of-speech and lemma information and a similarity measure which compares the trigraph of the concept labels. Lily [33] is an ontology mapping system for integrating information described by heterogeneous ontologies. The system employs hybrid matching strategies to create the mappings for both normal and large scale ontologies.

Summing up the suggested solutions ranges from creation of global views of ontologies, mapping or combining. However we believe that the creation of a global view (or global ontology) is a limited solution as it seems to work when the number of models is limited. The mapping solution is more appealing as the combining solution seems to be not scalable.

[1] http://oaei.ontologymatching.org/

2.3 Databases

In the database community several solutions have been proposed. However not all approaches [6] have been implemented in real life applications. The characteristics of these approaches are that they all have inputs and outputs, which is supplied or processed by a human designer. The inputs are usually the domain models including entity relationships, views and sometimes queries whereas the outputs are conceptual models, global schemas, mapping rules or conflicts. The majority of approaches based on a mediator architecture that involve logical database schemas, which are used as shared mediated views over the queried schemas. A number of systems have been proposed e.g. TSIMMIS [12], Information Manifold[21], InfoSleuth [7], MOMIS [8] , LSD [3] that shows the flexibility and the scalability of these approaches. In particular, MOMIS is focused a data integration from scientific data sources but it also been applied to other domains like building a tourism information provider [8]. The problem, however, is that these solutions rely on the initial idea of database schema integrations namely to create a global view, which will be used as a mediator between the different sources. According to Halevy [16] the major bottleneck in setting up a data integration framework in databases community is the effort required to create the source description and more specifically writing the semantic mappings between the sources and the mediated schema. Of course, we share the same opinion and this can be understood as we expand more in this issue.

The database integration schema's solution requires that an integrated global schema is designed from the local schemas, which refers to existing databases. This global schema is a virtual view of all databases taken together in a distributed database environment. The conceptual modelling of a database (DB) schema is mainly based on Entity-Relationship (ER) model or Unified Modelling Language (UML) class diagrams. There are two design approaches namely direct and gradual. In the direct approach, user requests are processed all at once and the whole DB schema is created directly. This approach is appropriate in the cases of designing small DB schemas, but it is inappropriate in cases when a complex DB schema should be integrated. The gradual approach is used when the number and complexity of user requests are beyond the designer's power of perception. Design of a complex DB schema is based on a gradual integration of external schemas. An external schema is a structure that, at the conceptual level, formally specifies the user's view on a DB schema. Each transaction program that supports a user request is based on an external schema, which is associated to it. After being created, external schemas are integrated into a conceptual DB schema. Nevertheless this idea has been developed further once the ontologies have been proposed as described in section 2.2.

2.4 Knowledge Management

Hung [18] present an empirical study that investigates the patterns of knowledge integration in the collaborative development of system on a chip (SoC) by semiconductor firms. The study focused on the central interactive process

for engineering applications and experimental practice to enhance knowledge integration and technology innovation for rapid product development. A process model for knowledge integration via experimental practice is presented; further explanation can be found in [18]. The process of knowledge integration is triggered by new requirements i.e. new product features or testing methods, which cannot be resolved based on the current knowledge. This integration process depends upon knowledge already existing in the organization as well as new external knowledge. The outcome of the process is a technological innovation and the fact that the knowledge of the organization is enhanced by means of knowledge integration. The Knowledge Management perspective which appears related to our work is the one based on the Distributed Knowledge Management (DKM) approach explored in the Knowledge Management community [22], in which subjective and social aspects of the real world are taken into account. However, this perspective is not going to be explored as is out of the scope of this chapter.

Summing up we could say that there are some commonalities between the four perspectives namely Knowledge Based Systems, Ontologies, Databases and Knowledge Management. The work on knowledge based systems community is concerned with interaction of knowledge when a new piece of knowledge is found and added to the knowledge base. The work on knowledge integration found in the ontology merging community appears to be suitable for detecting overlapping between different knowledge bases. The work in the Database community is more concerned with data integration. Finally, the fourth one is however more related to a processes/products specifications and organizational aspects, which need to be modified when adding new knowledge as the solution relies in adapting a known case to the new situation.

3 Scenario

The scenario presented in this chapter illustrates the problem of different views of knowledge modeled by different departments within an organization. Let us imagine the scenario where we have a Computer manufacturing firm formed with three departments, for example, Sales, Technical Design Manufacturing and Software Department. Each of these departments have already pre-established functions within the organization. These functions are not shown explicitly in Figure 1 but they are part of the box Enterprise Resource Planning (ERP). The Configuration Logic stores the CSP (Constraint Satisfaction Problem) applied to Computer Configuration.

Customers request a computer with certain specification using the interface provided by the “configuration system” then as, a second step, the “configuration system” returns “Quotes” and then the customer could make an order using the on-line system.

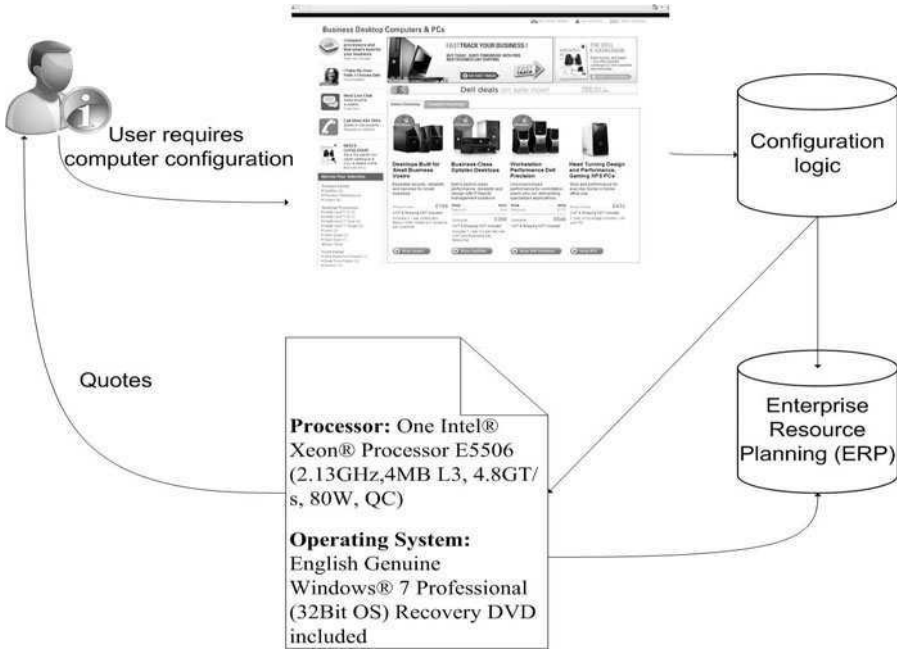


Fig. 1. Computer configuration system

3.1 Constraint Satisfaction Problem (CSP)

This section gives firstly a brief descriptions of Constraint Satisfaction Problem (CSP) and related terminology and secondly our Case Study namely computer configuration problem using the definition given in section 3.1.

Definition 1. *A CSP is a triple* $P = \{V, D, C\}$, *where* $V = \{v1, v2, ..., vn\}$ *is the set of variables called the domain variables;*

$D = \{D1, D2, ..., Dn\}$ *is the set of domains. The domain is a finite set containing possible values for the corresponding variables;*

$C = \{c1, c2, ..., cn\}$ *is the set of constraints. A constraint ci is a relation defined on a subset of* $\{vi, ..., vk\}$ *of all the variables; that is,* $\{Di, ..., Dk\} \supseteq ci$.

The structure of a CSP may be represented by a constraint graph, which is defined as follows: variables are represented with nodes, and the constraints between them are represented with edges. The labels of the edges represent the constraints and the labels of the nodes represent the domain of the variables.

Definition 2. *Assignment: It is a mapping from a set of variables to their corresponding domains. Let* v_i *be a variable and* D_i *its domain. The process that* v_i *takes a value, say* d_i *from the domain* D_i *is called assignment. Such an assignment is denoted* (v_i, d_i).

For a CSP problem which has a set of variables, say $v_1, v_2, ..., v_m$ the assignment for all the variables is denoted $\{(v_1, d_1), (v_2, d_2), ..., (v_m, d_m)\}$. When all the variables are assigned a value, the assignment is called complete, otherwise partial.

3.2 Case Study: Computer Configuration Problem

The case of study proposed in this chapter is a restricted version of a computer configuration problem. The problem of configuration is defined as a CSP (Constraint Satisfaction Problem) problem where we define our model using Variables, Domain for the variables and Constraints over the variables. The main goal is to obtain an assignment (i.e. a value for all the variables). The CSP is defined formally as follows:

Variables:

Table 1. Variables

V1	OS
V2	Memory
V3	Hard_disk_size
V4	CPU
V5	Monitor
V6	Mouse
V7	Video_card
V8	Graphics_card
V9	Gaming_PC
V10	Keyboard
V11	Monitor_resolution

Table 2. Domain

D1	OS	Vista, XP, MAC-OS, Windows_7,Linux
D2	Memory	512_MB, 1024_MB, 2048_MB, 3072_MB
D3	Hard_disk_size	160_GB, 180_GB, 320_GB
D4	CPU	Pentium_4, Intel_Centrino
D5	Monitor	14_inches,18_inches,19_inches,20_inches
D6	Mouse	Logitech, Magic_mouse
D7	Video_card	NVIDIA_600series, NVIDIA_700series, NVIDIA_800series
D8	Graphics_card	GeForce_7600series, GeForce_7800series, GeForce_7900series
D9	Gaming_PC	yes,no
D10	Keyboard	Win_keyboard,Mac_Keyboard
D11	Monitor_resolution	low,medium,high

Constraints:

Table 3. Constraints

C1	$IF\ OS = \text{“}XP''\ THEN\ Memory \leq 2048\_MB$
C2	$IF\ Monitor = \text{“}20\_inches''\ THEN\ Graphics\_card = \text{“}GeForce\_7800\_series''$
C3	$IF\ OS = \text{“}XP''\ THEN\ CPU = \text{“}Pentium\_4''$
C4	$IF\ OS = \text{“}Vista''\ THEN\ CPU = \text{“}Pentium\_4''$
C5	$IF\ OS = \text{“}XP''\ THEN\ Hard\_disk\_size \geq \text{“}500\_MB''$
C6	$IF\ Gaming\_PC = \text{“}yes''\ THEN\ Graphics\_card = \text{“}NVIDIA\_8000series''$
C7	$IF\ Gaming\_PC = \text{“}yes''\ THEN\ Memory \geq \text{“}2048''$
C8	$IF\ Gaming\_PC = \text{“}yes''\ THEN\ Hard\_disk\_size \geq \text{“}160GB''$
C9	$IF\ Monitor \geq 20inches\ THEN\ Monitor_Resolution = \text{“}high''$
C10	$IF\ OS = \text{“}MAC - OS''\ THEN\ Keyword = \text{“}Mac\_Keyboard''$

3.3 Constraint Graph

In order to represent the problem we use graphs, which can be defined as follows:

Definition 3. *Constraints are represented in a graph called constraint graph. Each node in this graph is labelled by a variable name together with a set of possible values for that variable. A directed constraint connected(i,j) connects a pair of nodes i and j if the value of the variable labeling i is constrained by the value of the variable labeling j.*

To illustrate our approach to mapping, we have taken the initial constraints graphs built by different engineers using different knowledge models. These original graphs hold by our individual departments are depicted in Figure 2 and 3. These graphs use standard computer jargon although, they have discrepancies on the name of variables used. The term *Video_card* and *Graphics_card* (variable names) were used by different knowledge engineers to refer to the same concept. The latest problem suggested that in order to perform knowledge integration we have to perform mappings between nodes in the constraints graphs. For the sake of clarity, we only presents overlaps in one node of the graph but this is not always the case. Figure 2 uses as variable name called *Video_card* whilst in Figure 3 the variable name is *Graphics_card*.

A unified view of two constraints graphs was produced (manually) by joining two initial constraints graphs. This unified view is depicted in Figure 4. In one hand, Figure 4 partially shows a constraint graph with nine variables namely OS, Memory, Hard_disk_size, CPU, Monitor, Gaming_PC, Video_card, Graphics_card and Monitor_Resolution.

4 Mapping Process

The main objective of the mapping is to identify that nodes in the constraint graphs are equivalent e.g. the “Video card” node (shown in Figure 3) is equivalent

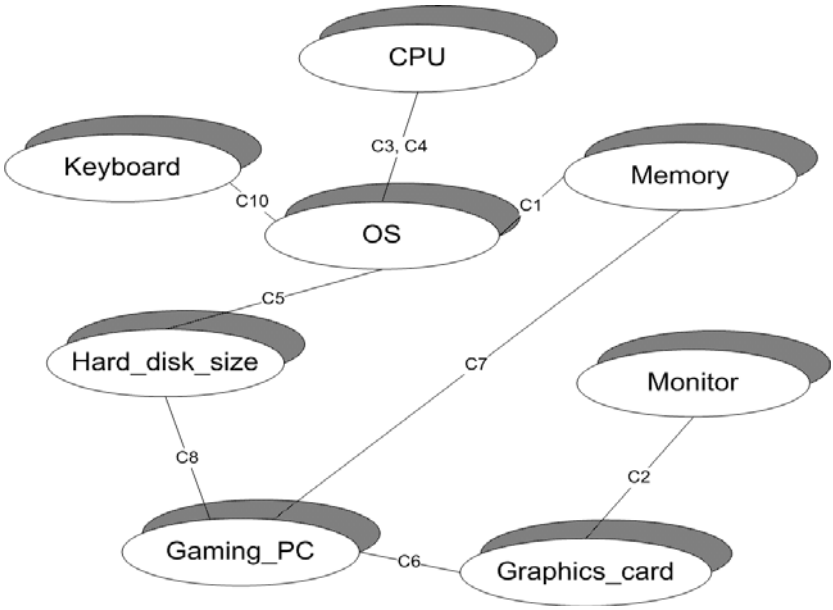


Fig. 2. A Constraint graph for the computer configuration problem using variable Graphics_card

with the "Graphics card" node presented in Figure 2. In order to proceed with the comparisons, we need to compare all possible node combinations of the graphs shown in Figure 2 and Figure 3 and select the ones, which are the most similar or nothing if there is no similarity between the nodes.

Once the constraint graphs have been established the system need to establish mappings between the hardware items represented as nodes in the graph. The problem can be represented as the ontology-mapping problem in order to find correspondences between the items. The objective of the ontology mapping is to use different similarity measures in order to establish the mappings. However in practice one similarity measure or some technique can perform particularly well for one pair of concepts or properties and particularly badly for another pair of concepts or properties, which has to be considered in any mapping algorithm. In our ontology-mapping approach we use different software agents where each agent carries only partial knowledge of the domain and can observe it from its own perspective where available prior knowledge is generally uncertain. Our main argument is that knowledge cannot be viewed as a simple conceptualization of the world, but it has to represent some degree of interpretation. Such interpretation depends on the context of the entities involved in the process. In order to represent these subjective probabilities in our system we use the Dempster-Shafer theory of evidence [32], which provides a mechanism for modelling and reasoning uncertain information in a numerical way, particularly when

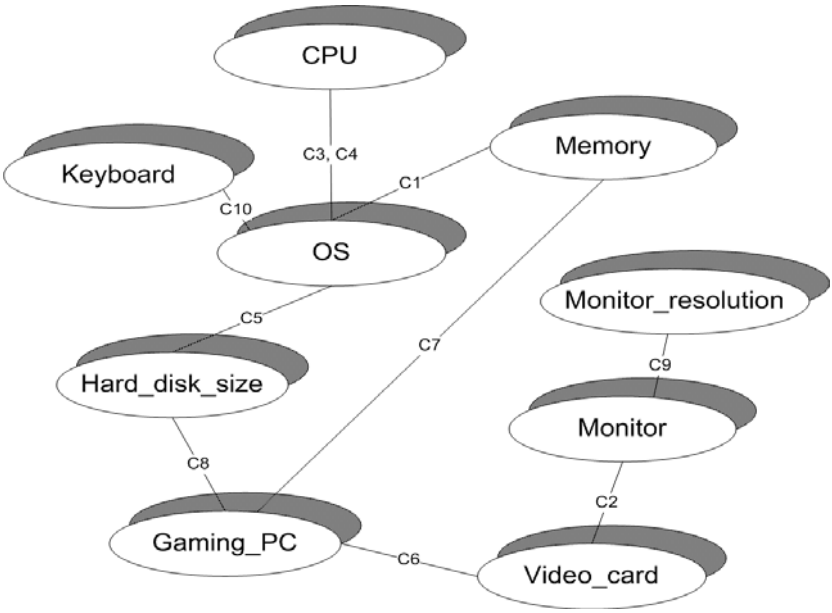


Fig. 3. A Constraint graph using variable Video_card

it is not possible to assign belief to a single element of a set of variables. Further our proposed solution involves consultation of background knowledge, assessment of similarities, resolving conflicts between the assessments and finally the selection of possible mappings i.e. items that are named differently but are the same in practice. As an example consider that we need to determine that the "Video_card" is equivalent to the "Graphics_card". In this case our hypothesis (H) is that these items are equivalent but we need to find evidences that support or contradict our initial hypothesis. In our case we create several hypothesises comparing each element of the constraint graph to each other. As an example consider that the following three hypothesises were selected from all available ones:

$$H_1(equivalent) = \{video\_card\} \Leftrightarrow \{graphics\_card\}$$
$$H_2(equivalent) = \{video\_card\} \Leftrightarrow \{mouse\}$$
$$H_n(equivalent) = \{video\_card\} \Leftrightarrow \{term_n\}$$

Further it is advisable that during the similarity assessment we use different similarity algorithms i.e. use different agents that are specialised in a particular similarity assessment. Since the hierarchy of the constraint graph cannot be exploited for similarity assessment the only way is to utilise the nodes in order to detect the mappings. As such consider that we use three agents using different string similarity measures. The steps to produce the mappings are as follows:

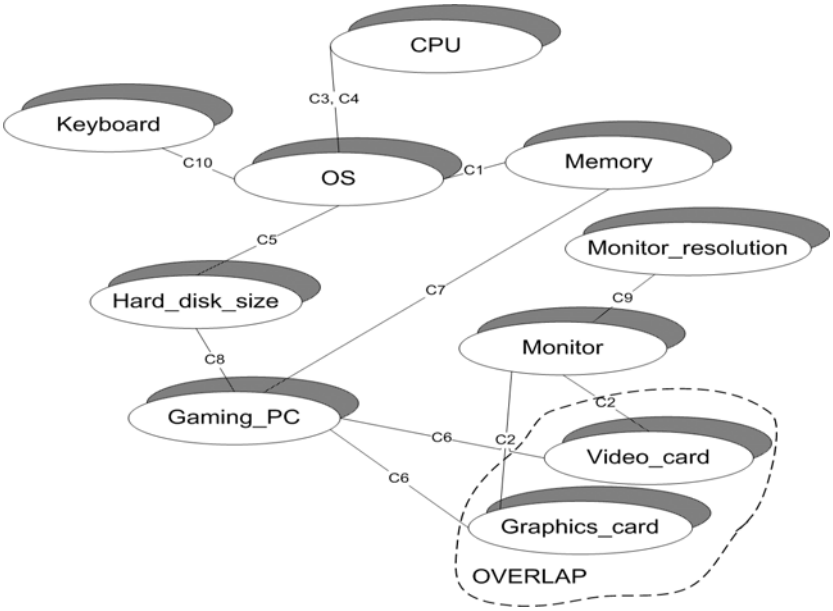


Fig. 4. A Constraint graph for the computer configuration problem with 9 variables

Step 1 consult background knowledge: In this step using general background knowledge e.g. WordNet we try to determine the meaning of the terms. Our case is specialised as the computer shop only sells electronics therefore other meanings e.g. art context of graphics can be excluded from the process. After consulting background knowledge we can extend our initial terms using sister terms and direct hypernyms with the following computer science related terms:

$$Video\_card = \{videodisplay, graphics, picture, graph\}$$
$$Graphics\_card = \{picture, movie, video, image, visual\ representation\}$$
$$Mouse = \{trackball, rotatableball, cursor\ control\ device\}$$

Step 2 similarity assessments: Using different string similarities e.g. Jaccard, Jaro-Winkler, Monge-Elkan we have found that

$$Agent1 : H_1(mapping) = 0.80; H_2(mapping) = 0.3$$
$$Agent2 : H_1(mapping) = 0.72; H_2(mapping) = 0.2$$
$$Agent3 : H_1(mapping) = 0.64; H_2(mapping) = 0.2$$

After the belief assessments we can establish that H1 is the preferred choice between the available hypothesises and that H2 does not contain contradictory beliefs. However H1 contains contradictions because Agent 2 belief does not support sufficiently that H1 can be selected. The different strategies for selecting

the contradicting belief is out of the scope of this paper but for our scenario we use the rule of thumb that in an ordered list of beliefs at least 2 agents should have the same belief otherwise there is a contradiction. In our framework all the numerical values represent the belief mass function that each agent can deduce from the similarity calculations. The represented beliefs are the interpretation of each agent and such they are subjective. Once the beliefs in similarities have been established agents need to select the hypothesis with the highest belief. In our example this corresponds to the H1 namely that the "Video_card" and "Graphics_ card" could be similar. Before the mapping is selected we need to verify that the original beliefs are not contradicting.

Step 3 verification and resolution of contradictions: It is important to point out that our proposed approach does not utilise thresholds for defining what is contradicting or not. e.g. if the difference is greater than 0.5 then there is a contradiction. Our solution makes use of comparisons between each agent's belief and eliminates the one that can be contradictory with the majority of the beliefs. The strategies for selecting, which agent should start evaluating trust is a complex issue and is out of the scope of this paper. However in our scenario we consider a basic rule that tries to establish similar beliefs of at least two agents. Therefore the beliefs in similarities need to be ordered and the agent whose belief function value is the smallest (smaller then the highest and greater than the smaller) will start to the trust evaluation process. In our example Agent 2 is in the position of detecting such contradiction as both Agent 1 and Agent 3 has different belief on the similarity. The question in this case is to trust Agent 1 and support that "Video_card" and "Graphics_card" is equivalent or trust Agent 3 whose belief is lower and probably discharge the mapping.

In order to resolve the contradiction we use the fuzzy voting model [5] because the different beliefs in similarity can be resolved if the mapping algorithm can produce an agreed solution, even though the individual opinions about the available alternatives may vary. We propose a solution for reaching this agreement by evaluating trust between established beliefs through voting, which is a general method of reconciling differences. Voting is a mechanism where the opinions from a set of votes are evaluated in order to select the alternatives that best represent the collective preferences. Unfortunately deriving binary trust like trustful or not trustful from the difference of belief functions is not so straightforward since the different voters express their opinion as subjective probability over the similarities.

For a particular mapping this always involves a certain degree of vagueness hence the threshold between the trust and distrust cannot be set definitely for all cases that can occur during the process. Additionally there is no clear transition between characterising a particular belief highly or less trustful. Therefore our argument is that the trust membership or belief difference values, which are expressed by different voters can be modelled properly by using fuzzy representation. Before each agent evaluates the trust in other agent's belief over the correctness of the mapping it calculates the difference between its own and the other agent's belief. Depending on the difference it can choose the available

trust levels e.g. if the difference in beliefs is 0.08 (belief of Agent 2 - Agent 3 and belief of Agent 3 - Agent 2) then the available trust level can be high and medium. We model these trust levels as fuzzy membership functions. In fuzzy logic the membership function (x) is defined on the universe of discourse U and represents a particular input value as a member of the fuzzy set i.e. $\mu(x)$ is a curve that defines how each point in the U is mapped to a membership value (or degree of membership) between 0 and 1. Our membership functions are as follows:

Definition 4. *Similarity is an input variable and is the result of some syntactic or semantic similarity measure between two terms/nodes in the ontology. These similarity measures can be obtained using a wide variety of standard techniques like Jaccard distance or node distance in source and target graphs that represent the ontology fragments. In terms of fuzzy representation we propose three values for the fuzzy membership value* $\chi(x) = \{low, average, high\}$.

Definition 5. *Belief is an input variable, which describes the amount of justified support to A that is the lower probability function of Dempster, which accounts for all evidence* E_k *that supports the given proposition A. Consequently the belief value is equivalent to the normalised sum of similarity values that is calculated based on the evidences that support the hypothesis. We propose two values for the fuzzy membership value* $\beta(x) = \{weak, strong\}$.

Definition 6. *Belief difference is an input variable, which represents the agents own belief compared to the other agents' belief over the correctness of a mapping in order to establish mappings between concepts and properties in the ontology. Therefore during conflict resolution we calculate the level of difference by comparing agent x belief to agents x-1, x+1 beliefs over the similarity. We apply three values for the fuzzy membership value* $\mu(x) = \{small, average, large\}$.

Definition 7. *Trust is the output variable and represent the level of trust we can assign to the combination of our input variables. The trust is therefore calculated by applying the fuzzy rules on the fuzzy input variables. We propose three values for the fuzzy membership value* $\tau(x) = \{low, medium, high\}$.

Once each input and output variables have been initialised we run the fuzzy system[2] that defuzzifies the result defined by our output variable i.e. trust. During fuzzy reasoning we have the linguistic output variables, which need to be translated into a crisp (i.e. real numbers, not fuzzy sets) value. The objective is to derive a single crisp numeric value that best represents the inferred fuzzy values of the linguistic output variable. Defuzzification is such inverse transformation, which maps the output from the fuzzy domain back into the crisp domain. In our ontology mapping system we have selected the Center-of-Area (C-o-A) defuzzification method. The C-o-A method is often referred to as the Center-of-Gravity method because it computes the centroid of the composite area representing the output fuzzy term. In our system the trust levels are proportional with the area

[2] http://jfuzzylogic.sourceforge.net/

of the membership functions therefore other defuzzification methods like Center-of-Maximum (C-o-M) or Mean-of-Maximum (M-o-M) does not correspond well to our requirements.

Consider the set of words $\{Low\_trust(L\_t), Medium\_trust(M\_t), High\_trust(H\_t)\}$ as labels of a linguistic variable trust with values in U=[0,1]. Given a set "m" of voters where each voter is asked to provide the subset of words from the finite set T(L), which are appropriate as labels for the value u. The membership value $\chi\mu(w)(u)$ is taking the proportion of voters who include u in their set of labels, which is represented by w. The main objective when resolving conflict is to have sufficient number of independent opinions that can be consolidated. To achieve our objective we need to introduce more opinions into the system i.e. we need to add the opinion of the other agents in order to vote for the best possible outcome. Therefore we assume for the purpose of our example that we have 10 voters (agents). Formally, let us define

$$V = \{A1, A2, A3, A4, A5, A6, A7, A8, A9, A10\} \quad (1)$$
$$T(L) = \{L_t, M_t, H_t\}$$

The number of voters can differ however assuming 10 voters can ensure that

1. The overlap between the membership functions can proportionally be distributed on the possible scale of the belief difference [0..1]
2. The work load of the voters does not slow the mapping process down

Let us start illustrating the previous ideas with a small example. By definition consider three linguistic output variables L representing trust levels and T(L) the set of linguistic values as $T(L) = \{Low\_trust, Medium\_trust, High\_trust\}$. The universe of discourse is U, which is defined as U=[0,1]. Then, we define the fuzzy sets per output variables $\{(Low\_trust), (Medium\_trust), (High\_trust)\}$ for the voters where each voter has different overlapping trapezoidal, triangular or Gauss membership functions as depicted in Figure 5.

The difference in the membership functions represented by different vertices depicted in Figure 5 ensures that voters can introduce different opinions as they pick the possible trust levels for the same difference in belief. The possible set of trust levels L=TRUST is defined by the Table 4. Note that in the table we use a short notation L_t stands for Low_trust, M_t stands for $Medium\_trust$ and H_t stands for $High\_trust$. Once the input fuzzy sets (membership functions) have been defined the system is ready to assess the output trust memberships for the input values. Both input and output variables are real numbers on the range between [0..1]. Based on the difference of beliefs, own belief and similarity of the different voters the system evaluates the scenario.

The evaluation includes the fuzzification which converts the crisp inputs to fuzzy sets, the inference mechanism which uses the fuzzy rules in the rule-base to produce fuzzy conclusions (e.g., the implied fuzzy sets), and the defuzzification block which converts these fuzzy conclusions into the crisp outputs. Therefore

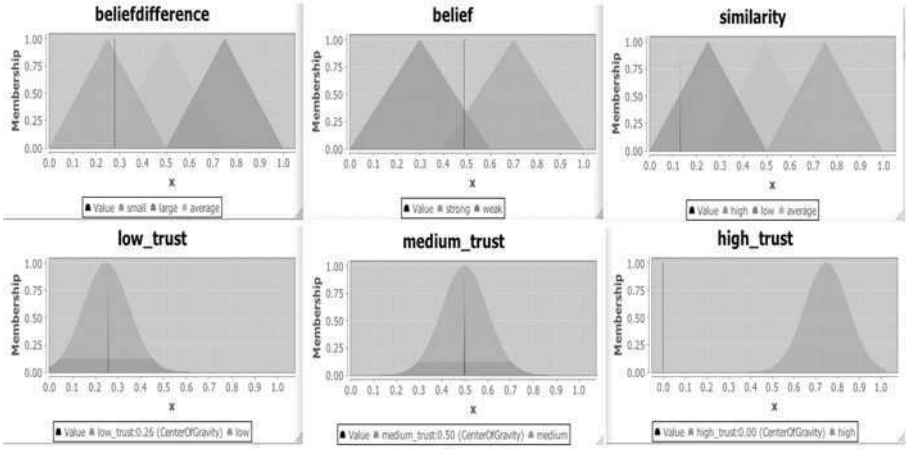


Fig. 5. Possible membership functions

each input (belief difference, belief and similarity) produces a possible defuzzified output (low, medium or high trust) for the possible output variables. Each defuzzified value can be interpreted as a possible trust level where the linguistic variable with the highest defuzzified value is retained in case more than one output variable is selected. As an example consider a case where the defuzzified output has resulted in the situation described in Table 4. Note that each voter has its own membership function where the level of overlap is different for each voter. Based on a concrete input voting agent nr 1 could map the defuzzified variables into high, medium and low trust whereas voting agent 10 to only low trust.

Table 4. Possible values for the voting

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
L_t	L_t	L_t	L_t	L_t	L_t	L_t	L_t	L_t	L_t
M_t	M_t	M_t	M_t	M_t	M_t				
H_t	H_t	H_t							

Note that behind each trust lever there is a real number, which represents the defuzzified value. These values are used to reduce the number of possible linguistic variables in order to obtain the vote for each voting agent. Each agent retains the linguistic variable that represents the highest value and is depicted in table 5.

Table 5. Voting

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
H_t	M_t	L_t	L_t	M_t	M_t	L_t	L_t	L_t	L_t

Taken as a function of x these probabilities form probability functions. They should therefore satisfy:

$$\sum_{w \in T(L)} Pr(L = w|x) = 1 \tag{2}$$

which gives a probability distribution on words:

$$\sum Pr(L = Low\_trust|x) = 0.6 \tag{3}$$

$$\sum Pr(L = Medium\_trust|x) = 0.3 \tag{4}$$

$$\sum Pr(L = High\_trust|x) = 0.1 \tag{5}$$

Therefore applying the appropriate input variables and the basic fuzzy rules the system will determine that as a result of voting and given the difference in belief x = 0.08 (belief of Agent2 - Agent 3 and belief of Agent 3 - Agent 2) the combination should not consider the belief of the third agent since based on its difference compared to another beliefs it turns out to be a distrustful assessment. The before mentioned process is then repeated as many times as many different beliefs we have for the similarity i.e. as many as different similarity measures exist in the ontology mapping system.

Step 4 belief combination: Once the conflicts have been resolved and the distrustful beliefs have been eliminated the individual beliefs can be combined to a more coherent belief. This belief combination is done using the Dempster's combination rule:

$$m_{ij}(A) = m_i \oplus m_j = \sum_{E_k E_{k'}} m_i(E_k) * m_j(E_{k'}) \tag{6}$$

where we have two individual belief mass functions $mi(Ek)$ and $mj(Ek')$ and i and j represent two different agents. After the belief combination the belief in H1 equals 0.79 and for H2 equals 0.25. As a consequence we can deduce that the "Video_card" and "Graphics_card" are the same component.

5 Knowledge Integration Framework

Ontologies offer interoperability and the possibility of a real integration of heterogenous sources. The vision of the Semantic Web predicts that existing resources i.e. databases, data on web pages will be described using ontologies.

These ontologies would either be created by individuals, organisations or as a result of converting existing thesaurus like WordNet. These resources will be user by software applications in order to determine the meaning of concepts, properties and to exchange these meanings in a certain context. Therefore in our Knowledge Integration framework we have included two ontologies containing the background knowledge for the Computer Configuration Problem. We assess similarities in the provided ontologies in order to find similar structures and terms. This background knowledge is used later at the level of the CSP solver so overlapping knowledge can be detected effectively using DSSim created background knowledge from domain ontologies.

The integration framework used in this work is depicted in Figure 6 where the flow of control between modules is shown. The picture shows the knowledge models used in the framework. In our particular case, we are using as knowledge models ontologies from PC online shops. These ontologies were built by ourselves and they are written in OWL the standard ontologies language. Then, we populate our ontologies using the values of two on-line shops. The suggested framework is generic as a first instance (for testing our ideas) the two models are OWL ontologies however, other formalisms could be used for building the models, for example, the formalisms which could be easily converted into the OWL ontology language.

We assume in our Knowledge Integration framework that each model has been built by different knowledge engineers and therefore, overlapping and contradictions of knowledge might occur. Our solution to the overlapping knowledge is performed by using DSSim. Although, for the purpose of this chapter, DSSim is presented as a black box as it has been described elsewhere [25] [26] [28] [27]. DSSim is an ontology alignment system based on Dempster Shafer and a fuzzy voting model. Besides DSSim uses background knowledge and WordNet to assess similarity between classes an properties in ontologies. The models are the input to our DSSim system which produces the mappings detected between the models. Currently, for the purpose of the experiment shown in the next section we are dealing only with two ontologies (i.e. two models). However, the framework can be extended to deal with multiple knowledge models. Preliminary results suggest that one problem addressed in our knowledge integration framework namely the detection of overlapping knowledge can be solved using the mappings obtained by DSSim.

The integrator module uses DSSim outputs and makes requests for approval of mappings to the knowledge engineer or human expert. The “mapping information” obtained by DSSim is passed to the Integrator module which then modifies the constraints graphs using the approval information provided by the knowledge engineer. Finally, the modified Constraint Satisfaction Problem (modified constraint graph) is passed to the choco solver which solves the Constraint Satisfaction Problem and returns results to the final user. In our research we used choco as it is a standard constraint solver which can be integrated with Java. Therefore, it fits very well with our overall knowledge integration framework.

The "Configuration System Interface" is the front-end to our users. Currently is not an elaborated interface as our priority was to test our ideas on the knowledge integration framework. Next section presents an experiment using two knowledge models on our case of study (a restricted version of a configuration problem).

5.1 Algorithms for Detecting and Correcting Overlappings

Two different algorithms play a key role in our framework as they deal with detection and correction of overlaps i.e. mappings and the resolution of the constraint satisfaction program (CSP). Our strategy of combining these algotithms is to create a clear split between the steps considering performance reasons. Knowledge integration on real domains is a challenging and complex task, involving two computationally expensive operations:

1. Mapping terms between different sources: In order to detect overlapping in the configurations we need to create mappings between all possible items in the inventory (all instances in both ontology 1 and ontology 2). This operation typically involve comparisons on a state space that is the cartesian product of the different inventory items.

   $$INV_n \times INV_m = \{(item_n, item_m) | item_n \in INV_n \; and \; item_m \in INV_m\} \quad (7)$$

   Consequently the possible comparisons increases as the number of inventory items increases therefore a real time comparison can easily become unfeasible. Further in real life scenarios the inventory cannot change between proposing two configurations to the user therefore the mapping can be and should be done only when the inventory in the sources are changing. As a result we create the mappings between the inventory items first and we run the mapping algorithm when necessary during the knowledge integration.
2. CSP search strategy and model solving: A key ingredient of any constraint solving approach is the appropriate construction of the search tree and the definition of the search algorithm. A common approach is by assigning variables to certain values however the size of the search tree is proportional to the number of variables and domain values we use in the model. Therefore finding a solution for a certain problem can easily become unfeasible in real time if the number of components for the configuration is increasing. Unfortunately contrary to the mapping generation the constraint satisfaction problem need to be resolved in the real time every time the user asks for a certain quote from the system. Therefore in our approach we need to put an upper limit on the time the algorithm can run in order to provide answer to the user' query.

The ontology mapping process that includes the fuzzy voting is described in Algorithm 1. The input of the algorithms are the similarity matrixes that contain various similarity measures. The output of this algorithm is the mapping file in the OAEI format. This mapping file describes, which items are equivalent

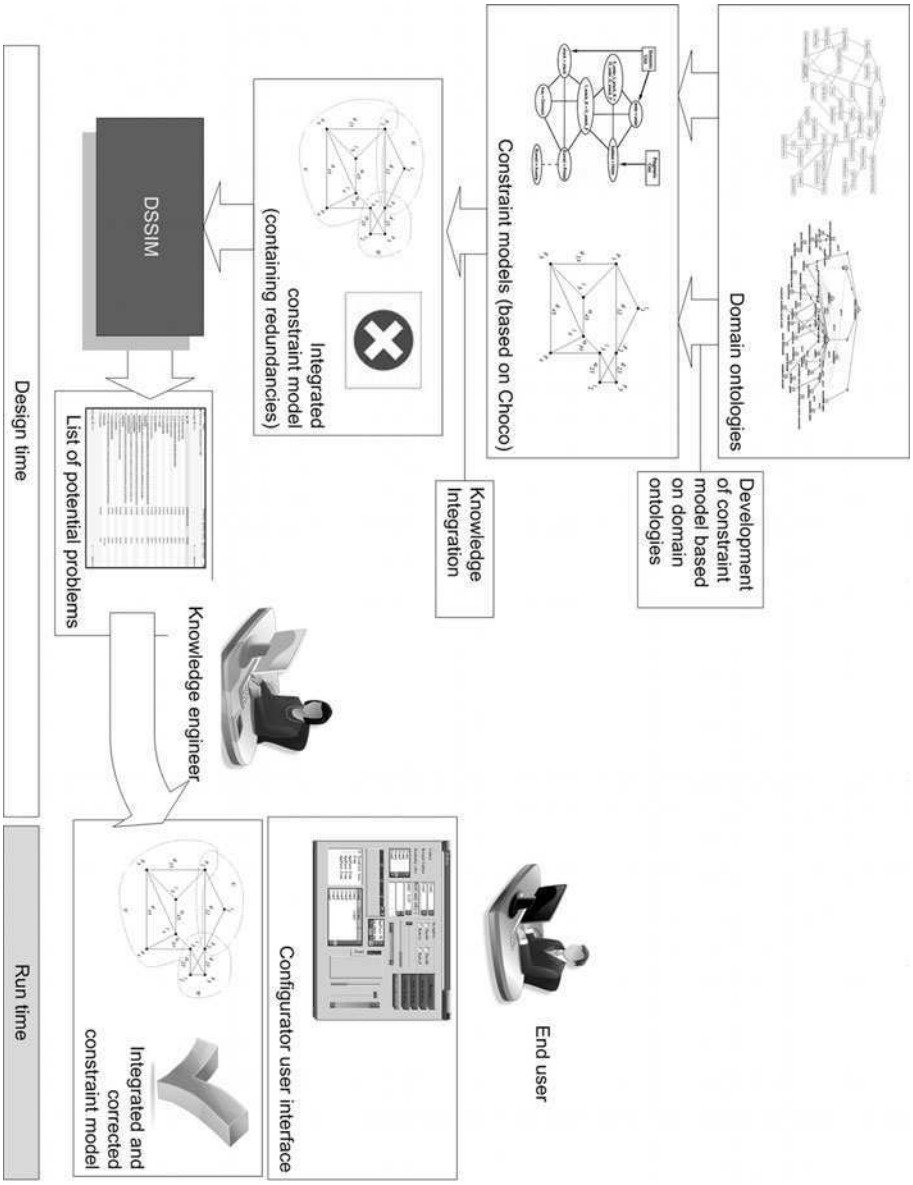


Fig. 6. Knowledge Integration Framework for a Configuration Problem

in the different inventories e.g. "Video_card and Graphics_card". As we have described before the algorithms iterates through the similarity matrixes and tries to establish and combine the Dempster belief functions using scenario and evidences (line 1-7). In case the evidences are contradictory a number of voters are created in order to determine, which belief can be trusted (line 8-14). Once the trusted beliefs have been selected the algorithm combines the beliefs and creates the mapping file based on these beliefs (line 16-17). This iterative process finishes once all items from inventory 1 have been compared with all the items in inventory 2. In our scenario these inventories contain all the instances in the ontologies.

```
   Input: Similarity belief matrixes S_{n×m} = {S_1, .., S_k}
   Output: Mapping candidates
 1 for i=1 to n do
 2     BeliefVectors BeliefVectors ← GetBeliefVectors(S[i, 1 − m]) ;
 3     Concepts ← GetBestBeliefs(BeliefVectors BeliefVectors) ;
 4     Scenario ← CreateScenario(Concepts) ;
 5     for j=1 to size(Concepts) do
 6         Scenario ← AddEvidences (Concepts) ;
 7     end
 8     if Evidences are contradictory then
 9         for count=1 to numberOf(Experts) do
10             Voters ← CreateVoters(10) ;
11             TrustValues ← VoteTrustMembership(Evidences) ;
12             ProbabilityDistribution ←
               CalculateTrustProbability(TrustValues) ;
13             Evidences ← SelectTrustedEvidences(ProbabilityDistribution)
               ;
14         end
15     end
16     Scenario ← CombineBeliefs(Evidences) ;
17     MappingList ← GetMappings(Scenario) ;
18 end
```

Algorithm 1. Creating mapping for component overlap detection

On the other side the configuration selection based on CSP solution is not an iterative process, however solving the problem itself can pose challenges in terms of computational complexity. The algorithm is described on Algorithm 2. The inputs of the algorithm are the inventory items in different sources, the mapping file and the requested items from the user if any.

The first step is to create a sample configuration using information from the requested items that has been specified by the user e.g. user wants Intel processors only. The sample configuration will contain items from both inventory 1 and 2 therefore it may contain similar items. These similar items are eliminated

from the configuration (line 2) using the mapping file that was generated by the ontology mapping algorithm. The next step is to create the CSP model that we wish to solve using the sample configuration (line 3). After we iterate through on each item in the configuration and transform it to a CSP variable together with its possible domain (line 4-6). Then we assign the constraints for the problem (line 7) and solve the CSP problem using our established model(line 8). Finally we can read out the suggested configuration from the model, which will be used as a quote to send back to the user.

Input: $Inventory_n, Inventory_m, Mappingfile, Requesteditems$
Output: Suggested configuration
1 SampleConfiguration ← `GetSampleConfiguration` (InventoryItems, RequestedItems) ;
2 SampleConfiguration ← `EliminateSimilarNodes` (Mappings) ;
3 CPMModel ← `CreateCPMModel` (SampleConfiguration) ;
4 **for** *i=1* **to** *NumOfComponents* **do**
5 | CPMVariable$_i$ = `CreateVariables` (`GetComponentDomain` (SuggestedConfiguration $_i$)) ;
6 **end**
7 Constraints ← `CreateConstraints` () ;
8 CPMModel ← `SolveCPMProblem` () ;
9 SuggestedConfiguration ← `GetSuggestedConfiguration` (CPMModel) ;

Algorithm 2. Resolving the configuration problem

6 Evaluation

In order to evaluate our approach we have carried out experiments (process is depicted in Figure 7) using two ontologies that were created from two on line PC (Personal Computers) store. Both ontologies contain categories and instances of items that are sold in the on-line shop. The main objective of our experiment was to evaluate how accurate our knowledge integration approach is. During the experiments we have generated 100 random configurations that simulates a customer choice and we have evaluated the correctness of the configurations after the two ontologies were mapped into each other.

The main idea of our experiment was to show the integration of our sample ontologies and then to use the integrated model for solving a computers configuration problem. The evaluation was performed in two phases. In the first phase we integrated two knowledge models (i.e. ontologies) from two online PC shops. The evaluation comprises to perform mapping between classes and properties of the two ontologies. In this task we had used our DSSim system [26] which is a mapping system based on Dempster-Shafer Theory described in detail in [25] [26] [27]. Although, the ontologies used in the experiment (presented in this chapter) are rather small our DSSim is equipped to deal with large ontologies [28]. In fact, DSSim has been tested with very large ontologies from the Ontology Alignment Evaluation Initiative (OAEI).

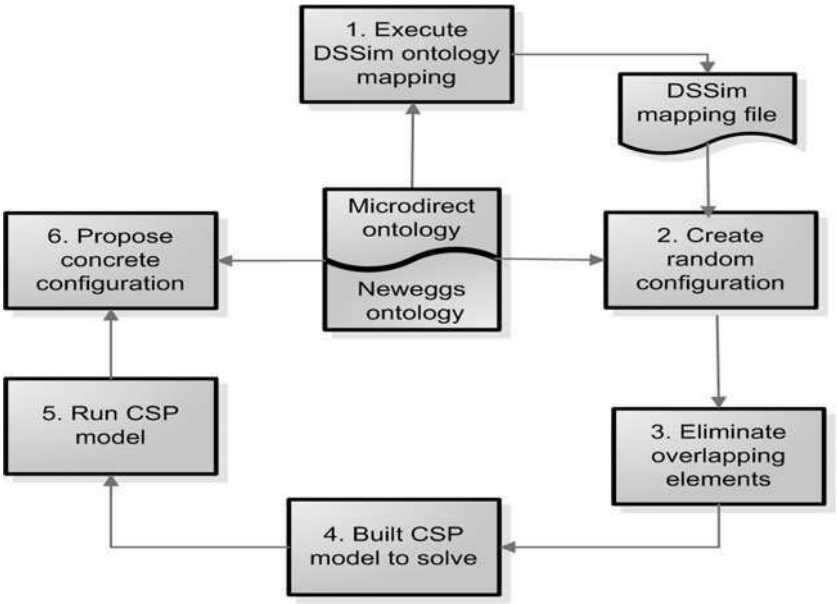


Fig. 7. Experimental process

The second part of the experiment is the solution of the CSP problem using the mappings generated in the first phase. To illustrate our solution we started by selecting just one PC configuration (basic configuration). Once the solution to the basic configuration of computers was obtained. Then, we carried out the experiments using the two remaining configurations namely medium and expensive configuration which were solved in a similar fashion. Therefore, in a first instance, we focussed our attention to the constraints associated to the basic configuration. These constraints are $C1...C10$ defined in section 3.2. Our solution used a constraint solver choco which is widely used in the CSP community. The notion basic, medium and expensive configurations have been represented with the number of components assuming that the more expensive a configuration is the more components the configuration will contain. In our experiment the basic configuration has 30, the medium has 50 and the expensive has 70 components.

We have carried out experiments based on the computer configuration problem described in the section 3.2. In order to make it as close to real situation as possible we have created 2 ontologies based on two online computer shops that sell a wide variety of PC Components and Accessories. One shop is the Micro Direct Ltd [3] from the UK and the second is Newegg [4] from the US. For the experiments we have created ontologies that contain only partial component list from both sites. The number of classes, properties and instances included in the ontologies are described on Table 6.

[3] http://www.microdirect.co.uk
[4] http://www.newegg.com/

Table 6. Example ontology complexities

	Microdirect.co.uk	Newegg.com
Classes	102	121
Properties	47	46
Individuals	197	242
Subclass axioms	96	118
Equivalent classes axioms	19	5

6.1 Mapping Quality

The first step of or experiment is to create a mapping file using DSSim in order to detect overlapping elements from the two ontologies. The idea behind our scenario and experiments is to integrate two data sources through ontology mapping. In practice this means that our solution should make it possible to create configurations from two different shops without physically integrating the databases. The mapping file generated by our algorithm contains 93 mappings. These mappings range from the very obvious to hidden correspondences between concepts and properties e.g. Memory - Memory, ATi_Graphics_Card Video_ card. In addition we have created manually a mapping file between the ontologies in order to compare with the one that is generated by the system. This evaluation was measured with recall and precision, which are useful measures that have a fixed range and meaningful from the mapping point of view.

There are two typical measures for assessing the performance:

Definition 8. *Precision: A measure of the usefulness of a hit list, where hit list is an ordered list of hits in decreasing order of relevance to the query.*

Definition 9. *Recall: A measure of the completeness of the hit list and shows how well the engine performs in finding relevant entities.*

Recall is 100% when every relevant entity is retrieved. However it is possible to achieve 100% by simply returning every entity in the collection for every query. Therefore, recall by itself is not a good measure of the quality of a search engine. Precision is a measure of how well the engine performs in not returning non relevant documents. Precision is 100% when every entity returned to the user is relevant to the query. There is no easy way to achieve 100% precision other than in the trivial case where no document is ever returned for any query. Both precision and recall has a fixed range: 0.0 to 1.0 (or 0% to 100%). A good mapping algorithm must have a high recall to be acceptable for most applications. The most important factor in building better mapping algorithms is to increase precision without worsening the recall.

Before we present our evaluation let us discuss what improvements one can expect considering the mapping precision or recall. Most people would expect that if the results can be doubled i.e. increased by 100% then this is a remarkable achievement. This might be the case for anything but ontology mapping.

In reality researchers are trying to push the limits of the existing matching algorithms and anything between 10% and 30% is considered a good improvement. The objective is always to make improvement in preferably both in precision and recall.

Table 7. Mapping quality

	Value
Precision	0.66
Recall	1.0

Based on the result (depicted on Table 7) we can conclude that the recall rate is 100% therefore all the possible mappings have been found by the system. However the precision is 66 %, which indicates that some additional mapping were found and they are incorrect. The precision rate is high and indeed the manual mapping has resulted in the mapping file that contain only the equivalence relationships e.g. CPU - CPU between items. Our algorithm also identified not equivalence relations e.g. Motherboards - Server_Motherboard and this decreases the precision of the system.

6.2 Configuration Quality

In the second experiment we create random configurations using components from both shops i.e. ontologies. For example we take the memory from Microdirect and select the Monitor from Neweggs. The number of components can range between 30 and 70 depending on the configuration type.

In step three using the mapping file created in step 1 we eliminate the overlapping components from the configuration. For example if Video_card was selected from ontology 1 and Graphics_card was also selected to the configuration we leave only one of them in the configuration.

During step four we take the available prices for each component in the configuration. In practice we take all instances of each component and add them as variables for the CSP problem. For example for the Video card we take Sapphire_Radeon_HD_5850_1GB or XFX_ATI_RADEON_4650. All these variables will feed into our CSP solver engine as textual variables.

In step 5 we run the CSP solver in order to get what are the process we can spend on each component in order to produce the suggested configuration. Given the fact that there is no guarantee that the CSP problem can be resolved in a timely manner we put a 10 second constraint on the choco solver in order to limit the available time for each experiment. In case the solver cannot find an optimal solution the random configuration will be returned.

In Step 6 we select the concrete components that fit into our maximum amount we can spend on each component.

The process from step 2 to 6 is repeated 100 times in order to obtain reasonable amount of data that can be analysed. We are interested in measuring how well our proposed approach can perform in order to integrate knowledge from different sources. We measure how often overlapping elements are removed from random configurations and how often overlapping items have to be evaluated from the random configurations.

The experimental results are depicted in Figure 8.

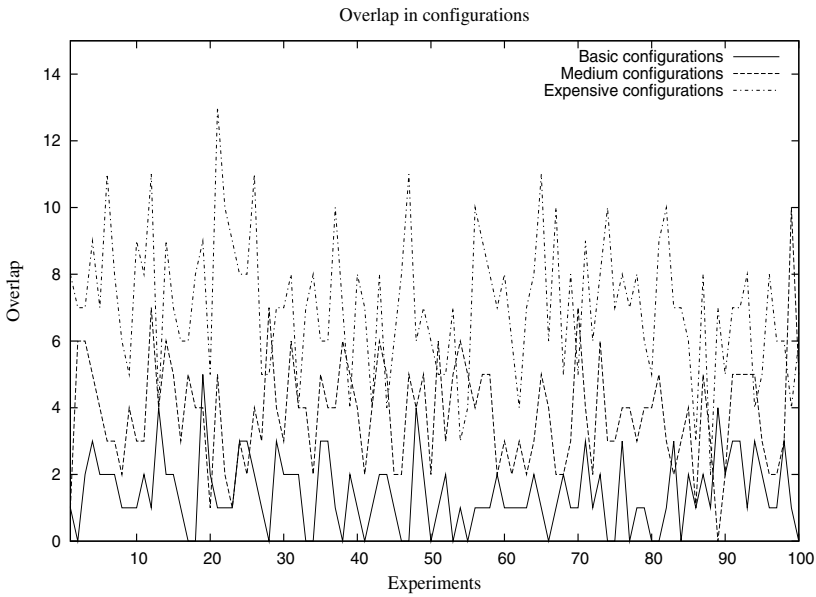


Fig. 8. Overlapping components

Table 8. Ovelapping component statistics

	Min overlap	Max overlap	Average overlap	No overlap
Basic configuration	0	5	1.45	21
Medium configuration	0	10	3.83	1
Expensive configuration	2	13	7	0

As depicted in Figure 8 and Table 8 the number of overlapping elements per configuration varies from 0-13. According to the experiments 79 % of the basic configuration, 99 % of the medium configuration 100 % for the expensive configuration represents cases where the overlapping elements have to be removed. In practice it means the knowledge integration occurs between 79-100 % percent of the cases. This is remarkable and in operational systems where users are involved this represents a considerable percentage. Based on our experiments we can establish that knowledge integration can improve the PC configuration precision in the majority of the cases.

Our experiments have showed that the result of constraint satisfaction problem for the PC configuration improves if the number of components for the configuration increases. This can be explained with the fact that with the more complex a configuration is the more overlapping in the CSP graph can occur. This is encouraging as our main objective is to establish a solution for the knowledge integration problem.

7 Conclusions

This chapter presented an approach to knowledge integration of several knowledge models. These knowledge models were created by different stakeholders. As a case of study to demonstrate our approach we introduced a restricted version of the computer configuration problem. Our case of study was modeled as a Constraint Satisfaction Problem and the constraints graphs were produced. The detection of overlapping pieces of knowledge and its solution was performed by means of DSSim, a agent-based system which uses similarity algorithms coupled with a fuzzy voting model.

The experiment shown was performed using two knowledge models and it was divided in two phases. The first phase was detection of overlapping knowledge and correction using our DSSim system. The second phase is the Constraint Satisfaction Problem using choco (the constraint solver). Our preliminary findings are encouraging and they are the baseline for assessing the usefulness of our Knowledge Integration. Of course, more work needs to be done in order to fulfill our expectations of a generic framework for Knowledge Integration. Future work comprise to carry out experiments using more knowledge models.

We have established a set of initial experiments and measures that combines ontology mapping and constraint satisfaction in a real word scenario. Our proposed experimental context for knowledge integration is the logical federation of two on-line PC stores, without physically creating a unified database. The federation is carried out only the overlapping elements of the two different data sources in order to being able to eliminate the number of equivalent components for the proposed configuration. Our ontologies used during the experiment contain only a fraction of the information that can be extracted from the two on line stores. Nevertheless our results are encouraging since even these relatively small ontologies produce 79-100 % of overlaps in the configurations. The more elements we include in our ontologies the higher overlapping components will emerge in these configurations. Therefore based on our current experiments we can conclude that the knowledge integration can occur in the majority of the cases and such approach can improve the overall situation of the system. However in the future we intend to investigate further what influences the number of overlapping elements that occur in random configurations. In terms of constraint satisfaction our experiments have showed that only the expensive configuration performs well as the medium and basic contains far too much configuration that do not match the users criteria. One explanation is the limited number of instances in the two ontologies. We expect that the more instances we will include

into our ontologies i.e. more PC components the better our constraint can be met for the basic and medium configuration. In general our experiments have showed that our approach is promising however it requires more experiments with larger ontologies in order to further assess the strengths and weaknesses of our approach.

References

1. Maedche, A., Motik, B., Silva, N., Volz, R.: MAFRA – A mApping fRAmework for distributed ontologies. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 235–250. Springer, Heidelberg (2002)
2. AnHai, D., Jayant, M., Pedro, D., Alon, H.: Learning to map between ontologies on the semantic web. In: WWW 2002: Proceedings of the 11th International Conference on World Wide Web, pp. 662–673. ACM, New York (2002)
3. AnHai, D., Pedro, D., Alon, H.: Reconciling schemas of disparate data sources: a machine-learning approach. SIGMOD Rec. 30(2), 509–520 (2001)
4. Ardissono, L., Felfernig, A., Felfernig, E., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schäfer, R., Zanker, M.: A framework for the development of personalized, distributed web-based configuration systems. AI Magazine 24, 93–108 (2003)
5. Baldwin, J.F.: Mass assignment fundamentals for computing with words. In: L. Ralescu, A. (ed.) IJCAI-WS 1997. LNCS, vol. 1566, pp. 22–44. Springer, Heidelberg (1999)
6. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. ACM Computing Surveys 18(4), 323–364 (1986)
7. Bayardo Jr., R.J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., Woelk, D.: Infosleuth: agent-based semantic integration of information in open and dynamic environments. In: SIGMOD 1997: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, pp. 195–206. ACM, New York (1997)
8. Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M.: The momis approach to information integration. In: ICEIS, pp. 194–198 (2001)
9. Doan, A., Domingos, P., Halevy, A.: Reconciling schemas of disparate data sources: A machine-learning approach. In: SIGMOD Conference, pp. 509–520 (2001)
10. Fauconnier, G.: Mental Spaces: Aspects of Meaning Construction in Natural Language. MIT Press, Cambridge (1985)
11. Fridman, N.N., Mark, M.: Prompt: Algorithm and tool for automated ontology merging and alignment. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 450–455. AAAI Press / The MIT Press (2000)
12. Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Sagiv, Y., Ullman, J., Vassalos, V., Widom, J.: The tsimmis approach to mediation: Data models and languages. Journal of Intelligent Information Systems 8, 117–132 (1997)
13. Gerd, S., Alexander, M.: Fca-merge: bottom-up merging of ontologies. In: IJCAI 2001: Proceedings of the 17th International Joint Conference on Artificial Intelligence, pp. 225–230. Morgan Kaufmann Publishers Inc., San Francisco (2001)

14. Ghidini, C., Giunchiglia, F.: Local models semantics, or contextual reasoning=locality+compatibility. Artificial Intelligence 127(2), 221–259 (2001)
15. Grant, R.M.: Toward a knowledge-based theory of the firm. Strategic Management Journal 17, 109–122 (1996)
16. Halevy, A., Rajaraman, A., Ordille, J.: Data integration: The teenage years. In: VLDB, pp. 9–16 (2006)
17. Hamdi, F., Niraula, B.S.N.B., Reynaud, C.: Taxomap in the oaei 2009 alignment contest. In: Proceedings of the 4th International Workshop on Ontology Matching (OM 2009). CEUR Workshop Proceedings, vol. 551, pp. 230–237 (2009)
18. Hung, H.F., Kao, H.P., Chu, Y.Y.: An empirical study on knowledge integration, technology innovation and experimental practice. Expert Systems with Applications 35(1-2), 177–186 (2008)
19. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Asmov: Results for oaei 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM 2009). CEUR Workshop Proceedings, vol. 551, pp. 152–159 (2009)
20. Johnson-Laird, P.: Mental Models. Harvard University Press, Cambridge (1983)
21. Levy, A.Y.: The information manifold approach to data integration. IEEE Intelligent Systems 13, 12–16 (1998)
22. Matteo, B., Paolo, B., Paolo, T.: Enabling distributed knowledge management: Managerial and technological implications. Tech. rep., Ingegneria e Scienza dell'Informazione, University of Trento (2002)
23. Murray, K.S.: Learning as Knowledge Integration. Ph.D. thesis, The university of Texas, Austin (1995)
24. Murray, K.S.: Ki: A tool for knowledge integration. In: AAAI/IAAI, vol. 1, pp. 835–842 (1996)
25. Nagy, M., Vargas-Vera, M., Motta, E.: Multi agent ontology mapping framework in the AQUA question answering system. In: Gelbukh, A., de Albornoz, Á., Terashima-Marín, H. (eds.) MICAI 2005. LNCS (LNAI), vol. 3789, pp. 70–79. Springer, Heidelberg (2005)
26. Nagy, M., Vargas-Vera, M., Motta, E.: Multi-agent ontology mapping with uncertainty on the semantic web. In: Proceedings of the 3rd IEEE International Conference on Intelligent Computer Communication and Processing, pp. 49–56 (2007)
27. Nagy, M., Vargas-Vera, M., Motta, E.: Managing conflicting beliefs with fuzzy trust on the semantic web. In: Gelbukh, A., Morales, E.F. (eds.) MICAI 2008. LNCS (LNAI), vol. 5317, pp. 827–837. Springer, Heidelberg (2008)
28. Nagy, M., Vargas-Vera, M., Stolarski, P.: Dssim results for oaei 2008. In: Proceedings of the 3rd International Workshop on Ontology Matching, pp. 147–159 (2008)
29. Rapanotti, L., Barroca, L., Vargas-Vera, M., Minocha, S.: deepjthink: a second life campus for part-time research students at a distance. Tech. rep., The Open University (2009)
30. Sanjay, M., Felix, F.: Towards a generic model of configuraton tasks. In: IJCAI 1989: Proceedings of the 11th International Joint Conference on Artificial Intelligence, pp. 1395–1401. Morgan Kaufmann Publishers Inc., San Francisco (1989)
31. Seddiqui, M. H., Aono, M.: Anchor-flood: Results for oaei 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM 2009). CEUR Workshop Proceedings, vol. 551, pp. 127–134 (2009)
32. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)

33. Wang, P., Xu, B.: Lily: Ontology alignment results for oaei 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM 2009). CEUR Workshop Proceedings, vol. 551, pp. 186–192 (2009)
34. Zhang, X., Zhong, Q., Shi, F., Li, J., Tang, J.: Rimom results for oaei 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM 2009). CEUR Workshop Proceedings, vol. 551, pp. 208–215 (2009)
35. Zygmunt, A., Koźlak, J., Siwik, L.: Agent-based environment for knowledge integration. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009. LNCS, vol. 5545, pp. 885–894. Springer, Heidelberg (2009)

Chapter 5
Simulation-Based Systems Design in Multi-actor Environments

Michele Fumarola, Mamadou D. Seck, and Alexander Verbraeck

Delft University of Technology, The Netherlands

Abstract. Much of the performance of a logistic system is determined by the quality of its design. A rich body of knowledge has been developed during the last decades that supports the product as well as the process of designing. Design methods for systems have initially defined the product by providing a framework to construct models to analyze the constructed artifact. This approach, which would later be coined the hard systems approach, would turn out to contain flaws in assumptions concerning of the existence of an (quasi-) optimal solution and by neglecting humans involved in the process. However, the techniques developed to support the approach, most notably simulation, have matured and are now commonly used to analyze designs. A major shift in approach took place to redeem for the perceived failures in systems design processes. This resulted in a tendency to focus on the diverging views of actors involved in the process, which was termed as the soft systems approaches. To profit from both sides, multimethodological approaches have been presented, assuming that a combination is feasible to take the best of both worlds. Participative simulation sessions have the potential to support the design processes: (1) in a multi-actor environment with diverging stakes, and (2) without ignoring the fact that human decision making relies on implicit knowledge that is insufficient and unreliable to evaluate decisions, thus requiring simulation for support. In this chapter, we present the rationale for requiring a multimethodological approach and discuss which aspects should be covered based on existing research literature.

Keywords: container terminal, system design method, simulation.

1 Introduction

Today's supply chains rely heavily on advanced logistics systems to transport goods. Efficiency is ever so important, leading to a preference for faster and more precise automated equipment. The technical challenges, high investments, and the societal and environmental impact depend upon multiple actors, or stakeholders, to lead the decision making process for designing these logistics systems. This imposes challenges that become increasingly difficult to address.

Logistics systems are typically designed and analyzed from a systems engineering perspective. Simulation, optimization methods, linear programming, and

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 107–127.
springerlink.com

other mathematical techniques are used to analyze and optimize logistics systems. One of the prominent examples of logistics systems that require this kind of methods, are container terminals [61][8]. The analysis and optimization is performed using a pre-defined utility function that is usually expressed in key performance indicators. The decision making process involves multiple actors which have varying stakes in the process. These stakes can often be contradictory, which can hinder the decision making process.

A case study that we performed at a large container terminal operator exposed the great complexity in the decision making process from both the system's as actor's perspective. De Bruijn and Herder [17] discuss how important it is to analyze both perspectives to achieve a complete analysis. The system's perspective of the design process of container terminals has been discussed often and thoroughly in literature: overviews are given by Steenken *et al.* [61], Stahlbock and Voss [60] and Vis and de Koster [66]. In Hu [32] and Derksen [19], whose work resulted from the case study, an analysis of the actors involved in the decision making process was carried out. The major identified actors are involved with

- innovation as a way to deploy novel equipment to increase productivity;
- business as to keep costs and revenues in balance;
- engineering as to design a technically feasible terminal;
- and, finally, environment and safety.

In previous research [24], we discussed the challenging design process in light of the complexity from both the system as the actor perspective. We concluded that the design process needs to be supported in order to face these challenges. In this chapter, we will explore existing literature and provide the constructs to support a design process.

1.1 Outline of the Chapter

Following the introduction on the relevance of our research, we will continue by exploring existing literature on the subject at hand. In Section 2, we will provide an extensive overview on the multiple discipline involved to design (logistics) systems. In Section 4 we will provide the constructs found in literature that are needed to support a design process from the system as well as the actor perspective. We will provide a discussion and conclusions in Section 5.

2 Designing Systems

In the early sixties of the previous century, Alexander [3] presented his seminal work that introduced the notion of methods in design studies. Up to that point, a design effort was considered a craft that relied solely on intuition and skills. Alexander postulated that design methods could aid the designer in fitting the form (the design artifact) to its environment: "every design problem begins with an effort to achieve fitness between two entities: the form in question and its context". To fit the form to its environment, formal methods and techniques should be required for both design products and design processes. Although the

method he presented turned out to be unsuccessful for numerous reasons, the prior misconception of rejecting formal procedures in design was abandoned. This led to an abundance of novel design methods from 1962 onwards [16].

During the same period as the design method movement, a modern understanding of systems thinking gained traction in science and engineering. Holistic thinking was put next to reductionism, which dominated science for a long period of time. Von Bertalanffy's general systems theory [68] was readily translated to new fields like operations research and systems engineering [1]. Hall [27] clarified the distinction between the two by pointing out that operations research is concerned with optimizing existing systems, and systems engineering's concern is focused on the design of new systems. Jenkins [36] sharpened the difference by suggesting that system engineering looks at the total system whereas operations research tinkers at the level of the more mechanical sub-systems. It is important to note that the recognition of systems engineering as focused on design binds system's thinking to Alexander's idea on formal procedures for design studies.

At the crossroad of system's thinking and design lies Simon's [58] contribution. He observed that a design process must follow a specific path of first structuring the problem, followed by a formulation of alternative solutions based upon selected criteria and finally by a selection of the best alternative. Human's bounded rationality limits the designer's capabilities in exploring the solution space, and thus precludes from finding an optimal solution. This limit is due to various factors such as incomplete information, cognitive limitation of the designer and time pressure. The designer cannot and should not aim for optimizing while designing, but should compromise between satisfying and optimizing, which Simon denotes as satisficing. Based on these premises, future system engineers would go about accepting the paradigm as follows: : "There is a desired state, S(1), and a present state S(0), and alternative ways of getting from S(0) to S(1). 'Problem solving,' according to this view, consists of defining S(1) and S(0) and selecting the best means of reducing the difference between them" [11]. More specifically, the idea ruled supreme that the problem task to tackle would be about selecting the efficient means.

Design studies are carried out in different fields, most notably architecture and engineering. Design is considered the essence of engineering [69]. Hubka [34][49] suggested to design technical artifacts as systems that are connected to their environment by means of input and outputs. The system can be divided into subsystems taking into consideration their boundaries. Hubka argued this would be fundamental to define appropriate systems at any stage of abstraction, analysis or classification. Pahl and Beitz [49] pinpoint that from this notion onwards, it was a short steps towards using system's theory in design processes and specify that "systems approach reflects the general appreciation that complex problems are best tackled in fixed steps, each involving analysis and synthesis". Similarly, van Gigch [64] discusses the systems approach as a methodology of design. In his work, he presents modeling as the fundamental aspect of the system design process. Modeling implies that "the modeler abstracts properties from things in order to obtain a representation of the physical world". The abstraction process

plays an important role in design as designers go through it to refine images of reality through different levels of conceptualization. The importance of models and emphasis on abstraction is further given by Hoover and Renderle [31]. They discuss that abstract models can be defined at different stages in the design process to test design decisions and to provide a framework for making design refinements.

In engineering design, the aforementioned notions on design space exploration, abstraction and modeling have found considerable impact in methods and support systems. Engineering design denotes a "systematic, intelligent process in which designers generate, evaluate, and specify concepts for devices, systems, or processes whose form and function achieve clients' objectives or users' needs while satisfying a specified set of constraints" [22]. Choosing among design alternatives is a common underlying concept for many methods. Hazelrigg [28][29] goes as far as framing a truly rational process to produce the best possible result using 'a mathematics for design'. Many others have proposed constructs to structure and choose among alternatives: such constructs include trees [7], matrices [20], rankings, and charts [48]. Methods focusing on modeling on different level of abstraction also exist, most prominently in the work presented by Paredis *et al.*. [50].

System's thinking as originally understood in operations research and systems engineering did not fulfill its promises. This induced Ackoff [2] to call for a new paradigm that would break away from the ever-increasing "mathematization" of the field. Operations research and systems engineering did not pay attention on the actors involved in the decision making process. Particularly with regards to management problems, this was recognized as an important shortcoming. Mingers [45] discusses that traditional systems engineering designs systems starting from the purpose of the envisioned systems and working backwards with mathematical techniques to find ways to achieve their objectives. This is based on the flawed assumption that the objectives are clearly stated at the beginning of the design process.

An important step forwards was taken with the introduction of the soft systems methodology by Checkland [13]. Acknowledging changing requirements, different actors with different stakes and opinions, and the importance of a learning process, proved to be fundamental to develop the soft systems methodology. The methodology stood the test of time and, after some revisions, is still successfully used today [14][15]. The so called hard systems methods in operations research and systems engineering had to face competition with a number of new soft systems methods. A taxonomy for system's approaches, called the system of systems methodology, presented by Jackson [35], was developed to help select a suitable system's approach for a given problem task. Subsequently, Mingers [43] argued that the whole set of methods could be classified according to ontology, epistemology and axiology, that is what they model, how they model and why the model. According to this classification, the hard systems method lays the importance on the artifact. Conversely, soft systems methods tend to focus on the users, whether they can understand and discuss the problem situation.

Because of the different foci of both types of methods, the question arises whether multimethodology can exist: why choose one or the other when we can exploit both to best tackle a problem (situation)?

3 Systems Approaches

Robinson [55] talks about a continuum between soft and hard instead of bipolar extremes. This is motivated by novel interactive simulation environments developed in the nineties that can deal with uncertainty (changing requirement and lack of knowledge about the system) by allowing incomplete or estimated data. These environments can lead to discussions, which can be framed as a soft systems approach. Soft system methods emerged due to unsatisfactory applications of hard system methods to wicked or ill defined problems. This led to an abandonment of formal methods developed with the hard systems mentality and favored the loose frameworks known as soft systems methods. However, this choice is unfortunate, as both types of methods have their disadvantages that can be overcome by seeking an appropriate combination.

A tendency towards combining hard and soft systems methods is reflected in the results of surveys conducted among practitioners and in the amount of papers reported using a multimethodological approach that are published in prominent journals of the field. A major survey reported by Munro and Mingers [46], show that in situations where a combination of hard and soft systems is considered useful there is a slight preference towards discrete event simulation. Unfortunately, no clear reasons are given for this preference. To understand what justifies a successful implementation of discrete event simulation with a soft systems method, we have to rely on other case studies.

Mingers and Rosenhead [44] report case studies where a multimethodology was used in problematic situations. Among these case studies, several instances of simulation studies were conducted using a soft systems approach, more specifically studies by Lehaney and Paul [42], Lehaney and Hlupic [40], and Bennett and Worthington [10]. In these cases, soft systems methodology is used to support the model building phase of the simulation studies. This participative action brings the model closer to the problem owner and allows discussion on the model to explore the problem. A brief list of recent contributions that explore this possibility further show that this is a viable approach:

- Kotiadis [38] discusses how soft systems methodology can be used to determine the conceptual model's most important component, the simulation study objectives. The case study was conducted to improve a health care system.
- Lehaney *et al.* [41] present a case study that was initially meant as a conventional resource allocation simulation study, but that ended up as a simulation based discussion to reveal issues of misunderstanding within the studied organization and poor communication that led to misallocation of resources.

- den Hengst et al. [18] Explore how soft OR principles can be used for collaborative simulation. They conducted a case study in the Dutch airline industry and concluded that the combination is promising, but more research is needed to tackle a number of issues encountered in their research.
- Pidd [51] advocates the use of 'soft' approaches to make sure analysts tackle the right problem in a situation study. He provides a general guideline on how such a study can be carried out.
- Baldwin [6] states that classical simulation studies cannot be conducted in health care or other fast-changing businesses. According to the authors, this is due to the vague problem formulation phase present in simulation studies. They argue this phase is crucial and present their approach that puts more attention on defining the problem.
- Robinson [53] discusses how discrete event simulation can facilitate a discussion among stakeholders to identify problems in a user support helpline.

Although this is most probably not a complete list of studies showing this type of approach, we can treat it as a list of prominent examples from which we can draw conclusions. These studies show that in designing socio-technical systems, multimethodology is used for in framing the problem, learning about the problem and finding viable solutions. In each study, strong points are identified in both types of approaches, and exploited in the overarching approach. In the next sections, we will analyze both the hard side and the soft side of these studies, to identify their weak and strong points.

3.1 Systems Simulation in Design

Many definitions of systems exist, which generally include the idea of parts interacting with one another to form a coherent whole realizing a certain purpose. In a mathematical sense, a system has been defined as a set of variables and a set of relations between them. This mathematical object is often used as a model of a system in nature or as a model of an envisioned system to be engineered. An evaluation of the natural or envisioned system's performance can be done through the computational analysis of this model.

As we saw in the previous sections, design is a goal seeking activity which tries to reduce a gap between a current state and a desired state with the help of an engineered artifact. In this activity, system models have been used to evaluate whether an alternative solution takes us close enough to the desired state. The complexity of the design problem generally forbids the use of optimization techniques alone. Thus, the goal seeking behavior is assumed by the designer, and not by the model itself, which is neutral and structurally static. The computational model - used as a replacement of reality- is altered until its output variables give satisfactory results.

In the classical systems engineering lifecycle, the modeling and simulation methodology is used for analysis. Other tools, such as CAD, are used for design. Modeling and simulation is only used to evaluate the design choices that have

been reached using other tools. This is because designers are not generally comfortable in modeling using simulation languages which still require specialized knowledge. Furthermore, model building is generally a costly and time consuming activity which is externalized to analysts or consultants. Integrating design and evaluation activities could improve the quality of the designed artifacts and make the design process smoother.

To achieve this goal of facilitating modeling for a better integration in the design cycle, component based approaches look promising. Modular system frameworks are key enablers. The discrete event systems specification [70] and similar formalisms are particularly well adapted to help achieving that goal. Well tested simulation model components with well defined interfaces can be made available to the designer. The design activity then means to couple the different components in order to achieve a certain function. With the predefined components, the user does not have to worry about the underlying complexities. The focus can be completely put on the design itself.

Computational models and components are mathematical objects, made of variables and operators. They are obtained after a process of abstraction which gets rid of most of the complexity in the real system. The modeler - to make the task feasible - chooses a certain theoretical or pragmatic perspective, draws a boundary as tight as possible around the system of interest, selects a set of relevant variables, and chooses to exclude other variables. After this process of abstraction, the model reflects the system of interest, but it also strongly reflects the intentions and preconceived ideas of the modeler. If this model has to be used in a multi-actor design activity, all stakeholders are implicitly asked to adhere to the paradigm of the conceptual designer, otherwise, all questions and ideas they may have based on the model can be out of scope or meaningless.

Using simulation in design also raises the question of model validity. The system being designed does not exist in real life, or at least not in its desired state. Replicative validation techniques, which are based on a statistical comparison between the model and the real system, are not applicable. To be of any use, the model used in a design activity should at least have predictive validity. To support fruitful discussion on the real envisioned system, the model should be structurally valid as well. Of course, the fact that the components are valid does not guarantee that the aggregated model is equally valid.

A final note on simulation based design concerns the static structure of most simulation languages. A framework for simulation based designed should allow models to see their structure altered dynamically during runtime. Formalisms with such capabilities have been introduced recently [9][33].

3.2 Soft Systems Methodology

Checkland's soft systems methodology (SSM) plays a prominent role in multimethodological approaches. The approaches presented in each study at the beginning of this section differ in how much they adhere to the methodology: some prefer to call their approach 'softer' whereas others explicitly state using SSM. We will briefly present SSM.

SSM differs from hard systems approaches in the way it perceives systems. Whereas hard systems approaches provide an ontological perspective on systems (Which entities are present? What are their relationships?), soft systems take an epistemological stance and discuss systems as a human's view on reality, hence a human construct used for understanding. In contrast with hard systems thinking, SSM does not focus on the solution, rather on a learning process actors go through while dealing with the problem situation. Hirschheim et al. [30] describe SSM as "a framework which does not force or lead the systems analysts to a particular 'solution', rather to an understanding". Throughout the many years of development, SSM has matured and has gone through many versions: the most widely used version is known for its seven stages.

The Seven Stage Soft Systems Methodology

The seven stages of classic SSM are about (1) defining the problem, (2) expressing the problem, (3) finding the root definitions of relevant systems, (4) developing conceptual models, (5) compare the conceptual model with reality, (6) define interventions, and (7) undertake action to improve the situation.

During the first stage, the problem situation is assessed. By collecting all sorts of data, one tries to gather a broad set of information available on the problem situation. No restrictions are given as to how much information is needed: this phase's sole purpose is to explore. In the next phase, expressing the problem, the information is structured to achieve a coherent expressive picture of the situation. The result of the second phase is composed of rich pictures, named to pinpoint the need of expressing the problem situation in all its richness. The rich pictures, which are preferably literally pictures (i.e. drawings or diagrams), could take into consideration structures, processes, climate, people issues expressed by people, and conflicts.

The third stage is about finding the so called root definitions: the perspectives or motivations of each actor in the rich picture. This phase commences by exploring the different perspectives of actors, in a rather unstructured form. After all perspectives have been identified, a structured analysis is carried out on the key perspective using the CATWOE model development process. CATWOE stands for customers, actors, transformation, welthanschauung (worldview), owner, and environment.

In the fourth stage, a conceptual model is constructed using the root definitions. The conceptual models would ideally be diagrams that demonstrate how each actor envisions a system that can fulfill the root definitions. The fifth stage is to compare the conceptual models with reality, to find incongruence but also to understand how the real world can be improved to meet the conceptual models. How these improvements can be achieved, is explored in the sixth phase by developing specific ideas. The last stage, contains the action to actually carry out these improvements in the real world.

It is important to note that all these stages do not serve as a strict framework to follow but are open to interpretations, variations, and iterations. The process

serves as base to achieve understanding among actors and identify objectives. The flexibility offered by this process, results in different adaptations that fit the best to the given problem, something we will discuss in the following section.

Simulation Studies from a Soft Systems Perspective

Pidd [51] explains why and how SSM can be useful in a simulation study. Modeling and simulation revolves often around implementing code, whether it is in general purpose languages such as Java or in simulation environments such as Arena (where the logic is constructed using building blocks). It is however important that a conceptual model is build first, to avoid implementing a simulation model that does not fully address the problem or that shows discrepancies with reality. The conceptual model that is developed in the fourth stage of SSM is ideal for developing a simulation model: it covers the different perspectives of the actors and, ideally, it went through much iteration.

Kotiadis [38] follows a similar approach as Pidd and uses the conceptual model gathered from SSM sessions. The focus lies mainly on the simulation study objectives that can be determined based on the definitions of efficacy, efficiency, and effectiveness. This extension of SSM [12] is used to specify the Performance Measurement Model (PMM): this model is constructed similarly to the conceptual model of the classic SSM. Using the PMM, one can go about finding out performance criteria, breaking the performance criteria into specific monitoring activities, decide what action might be taken based on these activities, and decide which activities might be evaluated in a simulation model.

Baldwin et al. [6] argue that Lehaney and Paul's [42] approach, and later also used by Lehaney et al. [41], is merely a first step towards combining SSM with simulation. Lehaney and Paul use simulation in the fourth phase of SSM to speed up the process by using quick-and-dirty simulation modeling. Modeling and simulation, in their view, can be used to enhance understanding and interpersonal communication of the stakeholders. However, contrarily to what is stated in other studies, the modeling effort should be present to understand the problem and should therefore not happen after the problem has been identified. Modeling becomes a way of communicating between stakeholders and the stakeholders should be involved into the modeling effort from the very beginning. They underline the need for an iterative process where stakeholders provide requirements for the modeling effort and the model produces new information for the stakeholder, something that is shown in Figure 1.

A looser approach to soft methods is taken by Robinson [53] who states that his "study involved a facilitated discussion around a simulation model about possible improvements to a problem situation". Instead of following a strict SSM process, he prefers to adapt the classic simulation process such as the one presented by Law and Kelton [39]. The methodology has different steps: (1) defining the objectives, (2) conceptual modeling, (3) model development, (4) verification, (5) validation, and (6) facilitation. In contrast to classic simulation

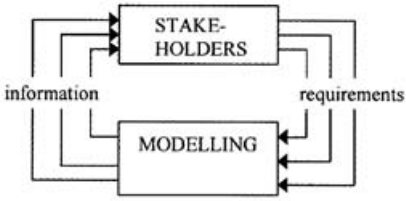


Fig. 1. The iterative modeling process presented by Balwin et al. [6]

studies, the data used for this study was neither complete nor reliable. Because of this, a facilitated discussion was organized instead of performing experiments with the simulation model. The simulation model was not used "as a tool that could accurately evaluate alternative options, but rather as a focus of debate, a means for learning about the problem situation and for reaching an agreement to act". This multimethodological approach is presented in a diagram, shown in Figure 2, that highlights the different aspects that can be considered "hard" and the one that can be considered "soft".

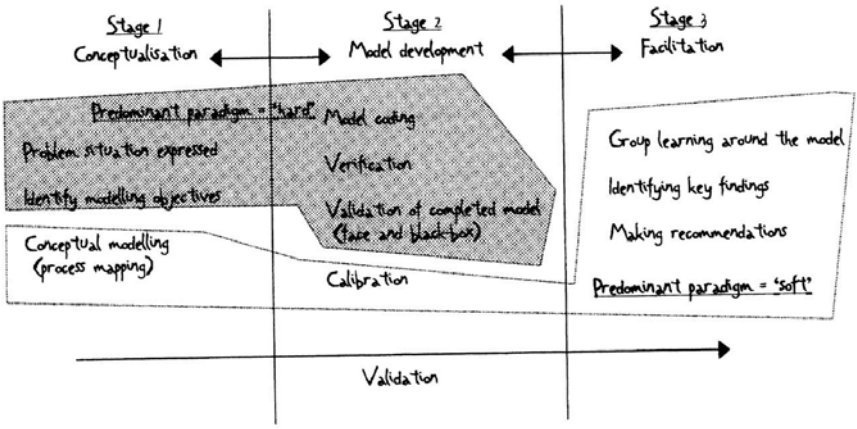


Fig. 2. The multimethodological approach presented by Robinson [53]

Den Hengst et al. [18] follow Robinson's approach in using simulation models to facilitate a discussion. Their approach follows five steps: (1) conceptualize problem, (2) create and validate empirical model, (3) construct alternative models and conduct experiments, (4) choose most preferred solution, and (5) implement solution. They discuss a case study that provides some valuable insights in the use of simulation models for this kind of interactive purposes. They present the following problems or difficulties:

- Building the simulation models required a lot of time and expertise: 'building the simulation model both conceptually and empirically took an average of two working days per week over a period of five months for two modellers who had significant simulation modelling experience' [18].
- Due to the complex code to build the simulation model, verification and validation was extremely difficult. In retrospect, they argue this could have been avoided because the validation did not add anything useful to the process.
- Lack of knowledge in simulation modeling resulted in a hard to accept model. The chosen simplifications and animation did not provide enough trust in the simulation model.
- Running the experiments took a long period of time. The chosen simulation environment required several hours to run full simulation experiments, which was unacceptable in an interactive session.

Discussion

It is clear that using simulation models as a base for discussion is useful and provides fruitful results. Aughenbaugh and Paredis [4] provide a very thorough and to-the-point explanation as to what simulation can bring to design and decision making:

> Without modeling and simulation, design relies on implicit knowledge. Implicit knowledge is unreliable in that designers do not know the assumptions and uncertainty in the knowledge explicitly. When decisions are coupled and require input from several experts, there is no way to make tradeoffs using only implicit knowledge about uncertainties.

Whereas traditional simulation studies focus on finding the optimal solution, soft systems approaches aim at achieving shared understanding in a group to support decision making. This is achieved by applying an iterative process that continues until all actors are satisfied with the model. However, as the studies have shown, using the simulation models does not go without any hurdles. The presence of various actors with diverging worldviews provides challenges to the simulation environment.

Robinson [54] identifies three modes of simulation practice: (1) simulation as software engineering, (2) simulation as a process of organizational change, and (3) simulation as facilitation. The predominant part of simulation practice takes place in the first mode (the construction of large and complex simulation models) whereas there is little to be found in the third mode. This implies that most simulation environments are constructed following a hard systems paradigm. According to Rosenhead [56] this means that

- problem formulation is in terms of a single objective;
- there are overwhelming data demands;
- people are treated as passive objects;
- there is a single decision-maker with abstract objectives from which concrete actions can be deduced;

– and there is an attempt to abolish future uncertainty, pre-taking future decisions.

This paradigm is implemented in modern simulation environments where users create one single detailed model per project (instead of various models for diverging opinions), users need to input a large amount of data, and there is a single optimal solution. If we were to use these simulation environments in a soft systems approach, a couple of issues would arise. A major issue is the construction the simulation model: simulation models take considerable time and expertise to develop. Although modern simulation packages relief developers from a lot of work, they are still not appropriate for interactive modeling sessions. Once build, the simulation environment requires some time to run full simulation experiments. Finally, the focus on a single model would restrict the exploration of the solution space.

4 Designing a Multimethodological Approach

In the previous section, we have introduced relevant literature on design methods from a systems engineering perspective. This led us to identify the main constructs needed for a design method, but also various shortcomings of the discussed approaches. The combination of these construct would entail a multimethodological approach. The constructs which will be discussed throughout this section are:

– Modularity and component based simulation: as proposed by den Hengst *et al.* [18], simulation models require a lot of effort and time to develop. Simulation models that are constructed of pre-defined components can be malleable to support an iterative design process, without requiring an intervention by simulation experts.
– Different levels of abstraction: in conceptually challenging domains such as engineering design, abstraction becomes a powerful tool to cope with large amount of data and complexity.
– Structing alternatives: as fundamental part of design methods, the definition and evaluation of alternatives
– Participatory design: a engineering design process is seldom performed by a actor or even a single person. Many actors are involved and all of them try to achieve their own objectives. A design process should support a certain convergence of interests of all actors.

In the following sections, we will discuss how these constructs can be implemented in a design method. We will formulate them in terms of our application domain, container terminals, to come to an instantiation of the design method.

4.1 Component Based Modeling

Den Hengst *et al.* argued that simulation models take much time and effort to construct. Pre-defined simulation components let users build new simulation

models by focusing on the structure without requiring knowledge about the inner behavior of the components. Another important feature of component-based modeling is modularity, which can be facilitated by port-based communication: such an approach has been adopted by Paredis *et al.* [50]. Once this approach is adopted, reusability of components becomes highly enhanced.

The notion of components has its origins in software engineering, which by now has become common practice for software development [59]. Component-based simulation has been adopted recently (compared to pure software engineering applications) but has quickly become a popular research topic [47]. A comprehensive overview on components in modeling and simulation is given by Verbraeck [65] who advocates the construction of a library of *easy to use* components to quickly create alternatives throughout a design process. Indeed, components provide the major advantage of being self-contained, reusable, replaceable and customizable. The use of simulation model components for design is also strongly advised in various other research for the same reasons [5][37][52][62].

In Section 3.1 we discussed the DEVS formalism that supports hierarchy, modularity, and composability. These are the exact features to be exploited to support simulation-based design [33][72][73]. To model a container terminal, an ontology can be determined to identify every single component that is needed to design a terminal. This would mainly boil down to identifying equipment present at a terminal and, as we will see in the following section, identifying the components for higher abstraction levels. Each component is modeled in DEVS and the ontology is presented in Figure 3 which is based on the System Entity Structure formalization [71].

4.2 Different Levels of Abstraction

Abstraction is powerful mechanism and is required to deal with uncertainty, problem complexity, and cognitive limitations of designers [26][31][63]. A typical design process starts with an incomplete picture of the design problem, which makes is not feasible to directly start with a detailed solution. From a cognitive perspective, Visser [67] notes that designing is an iterative process of generating designs, transform them and evaluate them. The difference between the first and last artifact is a question of degree of specification, completeness, and abstraction. A similar view is proposed by Goel [26], who writes "Design, at some very abstract level, is the process of transforming one set of representations (the design brief) into another set of representations (the constract documents)". Yet another similar formulation is put forward by Ullman *et al.* [63] as "a design problem typically begins with a set of functional constraints expressed very abstractly; during the design process, the level of abstraction of the design state is progressively reduced until is detailed enough to be manufactured".

During a design process, models can be constructed that are specific to predefined abstraction levels. Ullman *et al.* [63] do this by selecting three levels of abstractions: 'abstract', 'intermediate', and 'concrete'. For each level, they define a conceptual model. Similarly, Schimdt and Cagan [57] define an abstraction grammar which makes it possible to model components at different abstraction

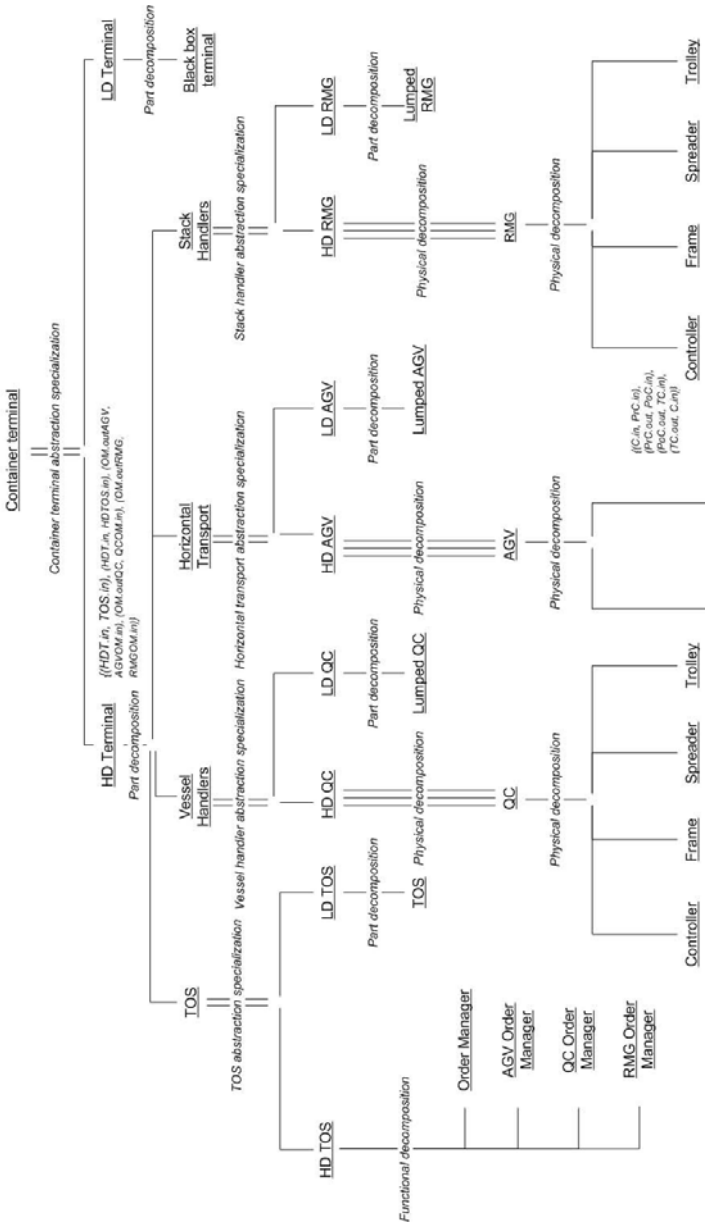


Fig. 3. An ontological representation of a container terminal helps defining the components needed for simulation-based design. The container terminal is mainly decomposed into a subset of the equipment used at a terminal. Other decompositions are due to different abstraction levels, which are explained in Section 4.2. (This ontology was originally presented in Fumarola *et al.*[23]).

levels. Paredis *et al.* [50] go a step further by constructing simulation models at different levels of abstraction. This solution lets designers analyze simulation results throughout the design process, even in the initial phases characterized by rough and incomplete designs. Further down the process, abstract simulation models are replaced by more concrete implementations, up until the end, where highly detailed simulation models are used to gather precise simulation results.

Abstraction in container terminal design processes can be enacted through spatial aggregation, as pictured in Figure 4. Looking at a container terminal at the whole, we achieve a typical black-box situation: we gather information about the behavior of the complete system but lack knowledge about the structural composition (noted as situation (A) in Figure 4). At a lower abstraction level, we can take the various zones of a container terminal into consideration such as yard and quay. The initial black-box container terminal is decomposed into interconnected zone-models (noted as situation (B) in Figure 4). At the lowest abstraction level, zones are decomposed into the contained equipment that operate in that zones (noted as situation (C) in Figure 4). The reversed function of the described decomposition is the aggregation of equipment: e.g. a set of quay cranes becomes the quay-zone, and the set of all equipment becomes the black-box container terminal. This decomposition is also sketched in Figure 3.

4.3 Structing Alternatives

Defining alternatives is one of the fundamental steps in most design methods, which we discussed extensively in Section 2. Human's bounded rationality impedes a large exploration of the design space: design studies are mostly limited to a few competitive alternatives. The large amount of alternatives that could be generated in a multi-actor environment is usually kept to a bare minimum by scrapping (supposedly) unfeasible designs in the early stages of the design process. The need for structuring and comparing alternatives has induced researchers in proposing solutions such as trees, matrices, rankings and charts.

Controlled experiments described by Dwarakanath and Wallace [21] bring more insights into the way designers think. Designers tend to follow paths along conceptual tree-like structures to assess alternative designs. Branches are made when different alternatives are being considered. Whenever a solution seems undesirable, pruning takes place to remove unwanted paths. The experiments led to a number of observations that stress the fact that designers go through an iterative process of finding alternatives, evaluate them (either formal or on intuition) and eliminate alternatives that perform less than the others. Although this experiment mainly studied informal methods, another experiment, documented by Girod *et al.* [25], discusses that using formal approaches tend to spawn better results. The formal approach forces designers to structurally evaluate their design and document the design process.

The results from these experiments lead us to propose a conceptual tree as a way to structure a design process. The tree elegantly embodies the generation of alternatives with branches. In light of the aforementioned constructs, the

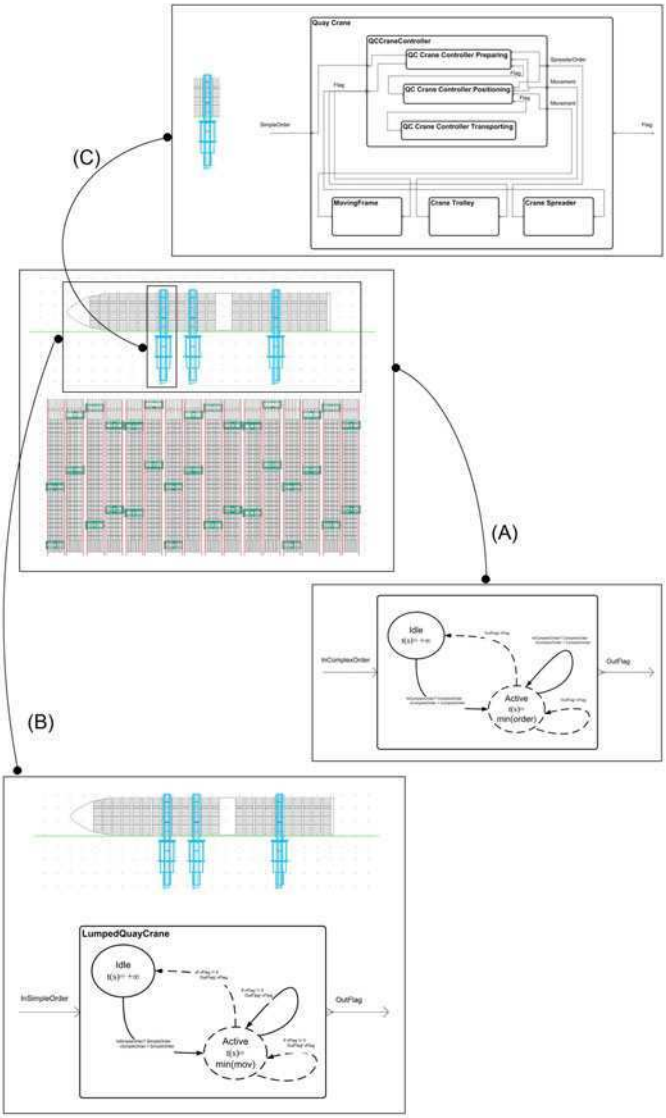


Fig. 4. Different levels of spatial abstractions that can be constructed of a container terminal. Each abstraction has a specific simulation model, specified in DEVS [70]. Situation (A) specifies a simulation model at the highest abstraction level: the model is characterized by the use of stochastic distributions to output performance measures. In situation (B), a container terminal is decomposed in different zones, each zone having a specific simulation model. Finally, situation (C) shows a highly structured container terminal, with simulation models for each type of equipment.

nodes of the tree contain alternative models (each model representing a design) that are constructed using pre-defined components. The breadth of the tree is defined by the number of alternative designs. The abstraction levels changes by constructing the tree in depth: the upper nodes of the tree contain the initial rough (highly abstract) models whereas deeper into the tree, we can find more detailed (concrete) models.

4.4 Participatory Design

The aforementioned constructs support a design process in a multi-actor environment: simulation components help to quickly generate solutions, abstraction levels lead the design process from rough artifacts to detailed solutions, and structuring the solutions supports designers in assessing too many alternatives.

A multi-actor design process is an iterative process where different actors try to achieve their own, sometimes conflicting, goals. A simulation environment supports analyzing decisions which are otherwise assessed based on the designer's implicit knowledge. By having a means to compare different alternative (pertaining to the different goals of the actors) analytically, grounded decisions can be made and insight can be provided into unforeseen failures or unexpected successes of certain decisions. The collaborative exploration of the solution space entails the learning process which is sought after by traditional soft systems approaches.

In the introduction given in Section 1, we discussed the different actors that are involved in the design process of container terminals. These actors are characterized by their goals: innovative solutions, reducing costs, feasible solutions, and adhering to environmental and labor regulations. In traditional simulation studies, conceptual designs are developed that are the results of negotiations between the actors. Major decisions are made before the actual simulation study has been performed and are based on informal assessments. By bringing formal techniques closer to the conceptual design process, discussions can be grounded and shared understanding can be achieved. This leads to an approach that is 'soft with a hard centre' [53].

5 Conclusion

Designing systems in a multi-actor environment is complex from the system as well as the actor perspective. Although classic operations research and systems engineering did not fully accomplish to provide successful approaches, they did provide valuable techniques to analyze systems. Due to an increased interest in facilitating the design process from an actor perspective, we do have insights in how actors behave and how we can support them. Multimethodological approaches are being developed and researchers commit themselves in covering both perspectives of complexity in design methods.

Starting from existing literature, we have presented the constructs that are needed in a multi-actor design method. The totality of constructs presents an

approach that is formal in describing the design artifact (using simulation components) and informal for the design process (actors have to explore the design space, without being restricted by an inflexible design method). Therefore, it is not a matter of "picking sides": to fully cover the complexity present in a design process, constructs from both approaches can be used.

We have shown how an approach that combines constructs from both the hard as well as the soft system's perspective that can be used in the design process of container terminals. To operationalize a design method using these constructs, a design environment has been implemented that visualizes and simulates designs that have been defined in AutoCAD. It can be concluded that to better support decision makers in a multi-actor environment, a multimethodological approach can be used wherein simulation is used as a tool for conceptual design and discussion.

References

1. Ackoff, R.: Science in the systems age: Beyond IE, OR, and MS. Operations Research 21, 661–671 (1973)
2. Ackoff, R.L.: The future of operational research is past. The Journal of the Operational Research Society 30, 93–104 (1979)
3. Alexander, C.: Notes on the synthesis of form. Harvard University Press, Cambridge (1964)
4. Aughenbaugh, J., Paredis, C.: The role and limitations of modeling and simulation in systems design. In: ASME 2004 International Mechanical Engineering Congress and Exposition, pp. 13–22. American Society Of Mechanical Engineers (2004)
5. Balci, O., Bertelrud, A., Esterbrook, C., Nance, R.: Visual simulation environment. In: Medeiros, D., Watson, E., Carson, J., Manivannan, M. (eds.) Proceedings of the 1998 Winter Simulation Conference, pp. 279–287. Institute of Electrical and Electronics Engineers, Inc., Piscataway (1998)
6. Baldwin, L., Eldabi, T., Paul, R.: Simulation in healthcare management: a soft approach (mapiu). Simulation Modelling Practice and Theory 12, 541–557 (2004)
7. Banares-Alcantara, R.: Representing the engineering design process: two hypotheses. Computer-Aided Design 23, 595–603 (1991)
8. Banks, J., Carson, J., Nelson, B., Nicol, D.: Discrete-Event System Simulation, 5th edn. Prentice Hall, Englewood Cliffs (2009)
9. Barros, F.: Dynamic structure multiparadigm modeling and simulation. ACM Transaction on Modeling Computer Simulation 13(3), 259–275 (2003)
10. Bennett, J., Worthington, D.: An example of a good but partially successful OR engagement: Improving outpatient clinic operations. Interfaces 28, 56–69 (1998)
11. Checkland, P.: The origins and nature of hard systems thinking. Journal of applied systems analysis 5, 99–110 (1978)
12. Checkland, P.: Soft systems methodology: a 30-year retrospective. John Wiley & Sons, Chichester (1981)
13. Checkland, P.: Systems thinking, systems practice. John Wiley & Sons, Chichester (1981)
14. Checkland, P.: Learning for Action: A Short Definitive Account of Soft Systems Methodology, and Its Use Practitioners, Teachers and Students. John Wiley & Sons, Chichester (2006)

15. Checkland, P., Poulter, J.: Systems Approaches to Managing Change: A Prac. Springer, London (2010)
16. Cross, N.: Science and design methodology: a review. Research in engineering design 5, 63–69 (1993)
17. de Bruijn, H., Herder, P.: System and actor perspectives on sociotechnical systems. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 39, 981–992 (2009)
18. den Hengst, M., de Vreede, G., Maghnouji, R.: Using soft OR principles for collaborative simulation: a case study in the Dutch airline industry. Journal of the Operational Research Society 58, 669–682 (2007)
19. Derksen, T.: Modelling Modes of Operation: A DSS and a decision making process on the selection of the mode of operation for APM Terminals. Master's thesis, Delft University of Technology, the Netherlands (2009)
20. Dieter, G.: Engineering Design: A Materials and Process Approach. McGraw-Hill, New York (1983)
21. Dwarakanatha, S., Wallacea, K.: Decision-making in engineering design: Observations from design experiments. Journal of Engineering Design 6, 191–206 (1995)
22. Dym, C., Agogino, A., Frey, D., Leifer, L.: Engineering design thinking, teaching, and learning. Journal of Engineering Education 94, 103–120 (2005)
23. Fumarola, M., Seck, M., Verbraeck, A.: A DEVS component library for simulation-based design of automated container terminals. In: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (2010)
24. Fumarola, M., Versteegt, C.: Supporting automated container terminal design processes with 3d virtual environments. In: Yand, H., Yuen, S. (eds.) Handbook of Research on Practices and Outcomes in Virtual Worlds and Environment, IGI Global (2011)
25. Girod, M., Elliott, A., Burns, N., Wright, I.: Decision making in conceptual engineering design: an empirical investigation. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 217, 1215–1228 (2003)
26. Goel, V.: Sketches of Thought. The MIT Press, Cambridge (1995)
27. Hall, A.: A Methodology for Systems Engineering. von Nostrand (1962)
28. Hazelrigg, G.: A framework for decision-based engineering design. Journal of Mechanical Design 120, 653–659 (1998)
29. Hazelrigg, G.: An axiomatic framework for engineering design. Systems engineering 121, 342–348 (1999)
30. Hirschheim, R., Klein, H., Lyytinen, K.: Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations. Cambridge University Press, Cambridge (1995)
31. Hoover, S., Rinderle, J.: Models and abstractions in design. Design studies 12, 237–245 (1991)
32. Hu, H.: Choosing the Optimal Mode of Operation for Marine Container Terminals. Master's thesis, Delft University of Technology, the Netherlands (2008)
33. Hu, X., Zeigler, B.: Model continuity in the design of dynamic distributed real-time systems. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 35(6), 867–878 (2005)
34. Hubka, V., Eder, W.: Theory of technical systems. Springer, Heidelberg (1988)
35. Jackson, M., Keys, P.: Towards a system of systems methodologies. The Journal of the Operational Research Society 35, 473–486 (1984)

36. Jenkins, G.: The systems approach. In: Beishon, J., Peters, G. (eds.) Systems Behaviour, 2nd edn. Harper & Row, London (1969)
37. Kindler, E., Coudert, T., Berruet, P.: Component-based simulation for a reconfiguration study of transitic systems. Simulation 80, 153–163 (2004)
38. Kotiadis, K.: Using soft systems methodology to determine the simulation study objectives. Journal of Simulation 1, 215–222 (2007)
39. Law, A., Kelton, W.: Simulation Modelling and Analysis. McGraw Hill Higher Education, New York (2000)
40. Lehaney, B., Hlupic, V.: Simulation modeling for resource allocation and planning in the health sector. Journal of the Royal Society of Health 115, 382–385 (1995)
41. Lehaney, B., Malindzak, D., Khan, Z.: Simulation modelling for problem understanding: a case study in the East Slovakia coal industry. Journal of the Operational Research Society 59, 1332–1339 (2008)
42. Lehaney, B., Paul, R.: The use of soft systems methodology in the development of a simulation of outpatient services at Watford General Hospital. Journal of the Operational Research Society 47, 864–870 (1996)
43. Mingers, J.: A classification of the philosophical assumptions of management science methods. Journal of the operational research society 54, 559–570 (2003)
44. Mingers, J., Rosenhead, J.: Problem structuring methods in action. European Journal of Operational Research 152, 530–554 (2004)
45. Mingers, J., White, L.: A review of the recent contribution of systems thinking to operational research and management science. European Journal of Operational Research (2010)
46. Munro, I., Mingers, J.: The use of multimethodology in practice: results of a survey of practitioners. Journal of the Operational Research Society 53, 369–378 (2002)
47. Oses, N., Pidd, M., Brooks, R.: Critical issues in the development of component-based discrete simulation. Simulation Modelling Practice and Theory 12, 495–514 (2004)
48. Otto, K.: Measurement methods for product evaluation. Research in Engineering Design 7, 86–101 (1995)
49. Pahl, G., Beitz, W.: Engineering Design: A Systematic Approach, 2nd edn. Springer, Heidelberg (1995)
50. Paredis, C., Diaz-Calderon, A., Sinha, R., Khosla, P.: Composable models for simulation-based design. Engineering with Computers 17, 112–128 (2001)
51. Pidd, M.: Making sure you tackle the right problem: linking hard and soft methods in simulation practice. In: Henderson, S., Biller, B., Hsieh, M.H., Shortle, J., Tew, J., Barton, R.R. (eds.) Proceedings of the 2007 Winter Simulation Conference, pp. 195–204. Institute of Electrical and Electronics Engineers, Inc., Piscataway (2007)
52. Qin, S., Harrisonb, R., Westb, A., Jordanovc, I., Wrighta, D.: Conceptual design of industrial systems: an approach to support collaboration. Computers in Industry 50, 153–164 (2003)
53. Robinson, S.: Soft with a hard centre: Discrete-event simulation in facilitation. The Journal of the Operational Research Society 52, 905–915 (2001)
54. Robinson, S.: Modes of simulation practice: approaches to business and military simulation. Simulation Modelling Practice and Theory 10, 513–523 (2002)
55. Robinson, S.: PSMs: looking in from the outside. Journal of the Operational Research Society 58, 689–691 (2007)
56. Rosenhead, J., Mingers, J.: Rational Analysis for a Problematic World Revisited: Problem Structuring Methods for Complexity, Uncertainty and Conflict. John Wiley & Sons, Chichester (2001)

57. Schmidt, L., Cagan, J.: Recursive annealing: A computational model for machine design. Research in Engineering Design 7, 102–125 (1995)
58. Simon, H.: The Sciences of the Artificial, 3rd edn. The MIT Press, Cambridge (1996)
59. Sommerville, I.: Software engineering, 7th edn. Addison Wesley, Reading (2004)
60. Stahlbock, R., Voss, S.: Operations research at container terminals: a literature update. OR Spectrum 30, 1–52 (2008)
61. Steenken, D., Voss, S., Stahlbock, R.: Container terminal operation and operations research – a classification and literature review. In: Gunther, H.O., Kim, K. (eds.) Container Terminals and Automated Transport Systems. Springer, Heidelberg (2005)
62. Stein, J., Stein, J., Louca, L.: A component-based modeling approach for system design: Theory and implementation. In: Proceedings of the 1995 International Conference on Bond Graph Modeling and Simulation, pp. 109–115 (1995)
63. Ullman, D., Dietterich, T., Stauffer, L.: A model of the mechanical design process based on empirical data. Artificial Intelligence for Engineering, Design, Analysis and Manufacturing 2, 33–52 (1988)
64. van Gigch, J.: System design modeling and metamodeling. Plenum Press, Plenum Publishing Corporation, New York (1991)
65. Verbraeck, A.: Component-based distributed simulations. the way forward? In: Proceedings of the 18th Workshop on Parallel and Distributed Computer Simulation, pp. 141–148. IEEE Computer Society Press, Los Alamitos (2004)
66. Vis, I., de Koster, R.: Transshipment of containers at a container terminal: An overview. European Journal of Operational Research 147, 1–16 (2003)
67. Visser, W.: The Cognitive Artifacts of Designing. Lawrence Erlbaum Associates, Mahwah (2006)
68. Von Bertalanffy, L.: An outline of general system theory. The British Journal for the Philosophy of Science 1, 134–165 (1950)
69. White, K.: Systems design engineering. Systems Engineering 1, 285–302 (1999)
70. Zeigler, B.P., Praehofer, H.: Theory of Modeling and Simulation. Academic Press, London (2000)
71. Zeigler, B., Hammonds, P.: Modeling and simulation-based data engineering: introducing pragmatics into ontologies for net-centric information exchange. Academic Press, London (2007)
72. Zeigler, B., Mittal, S.: Enhancing DoDAF with a DEVS-based system lifecycle development process. In: Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics, pp. 3244–3251. IEEE Computer Society Press, Los Alamitos (2005)
73. Zia, M., Mustafiz, S., Vangheluwe, H., Kienzle, J.: A modelling and simulation based process for dependable systems design. Software and Systems Modeling 6(4), 437–451 (2007)

Chapter 6
Distributed Simulation Using RESTful Interoperability Simulation Environment (RISE) Middleware

Khaldoon Al-Zoubi and Gabriel Wainer

Department of Systems and Computer Engineering,
Carleton University Centre for Visualization and Simulation (V-Sim)
Ottawa, ON K1S-5B6 Canada
kazoubi@connect.carleton.ca, gwainer@sce.carleton.ca

Abstract. Distributed simulation practice outside the military sector is still limited. Having plug-and-play or automatic middleware interoperability is one of the main challenges is needed to advance distributed simulation, as indicated by several surveys; hence, interoperability must be achieved effortlessly with rational cost. They further indicate the need of having general pluggable container where lightweight commercial-off-the-shelf (COTS) simulation components can be plugged into the container with minimal development time. However, existing middleware solutions have been complex so far to overcome these distributed simulation issues. The RESTful Interoperability Simulation Environment (RISE) is the first existing middleware to be based on RESTful Web-services. RISE uses the Web plug-and-play interoperability style to overcome distributed simulation issues. Our focus here on plugging simulation components into RISE and on interoperating independent-developed simulation engines to perform the same distributed simulation session.

Keywords: Distributed Simulation, REST, Web-service, SOA, Interoperability, DEVS, CD++.

1 Introduction

Distributed simulation technologies were created to execute simulations on distributed computer systems (i.e., on multiple processors connected via communication networks) [15]. Distributed Simulation is a computer program that models real or imagined systems over time. On the other hand, distributed computer systems interconnect various computers (e.g. personal computers) across a communication network. Distributed simulation offers many benefits such as: (1) allowing across-organization simulation collaboration in order to participate in same simulation run without the need of physically being in the same location, hence enabling simulation assets reuse, (2) Allowing complex simulation incremental development. In this case, a complex model can be divided into smaller models so they can be developed and verified individually. Afterward, these smaller models can be integrated together to form the overall complex simulation model. Other benefits also include reducing execution time, interoperating different vendor simulation toolkits, providing fault

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 129–157.
springerlink.com

tolerance and information hiding – including the protection of intellectual property rights [6][15].

Distributed Simulation middleware is responsible of connecting and synchronizing several simulation components across geographical regions, allowing simulation assets reuse without being physically at the same location. Interoperating scattered simulation assets is the main challenge of a distributed simulation middleware. In practice, making independently developed applications interact with each other is not a trivial task, since this interaction involves not only passing remote messages, but also synchronizing them (i.e. interpreting messages and reacting to them correctly). Particularly, simulation packages can be based on different formalism, implemented independently by different teams, or support different synchronization algorithms. In general, modelers use the simulation tools that they are familiar with, and can be experts within a simulation tool environment, but unable to use others.

The defense sector is currently one of the largest users of distributed simulation technology, mainly to provide virtual distributed training environment between remote parties, relying on the High Level Architecture (HLA) [21] middleware to provide a general architecture for simulation interoperability and reuse. On the other hand, the current adoption of distributed simulation in the industry is still limited in spite of HLA introduction in 1996. Other technologies such as CORBA and SOAP-based Web-services (WS) were used outside the military sectors to overcome HLA interoperability and scalability issues. However, existing distributed simulation middlewares still lack of plug-and-play interoperability, dynamicity, and composition scalability. Those approaches are described in the background section.

Lack of plug-and-play and dynamic interoperability to interface independent-developed simulation components, and the ability to reuse commercial-off-the-shelf (COTS) simulation components effortlessly are documented needed features in future distributed simulation middleware, as indicated by a number of surveys of experts from different simulation backgrounds such as [6][28]. Those surveys pointed out that having plug-and-play or automatic middleware interoperability is one of the main challenges to advance distributed simulation use in the industry; hence, interoperability must be achieved effortlessly with rational cost. They further indicate the need of having general pluggable container where lightweight commercial-off-the-shelf (COTS) simulation components can be plugged into the container with minimal development time. COTS concept reduces the cost of distributed simulation with the "Try-before-buy" mentality. This concludes that plug-and-play can mean two things. The first one is that any component in the overall system structure can be replaced with another one easily without affecting the entire system. The second one is that independent-developed simulation components can interoperate (synchronize) with each other for the same distributed simulation run. To achieve plug-and-play or Automated/semi-automatic interoperability between independent-developed simulation packages, not only semantics must be standardized but also flexible to adapt to future changes. Further, simulation functionalities should be self-contained components (black boxes) that: (1) Hide their internal software design and implementation, hence interact with other components with self-contained messages (e.g. XML messages) that are not tied to software implementation, uncomplicated to standardize and easy to adapt to future changes. Further, this point becomes more important since already existing simulation packages should be expected to have no or minimal software implementation changes to comply

with any new proposed standards, (2) Connect with other components via universal standardized interface (i.e. uniform connectors). In this case, components can be plugged into a complex structure easily, since they already know how they will be connected to other existing components in the structure, (3) Reached via unique universal standardized addressing scheme from anywhere, and (4) Support dynamic interoperability at runtime. Simulation components should be able to join/disjoin the overall structure without other components pre-knowledge. In other words, no new code or compilation should be required to achieve components interoperability. This point goes beyond simulation components to any device. In this case, real devices may be introduced into the simulation loop without stopping (and perhaps recompiling code) and restarting the current simulation-run in progress. We show here that RESTful Web-services interoperability contains the ingredients to advance distributed simulation on those fronts.

RESTful Web-services [26] imitates the Web interoperability style. The major RESTful Web-services (i.e. Web style) interoperability principles are universal accepted standards, resource-oriented, uniform channels, message-oriented, and implementation hiding. These principles are the Web interoperability characteristics; hence, REST is a reverse engineering of the Web interoperability style. Thus, REST has been in used in many products since the 1990's, but without its official name "REST". On the other hand, the Representational State Transfer (REST) is first used in [13] to describe the Web architecture principles. The name is derived because of the fact that on the Web a resource transfers its representation (state) in a form of a message to another resource. For example, a Web browser transfers a URI representation (e.g. as HTML document) using HTTP GET channel. REST exposes all services as resources with uniform connectors (channels) where messages are transferred between those resources through those uniform channels (i.e. called methods in HTTP standards). Because those characteristics conform to universally accepted standards, REST subsequently contains the recipe of plug-and-play and dynamic interoperability with infinite composition scalability. REST is a style, analogy with object-oriented, therefore, system designers must conform to those principles to obtain those benefits [26].

REST is usually implemented using HTTP, URIs, and usually XML because these are the main pillars of the Web today. In this case, resources (services) are named and addressed by URIs, resources connectors are HTTP channels, and connectivity semantics are usually described in XML messages. RESTful Web Services has been gaining increased attention with the advent of Web 2.0 [25] and the concept of mashup. Mashup applications deliver new functions and services on the Web by combining different information or capabilities from more than one existing source, allowing reusability and rapid development. Nowadays, RESTful Web-service is supported in conjunction with SOAP-based Web-services in tools developed by leading companies such as IBM [19] and Sun Microsystems (e.g. NetBeans IDE [24]).

Based on these ideas, we designed RESTful Interoperability Simulation Environment (RISE) middleware (formally called RESTful-CD++ [1][2][35]). RISE strictly follows the Web standards and interoperability style, hence, to avoid losing the main provided benefits such as plug-and-play and dynamic capabilities. Our main

motivations behind proposing such plug-and-play middleware with dynamic capabilities is to provide practical solutions for distributed simulation identified needed capabilities to overcome its limited use in the industry while maintaining rational cost. Having dynamic plug-and-play/automatic interoperability is recognized needed capabilities in a distributed simulation middleware [6][28][35].

Thanks to RESTful Web-services principles, RISE, which is the first existing RESTful WS middleware, is designed as a multipurpose online plug-and-play simulation interoperability middleware. First, the middleware provides a pluggable container to support different simulation components (e.g. CD++ [34]); hence, components become online Web services with minimal development time. Plugging commercial-off-the-shelf (COTS) simulation components quickly reduces cost and increases reusability. We plugged the distributed CD++ (DCD++) into RISE, allowing conservative-based distributed simulation between different CD++ instances. RISE-based DCD++ is described here along with its synchronization algorithms. Second, RISE forms the foundation for developing distributed simulation standards [3][30][31][32][33]. From our DEVS standardization [30][31][32][33] experience and the rationale behind CORBA declining [17], having practical standards indicates certain features that standards must be have such as simple to support, avoid software changes to legacy systems, allow legacy systems to use their existing resources (e.g. modeling methods), and allow different teams to evolve independently. The RISE-based standard is described here, aiming in interoperating independent-developed simulation packages.

In addition, RISE provides different functionalities that are not covered here such as making simulation assets part of workflows, Web 2.0 mashup, and Data fusion (DF). Workflows enable simulation experiments automation, repeatability and reusability, as described in [4]. Mashup concept groups various services from different providers and presents them as a bundle in order to provide single integrated service. IBM enterprise mashup solutions [19] [20] argue that integrating different RESTful plugging functions (called widgets) enable self-designed service Aggregation and information, rapid application development, unlock legacy systems via Web 2.0 [25] without major software upgrade. Thus, one of RISE objectives is to mashup applications/devices into simulation loop, allowing better-obtained results and analysis. DF is defined as collecting information from different sources to achieve inferences, which potentially leads to better accuracy from relaying on a single source of information. DF is applied by the military to build integrated images from various information sources in battlefields [27]. DF is similar to mashup in a sense of putting information into simulation loop. DF is highly dynamic, which makes it easier to achieve using RESTful WS plug-and-play interoperability.

2 Background on Distributed Simulation

At present, most works in distributed simulation are invested in optimizing simulation algorithms and in achieving efficient interoperability between different independent-developed simulation entities. These two areas define the current challenges of distributed simulation and future trends [28]. For further thorough details, we discuss distributed simulation current state-of-the-art in [35].

Parallel/distributed simulations are typically composed of a number of sequential simulations where each is responsible of part of the entire model. Each of these subparts is a sequential simulation, which is usually referred to as a logical process (LP). The main purpose of synchronization algorithms is to produce the same results as if the simulation were preformed sequentially in a single processor. The second purpose is to optimize the simulation speed by executing the simulation as fast as possible. They fall in two categories: Conservative and optimistic. Conservative algorithms were introduced in late 1970s by Chandy-Misra [9] and Bryant [8]. This approach always satisfies local causality constraint via ensuring safe timestamp-ordered processing of simulation events within each LP. In current systems, the common implementation of conservative-based distributed simulation cycle to advance simulation time (e.g. [9][10][39]) is summarized as follows: (1) Time-coordinator requests minimum time from all LPs. (2) Time-coordinator calculates global minimum time, broadcasts it to all LPs, and waits for their replies. (3) Time-coordinator instructs all LPs to execute events with the minimum global time, waits for all LPs replies, and starts again with step #1. In optimistic algorithms, each LP maintains its Local Virtual time (LVT) and advances "optimistically" without explicit synchronization with other processors. On the other hand, a causality error is detected if a LP receives a message from another processor with a timestamp in the past (i.e. with a time-stamp less than the LVT); such messages are called straggler messages. To fix the detected error, the LP must rollback to the event before the straggler message timestamp; hence undo all performed computation. Time Warp algorithms focus on providing efficient rollback by reducing memory and communication overhead such as the mechanisms presented in [15].

Distributed simulation Middleware main objective is interfacing different simulation environments, allowing synchronization for the same simulation run across a distributed network. Those simulation entities are usually heterogeneous. For example, each simulation environment may differ from other entities in its simulation engine, algorithms, model representation, and formalism. This comes as no surprise that a number of surveys placed the middleware of distributed simulation as the most area of interest to overcome current distributed simulation challenges and to meet future expectation, as indicated by a number of surveys of experts of different simulation background [6][28].

The defense sector is currently one of the largest users of distributed simulation technology, mainly to provide virtual distributed training environment between remote parties, relying on the High Level Architecture (HLA) [21] middleware to provide a general architecture for simulation interoperability and reuse. On the other hand, the current adoption of distributed simulation in the industry is still limited. Further, HLA could not make a breakthrough in the industry since its adoption in 1996 due to a number of issues such as its complexity, tied to programming languages and lack of interoperability in interfacing different Run-Time Infrastructure (RTI) vendors, since RTI-to-RTI interface is not standardized. RTI is the software layer that connects and synchronizes different HLA simulation entities (called federates) together where federates are interfaced with local RTIs via callback function interface (Figure 1). The HLA interoperability and scalability issues have caused the consideration of using existing Service-oriented architectures (SOA) technologies in

distributed simulation middleware, mainly CORBA [18], SOAP-based WS [12], and RESTful WS [26].

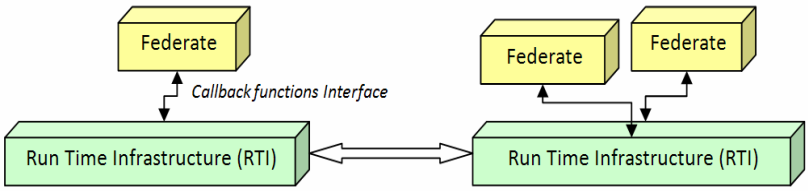


Fig. 1. HLA Interaction Overview Model

WSDL and SOAP are the main elements enable SOAP-based Web-services (WS) interoperability. SOAP-based Web-services provides interoperability in a similar way as CORBA: WSDL corresponds to IDL role whereas SOAP corresponds to ORB data marshalling/serialization function. Further, Web-service ports addressed by URIs whereas CORBA objects addressed by references. Both ports and objects contain a collection of procedures (i.e. called services by WS) similar to a Java/C++ classes. Those procedures glue software components across the network, hence providing and RPC-style type of software interoperability, as shown in Figure 2. The server exposes a group of services via ports (Figure 2). Each service is actually an RPC where semantic are described via that procedure parameters. Client programmers need to construct service stubs with their software at compile time. Clients, at run time, consume a service by invoking its stub, which is in turn converted into XML SOAP message (to describe the RPC call), wrapped within HTTP message and sent to the server port, using the appropriate port URI. Once the message is received at the server side, the HTTP server passes the message into the SOAP layer (usually called SOAP engine like Apache AXIS [36]). SOAP engines are usually running inside HTTP servers as Java programs, called Servlets. The SOAP layer parses the SOAP message and converts it into an RPC call, applied to the appropriate procedure of the proper port. The server returns results into clients in the same way. Thus, the SOAP message role is to provide a common representation among all parties to the invoked procedure at runtime. In a distributed simulation environment, different components act as peers to each other. This means that each acts as client when it needs to send information while acts as a server via exposing different RPCs (i.e. services), as shown in Figure 2. Service providers need to publish their services, as XML WSDL documents. Clients programming stubs (Figure 2) are generated via compiling the WSDL document into programming stubs. Programmers then need to write the body of those stubs and compiling them with their software. [23][29] are examples of SOAP-based WS distributed simulation.

In reality, RPCs are heterogeneous interface, since they were invented by different programmers, and need to be written and compiled before being used. RPCs also expose internal implementation, leading to impractical and complex interoperability standards. It is almost impossible to interoperate independent-developed simulation systems via RPC-style without requesting major software implementation changes. This makes it impractical to support. Further, existing solutions lack of composition

scalability, for example, programming stub is needed for every remote service. However, in case of HLA the scalability is even worst, since the RTI is treated like a bus where all simulation entities use it for synchronizations. Furthermore, API complexity makes it difficult for distributed simulation to break outside expert programmers circle.

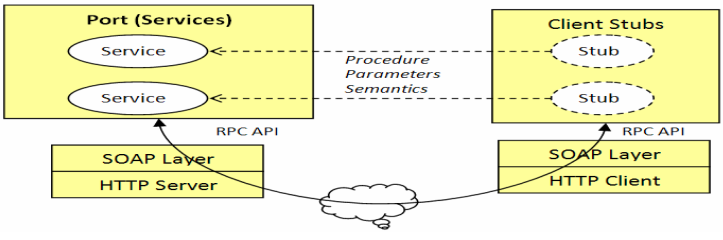


Fig. 2. SOAP-based WS RPC-based Architecture Model

RESTful WS exposes all services as resources with uniform connectors (channels) where messages are transferred between those resources through those uniform channels. REST is usually implemented using HTTP, URIs, and usually XML because these are the main pillars of the Web today. In this case, resources (services) are named and addressed by URIs, resources connectors are HTTP channels (usually called methods), and connectivity semantics are usually described in XML messages (Figure 3). RESTful applications APIs are expressed as URI templates [16] that can be created at runtime. Variables in URI templates (written within braces {}) are assigned at runtime by clients before a request is sent to the server, enabling clients to name their services URIs at the server side. For example, username in template <…/users/ {username}> can be substituted with any string to get the actual URI instance (such as <…/users/user1> or <…/users/user2>). Further, URIs may include query variables to define the request scope by appending them to a URI after the question mark "?". For instance, request via GET channel to URI <http://www.google.com/search?q=DEVS> would instruct Google search engine to return information only about keyword "DEVS". RESTful services can be described formally using XML either using Application Description Language (WADL) [37] or WSDL 2.0 [22][38].

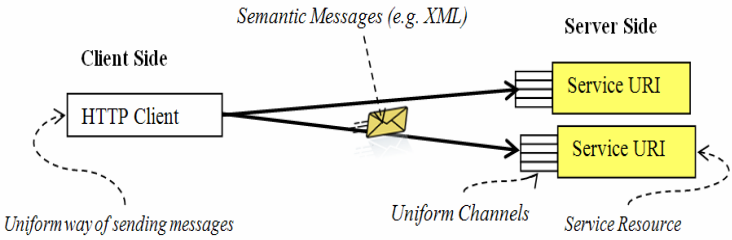


Fig. 3. RESTful WS Architecture Model

From distributed simulation viewpoint, there are some differences between SOAP-based WS and RESTful WS as follows: (1) SOAP groups all services as procedures and expose them via a port (i.e. addressed by single URI) whereas REST exposes each service as a resource (i.e. addressed by single URI). (2) SOAP-based WS communicates simulation information (i.e. semantics) in form of procedure parameters whereas REST defines them as XML messages. (3) SOAP-based WS transmits all SOAP messages (i.e. RPC description) using HTTP POST channel whereas REST uses all HTTP channels to transfer simulation semantics. (4) SOAP-based WS clients need to have a stub for each corresponding service while REST clients communicate in the same uniform way. (5) SOAP-based WS client stubs skeleton usually built via tools, but they still need to be written, integrated with existing software and compiled by programmers whereas REST does not usually require this process, hence follows a dynamic approach.

REST critics usually raise few issues such as REST only uses the four HTTP channels to transfer all messages so that those methods might not be enough for some applications: mainly, GET (to read), POST (to append new data), PUT (to create/update), and DELETE (to remove). This misleading comes from naming those virtual channels as "methods" in HTTP standards (RFC 2616 [14]), hence being confused with regular programming methods. Perhaps, it is ample to mention that SOAP-based WS transfers all SOAP messages using only the HTTP POST channel, thus, single method is enough in this case. Another issue is that REST heavily depends on HTTP, on other hand; SOAP-based WS can send SOAP messages using different protocol from HTTP like TCP/IP. This is because SOAP is a message describes an RPC via a network so that it can be sent using TCP socket. This is a misleading issue because: (1) HTTP is the Web protocol, thus sending SOAP messages using different protocol from HTTP makes it not Web service any more, hence complicates interoperability with other heterogeneous even further. (2) REST is message-oriented, thus, those messages are portable to different protocols like TCP/IP. For example, all simulation synchronization messages presented here portable to different protocol, similar to SOAP. However, a universal standard is part of REST principles and makes no sense to use different protocol from HTTP.

3 RISE Middleware API

Each experiment is wrapped up and manipulated via a set of URIs (i.e. an experiment API), hence allowing their online access from anywhere. Simulation experiment is various resources (URIs) hold all necessary information for simulation setup such as model scripts and model partitions where they are simulated in a single simulation run. These URIs are created and manipulated according to the middleware URI template (API), shown in Figure 4. The full RISE design and API described in [1][2]. The URI API template can be created at runtime. Variables (written within braces {}) in URI templates are assigned at runtime by clients before a request is sent to the server. The resource that best matches the request's URI will receive the request and it will become its responsibility to respond to the client.

Line #1 (Figure 4) shows a specific user workspace. This allows multiple users to use the middleware where each owns a single workspace (e.g. …/workspaces/Bob).

Line #2 holds a specific service supported by RISE such as DCD++ (e.g. …/workspaces/Bob/DCDpp). In this case, for instance, other simulation components may be supported by RISE similar to adding new links to a Web site. Modelers (clients) usually interact with a number of resources during the course of a simulation experiment, as shown in Lines 3-6: (1) the framework resource (Line #3) holds an experiment input data (such as the model source code, simulation input variables and sub-models interconnections). The POST channel is used to submit files to a framework. PUT is used to create a framework or update simulation configuration settings. DELETE is used to remove a framework. The GET channel is used to retrieve a framework state. (2) A simulation resource (Line #4) wraps an active simulation engine (e.g. CD++), which interacts with other remote simulation, if any. It is worth to note that in case of DCD++, this URI is the modeler single entry to a simulation experiment. However, it needs to communicates with other URIs (e.g. on different machines) to perform distributed simulation. This resource exchanges synchronized messages with other simulation entities (in case of distributed simulation) via the POST channel, and POST can be used by modelers to input variables in order to manipulate simulation at runtime dynamically. The PUT channel is used to create this resource, hence to start simulation. The DELETE channel is used to abort simulation and remove this resource. (3) The results resource (Line #5) holds the simulation output files (if the simulation was completed successfully). The GET channel is used to retrieve results where the DELETE channel is used to remove those results. The PUT and POST channels are disabled for this resource. (4) The debug resource (Line #6) holds model-debugging files. For example, a modeler can print debugging information inside his model source code to be retrieved later via this resource. The GET channel is used to retrieve model-debugging files where the DELETE channel is used to remove those files. PUT and POST channels are disabled for this resource.

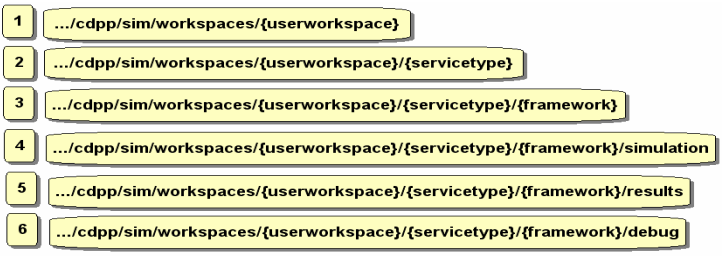


Fig. 4. Simulation Experiment API in RISE

4 RISE-based Distributed CD++ Simulation Algorithms

This section discuss the distributed CD++ (DCD++) simulation session between different CD++ instances. At this point, a modeler should have already created an experiment URI (i.e. …/{framework}) where {framework} is the experiment name created by the modeler. Note that this URI is the parent for all other URIs that are created or deleted during the simulation process. This section is divided into two

parts: the first part discusses the distributed simulation architecture while the second part discusses the simulation synchronization algorithms. The CD++ is plugged into RISE as shown in Figure 5 where each CD++ instance is reached via a URI and accessed via HTTP channels.

The purpose of the simulation manager (Figure 5) component is to manage a distributed CD++ (DCD++) simulation engine instance in the distributed simulation environment where various DCD++ instances participate to execute single simulation experiment. A simulation engine instance is usually called Logical processor (LP) in the distributed simulation environment, CD++ in our case. The simulation manager is able to synchronize a DCD++ instance with another remote DCD++, using the presented algorithms and semantics here. It is also capable to synchronize a DCD++ instance with none-CD++ simulation engine using standard protocols semantics, hence the ability of multiple semantics support. In DCD++, single DEVS or Cell-DEVS model is partitioned among those DCD++ engines where each instance simulates its partition.

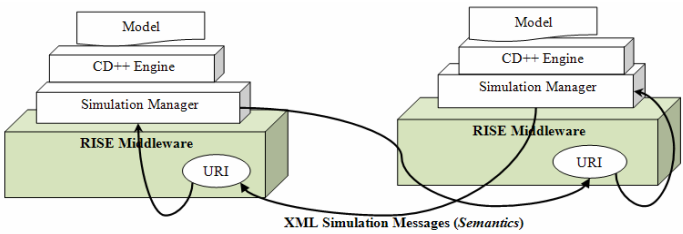


Fig. 5. Distributed Simulation between two CD++ instances

DCD++ follows the conservative synchronization approach in which the casualty is strictly prohibited. On the other hand, it provides a number of improvement techniques comparing to other existing conservative-based simulation summarized as follows: (1) it avoids the required steps to loop all simulation entities to calculate the simulation global minimum time and then broadcasting it to all entities before an entity being able to proceed. This allows Root coordinator (which manages time) to start a new simulation phase without asking each logical processor (LP) its minimum time. (2) It aggregates remote simultaneous events together in single XML message, hence reducing the cost of several network messages to the cost of one message. (3) Provides modelers with experimental framework template where they can freely create as many as they like of different simulation scenarios. (4) It avoids unnecessary remote message transmission when it can be performed locally. (5) It avoids involving irrelevant models within current simulation phase (i.e. models that do not have events to execute or to send/receive at current time). This method can ignore huge part of the model partitions at certain simulation phases. (6) It uses simultaneous message transmissions to avoid blocking messages when a number of messages need to be sent to multiple remote LPs. (7) Exploiting thread-pool concepts to avoid creating a thread every time a message is sent. (8) Reusing TCP connections to transmit multiple HTTP messages to avoid establishing a connection with every message, which is very expensive.

4.1 Distributed CD++ (DCD++) Architecture

In the RESTful DCD++ grid various machines need to coordinate and exchange simulation messages (as HTTP messages) to carry out the simulation. Each physical machine in the grid needs to have at least one instance of the RISE middleware installed on it, since the DCD++ is plugged into it, as shown in Figure 6. DCD++ instances act as peers to each other. This means that when a simulation message is sent to an URI, the sender is an HTTP client, delivering an HTTP request using an HTTP channel where the receiver URI is a server, processing HTTP requests and responding with HTTP responses according to the HTTP standards.

Figure 6 shows an example of three DCD++ engines in distributed simulation conference where each DCD++ instance is plugged into RISE middleware. This conference represents an experiment during active simulation. The modeler manipulates and interacts with the simulation via the main DCD++ instance URIs, which resides on the main RISE middleware. The main RISE is the server that the modeler has on it a user account, selects it to setup experiments, and executes them. CD++ simulation engines are actually online simulation services that can be reached via URIs and accessed via HTTP channels. Thus, a main RISE in an experiment is not necessary the main middleware in another experiment. Further, the main server (e.g. machine #3 in Figure 6) sets up experiment resources on supportive servers on behalf of the modeler. In this case, the main RISE owns those resources; hence, it instructs supportive servers to hide all of its resources from external users. After all, those resources are URIs on the Web.

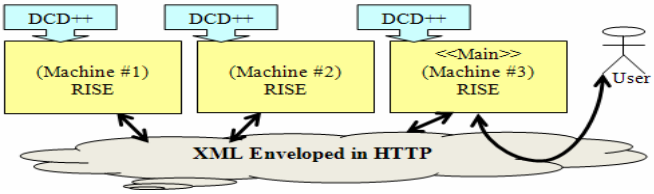


Fig. 6. Conceptual View of a Distributed Simulation Session

Plugging components (e.g. DCD++) into the middleware provides a separation between provided services and the middleware. This clearly provides a number of advantages such as simulation components become independent of underlying technology, hence moving easily to another technology that might appear in the future, and applying the concept of pluggable container middleware where lightweight commercial-off-the-shelf (COTS) simulation components can be plugged into the middleware with minimal development time. COTS concept reduces the cost of distributed simulation with the business mentality of "Try-before-buy" attitude [6][28].

Each active DCD++ simulation component instance is wrapped by URI (…/{framework}/simulation), as shown in Figure 7. The modeler creates this URI via PUT channel on the main RISE server to start the simulation, which in turn starts the simulation on other supportive RISE servers. The request to start a simulation on RISE creates all necessary Inter-Process Communication (IPC) queues, simulation

managers and the DCD++ simulation engines. During active simulation, as shown in Figure 7, simulation managers send messages to remote active-simulation URIs (where it is then passed to the corresponding simulation manager). Simulation managers communicate with the actual CD++ simulation engines via operating system IPC queues, since CD++ runs as a separate process outside RISE middleware. It is worth to note that the modeler may use URI (…/{framework}/simulation) to manipulate simulation during runtime such as inserting an external event (i.e. simulation input variable) to change the course of the simulation. This is helpful during simulation training session when instructors like to change conditions during an exercise.

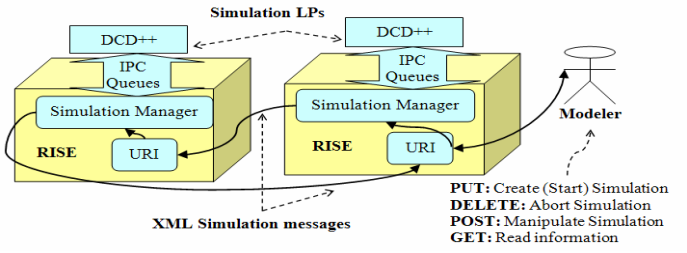


Fig. 7. DCD++ Simulation Session between Two Machines

The DCD++ virtual network (Figure 7) is constructed and destructed based on the way a modeler partitions the model under simulation between different machines. Figure 8 shows example of DCD++ XML model partitioning information for both standard DEVS models (the top figure) and the Cell-DEVS model (the bottom figure). Model partitioning is a section of a larger XML configuration document for customizing the entire experiment options. The model-partitioning document describes each atomic model or cells zone location. Thus, the DCD++ virtual network shown in Figure 7 is reconstructed, if modeler redistributed models across different machines. Note that the DCD++ simulation session is actually performed among different URIs (within one or more RISE instances) coordinating among each other. However, those URIs are usually located on different physical machines. Note further that a RESTful-CD++ is identified via its port and IP address (Figure 8), relieving modelers of figuring out full URIs path every time a model is moved to another machine.

```
1  <DCDpp>
2      <Servers>
3          <Server IP="10.0.40.162" PORT="8080">
4              <MODEL>reception</MODEL>
5          </Server>
6          <Server IP="10.0.40.175" PORT="8080">
7              <MODEL>cuthair</MODEL>
8              <MODEL>checkhair</MODEL>
9          </Server>
10     </Servers>
11 </DCDpp>
```

(A) Distributing Three Atomic DEVS Models Between Two Machines

```
1  <DCDpp>
2      <Servers>
3          <Server IP="10.0.40.175" PORT="8080">
4          <ZONE>fire(0,0)..(14,29)</ZONE>
5      </Server>
6          <Server IP="10.0.40.162" PORT="8080">
7          <ZONE>fire(15,0)..(29,29)</ZONE>
8      </Server>
9      </Servers>
10 </DCDpp>
```

(B) Spliting a Cell-DEVS Model Between Two Machines

Fig. 8. XML Model Partitioning Example

The modeler (i.e. client GUI software) is expected to check on the active simulation status periodically. This is usually done via GET channel to URI (…/{framework}?sim=status). In this case, RISE responds with an XML document similar to the following: *<Simulation><Status>RUNNING</Status></Simulation>*. The simulation goes into different states (from the modeler viewpoint), as shown in Figure 9: IDLE, INIT, RUNNING, ABORTED, ERROR, DONE and STOPPING. When a framework is created, the status is initialized with the IDLE state, which indicates that the simulation was never run on this framework. It moves into the INIT state upon receiving the request to start the simulation. The simulation goes into RUNNING state, if initialization was successful. The RUNNING state indicates that all simulation engines everywhere are up and running. In this state, the CD++ simulation engines can exchange simulation messages. Further, the modeler can manipulate simulation like inserting external events. Furthermore, dynamic online simulation results can be retrieved during this state. The simulation goes from RUNNING state to ERROR because of various possible errors such as failing to transmit a simulation message or a server failure in the grid. Further, the simulation goes into ABORTED state, if the modeler chooses to stop the simulation during the RUNNING state (via applying DELETE method to resource {framework}/simulation). In the normal completion, the simulation goes into STOPPING state. In this state, the main server collects simulation results from all supportive servers. The simulation goes into ERROR if it fails to stop properly such as failing to collect results from supportive servers or failing to stopping supportive simulations. Upon normal completion, the simulation status goes into DONE state, which means simulation results can now be retrieved from URI …/{framework}/results. Note that releasing system resources such as Linux queues, threads and processes occur in all exiting states: ABORTED, ERROR and STOPPING.

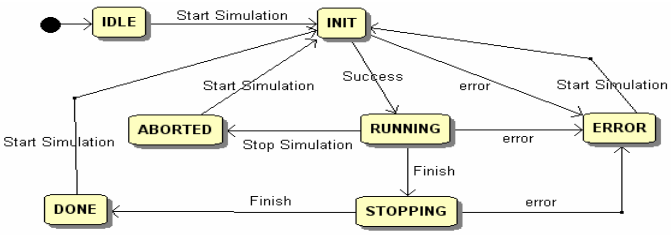


Fig. 9. Simulation State Diagram

Simulation is automatically aborted (to ensure simulation accuracy) by a simulation manager, if, for any reason, it fails to transmit a simulation message to a remote simulation URI during a session. In this case, if the simulation manager is supportive, it aborts simulation and silently removes itself from the distributed simulation conference. On the other hand, if it is the main simulation manager, it also aborts simulation on all other supportive servers, since it is the actual owner of all simulation resources in the session. To make the matter worse, suppose a supportive server fails while the main server is waiting for a DONE simulation message from a process on that failed supportive server (simulation phases are discussed in next section). In this case, the Root coordinator, which drives the whole simulation, cannot advance the simulation to another phase because it is waiting for a DONE message from a dead simulation participant. This

is a deadlock situation. To overcome this possibility of deadlock, the main simulation manager starts a watchdog thread at the beginning of the simulation (and stops it at the end of the simulation) to keep watching all supportive simulation resources, as shown in Figure 10. The watchdog sends periodic (e.g. every two minutes) messages to every simulation URI checking if it is alive or dead. The main simulation manger only hears from the watchdog the bad news, which leads to aborting the simulation everywhere. Therefore, the session stays in deadlock at most for the watchdog period before the simulation is aborted. Supportive servers also need to watch the main server (Figure 10). This allows them to release system resources such as processes, threads, connections, and IPC queues.

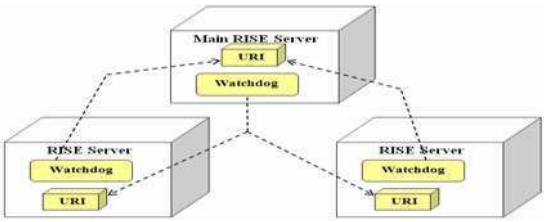


Fig. 10. Watchdog Periodic Checking in a Simulation Session

HTTP messages are synchronous. This means that when an HTTP message is sent via TCP connection, the sender is blocked until a response is received. This argument also applies to the RPC-style SOAP-based web-services because SOAP messages (that describe RPCs) are enveloped in HTTP messages; hence, it is still an HTTP synchronies transmission. This fact often goes unnoticed by SOAP-based WS programmers. This is because SOAP engines handle SOAP messages at a different layer of the software stack. Further, SOAP engines are often used from a third-party provider through available open source like Apache AXIS [36]. HTTP synchronies transmission is obviously a performance concern, particularly when multiple messages need to be sent at the same time to different destinations. For this reason, simulation messages are transmitted concurrently where each message lives on its own thread. Figure 11 shows example of two simulation managers. The top manager is sending two concurrent messages (each message is actually an HTTP client thread) where the bottom manager is receiving two messages concurrently (assuming via the same URI). Therefore, receiving messages by a simulation manager must be thread-safe to avoid message contention, since each request is handled by a separate thread. Further, in this case only the sender-message thread is blocked until the HTTP response is received back without blocking the entire application or other messages transmission. Note that all RISE threads are started from a thread pool, avoiding a new thread creation every time a thread is started.

Security is always a concern when communicating in cloud computing environment as in the case of RESTful-Web services. Other based RESTful Web-services such as Amazon Web-service (AWS) which requires developers to apply for an “Access Key ID” and a “Secret Access Key” [5]: The “Access Key ID” identifies the developer who is accessing AWS while the “Secret Access Key” is used to generate a keyed-Hash Message Authentication Code (HMAC), enabling AWS to authenticate the user. HMAC is calculated over service (i.e. URI), operation (e.g. user authorized to use POST channel or not to use), and timestamp (i.e. to prevent replay attacks). To prevent in-flight

tampering, AWS recommends all requests should be sent over HTTPS [5]. This scenario is portable for RISE. On the other hand, we chose to encode user name and password into a single string with base 64 encoding according to HTTP Basic Authentication method, defined in RFC 2617. This method does not add extra overhead, and it is supported by Web browsers and Web programming languages. Therefore, all simulation messages need to be authenticated according to this method. Note that the main server authorizes all simulation participants to use POST channel on all URIs, allowing them to pass simulation messages within the simulation conference.

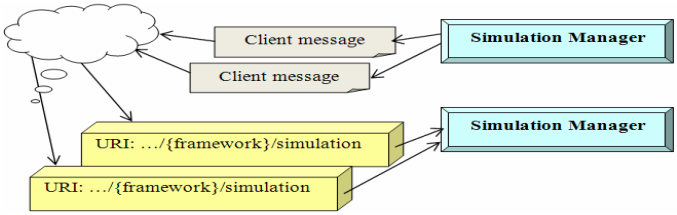


Fig. 11. Concurrent Message Passing to/from Simulation Managers

4.2 DCD++ Simulation Synchronization Algorithms

DCD++ executes the model by passing messages among the different processors in the model hierarchy. Coordinators are the processors responsible for executing coupled models while Simulators are associated with atomic models and they are responsible for executing each of the DEVS functions defined by the model depending on the time and type of the received message. A Root coordinator is in charge of driving the simulation as a whole and interacting with the environment, since DCD++ is a conservative-based engine. Because DCD++ is a conservative-based engine, there is a special coordinator called Root coordinator which is responsible for the following: (1) Starting and stopping the simulation, (2) Connecting the simulator with the environment in terms of passing external events/output from/to the environment, and (3) Advancing the simulation clock. As shown in Figure 12, "coordinator" processors coordinate the simulation of one or more coupled/atomic models where "simulator" processors simulate atomic models. The processors are created and initialized at the beginning of the simulation in a hierarchy that matches the model hierarchy in terms of the parent-child relationship.

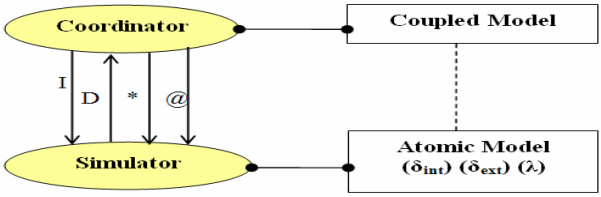


Fig. 12. Message exchange during a simulation cycle

A number of simulation messages are used to synchronize simulation among processors hierarchy, shown in Figure 12. Simulation messages can be categorized as follows: (1) Content messages represent events generated by a model. Content simulation

messages include External messages (X) and Output messages (Y). Output messages (Y) are usually converted to external messages for their destinations. (2) Synchronization messages cause the simulation to move into another simulation phase (those phases discussed next). Synchronization messages include Initialize message (I), Internal message (*), Collect message (@), and Done message (D). Initialize message (I) starts the initialization phase. Internal message (*) starts the transition simulation phase. The top model Coordinator propagates it downward in the hierarchy. Collect message (@) starts the collection phase. Done message (D) marks a simulation phase end. It is also used by Coordinators to identify which children needs to be simulated at the next phase. It further used to calculate the global minimum simulation time.

The simulation phases for the entire simulation are driven by the Root coordinator (which is the parent of the highest model's coordinator). They are divided into three phases (shown in Figure 13): (1) Initialization phase initializes all models in the hierarchy; hence, it eventually executes every initialization method of every atomic model. In response, a DONE message propagates upward in the model hierarchy where each Coordinator calculates the minimum next change of its children and passes it in a DONE message to its parent. Eventually, the Root receives DONE message with smallest time, which updates the simulation clock and starts the Collection phase. (2) In the Collection phase, some of the output messages are collected to ensure their execution at the same time with internal events. (3) In the Transition phase, all the collected external messages are executed along with simultaneous internal events. The Root coordinator handling of a DONE message arrival is described in Figure 14.

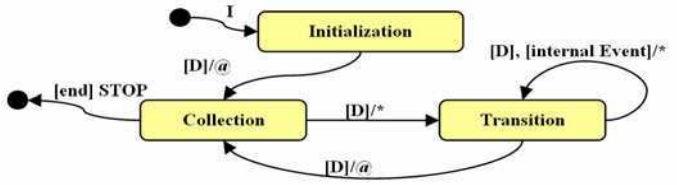


Fig. 13. Root Coordinator Simulation Phases State Diagram

```
// Next Phase is initialized to Transition
Root Coordinator::ReceiveDoneMessage (DoneMessage msg) {

   If (Next Phase == Transition) {
      // Start transition phase
      Next Phase = Collect;
      Send Internal (*) Msg to highest model;
   } Else {
      Time = Current Time + Next Change in msg;
      If (Time <= STOP_TIME) {
         Send Stop to all;
      } Else {
         While (environment event == Time){
            Send environment external event to highest model;
         }
         If (Next Event is NOT external) {
            // Start the Collect Phase
            Next Phase = Transition;
            Send Collect (@) Msg to highest model;
         } Else {    // Start transition phase
            Next Phase = Collect;
            Send Internal (*) Msg to top model;
         }
      }
   }
}
```

Fig. 14. Root Coordinator Handling DONE Message Algorithm

The head/proxy is originally intended to solve redundant number of messages from remote CD++ processors back to their parent coordinator. The main motivation behind Head/Proxy algorithm is that network messages in distributed environment are expensive

and have direct affect on performance. For example, assume the coordinator in Figure 15 is coordinating three simulators where two of its children (simulator #2 and #3) are residing on a remote machine. Figure 15 shows a fragment of the collection phase messages when the coordinator receives a collect (@) message from its parent. As shown in Figure 15, Simulator 3 sends an output message to the parent coordinator to translate it to external message for Simulator 2. Obviously, the transmission of those two messages (in Figure 15) could have been avoided if another coordinator (we call proxy) was placed in server 2 so that converting the output message (from simulator 3) to an external message (to simulator 2) is done locally, as shown in Figure 16.

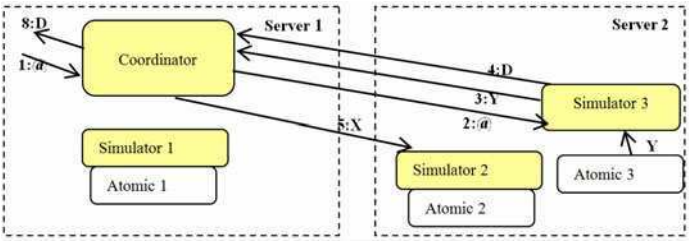


Fig. 15. Unnecessary remote messages in distributed simulation

The idea of the head/proxy depends on using two kinds of coordinators for each coupled DEVS/Cell-DEVS model: (1) Head Coordinator: is responsible for synchronizing the model execution, interacting with upper level coordinators and message routing among the local and remote model components. (2) Proxy Coordinator: is responsible for message routing among the local model components dispensing with the need to send remote messages if the head coordinator is residing on a different machine than that used to run the sending and receiving processors. The advantage of using proxy coordinators (as shown Figure 16) is that converting all remote messages between local processors to local messages. The proxy coordinator forwards one DONE message to the head coordinator once it receives all DONE messages from its children. Note that in this collection phase (Figure 16) simulator #2 does not forward the external message to the Atomic #2 model. In this phase, simulator #2 inserts the external message in its bag, waiting for the next internal (*), which starts the next transition phase. This allows simulator #2 to execute any scheduled internal events along with the already collected external message simultaneously.

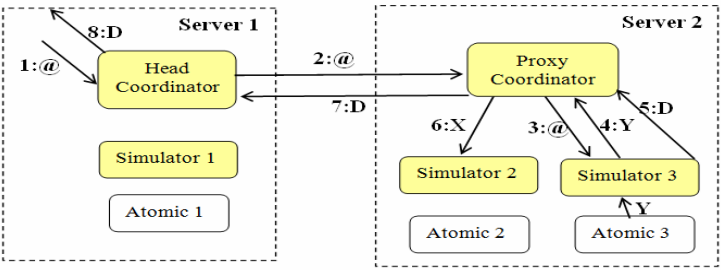


Fig. 16. Proxy Advantage of Preventing Unnecessary remote messages

Proxy coordinators avoid remote message transmission when it is possible to route them locally, but still forward all none-local messages to the head coordinator. For example, suppose the output message from simulator #3 is transmitted to simulator #1 (instead of simulator #2), as shown in Figure 17. In this case, the proxy coordinator has no choice but to transmit the external message to the head coordinator remotely. The external message ends up queued at simulator #1, waiting to be executed in the next transition phase upon receiving internal (*) message from head coordinator. In fact, simulation events that are exchanged in the same simulation phase are simultaneous events; hence, they need to be executed in the same virtual time. To clarify this point, consider how Root coordinator advances simulation Time and phases, as shown Figure 18, which is a depiction of the model hierarchy partitions shown in Figure 17. Assume that Simulators #2 and #3 outputs a job to simulator #1 every two seconds where Simulator #1 takes one second to process each regardless of the number of jobs are being process. In this simple example, shown in Figure 18-A: (1) as part of simulation initialization, I message is sent to the Head coordinator, which passes it to Simulator #1 and Proxy coordinator. Consequently, the Proxy reroutes message I to Simulator #2 and #3. Simulator #2 and #3 reply with D messages with a scheduled change in two seconds from now. (2) Root advances time to (t2) and starts Collection phase by sending @ message to Head coordinator, which only sends it to the proxy. This message is not send Simulator #1 because it did not schedule a change in previous phase, hence becomes irrelevant in this phase. The Proxy passes @ message to Simulator #2 and #3, which cause them to send two jobs (i.e. Y message) to Simulator #1 (via Head and Proxy coordinators). Simulator #1 receives these Y messages as external messages (X) where it holds them to be executed in the next phase. (3) Root starts transition phase causing Simulator #1 to schedule a change at one second from now (when it will execute the two received jobs). In addition, Simulators #2 and #3 schedule a change at two seconds from now (when they will produce their next jobs).

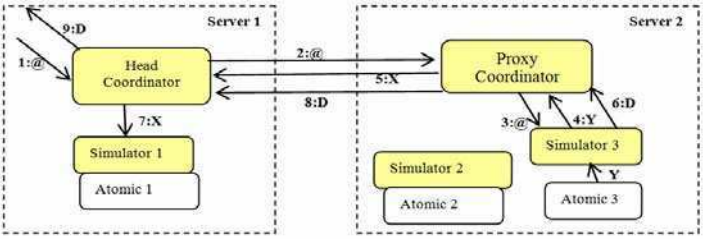


Fig. 17. Head/Proxy Remote messages Transmission

Figure 18 shows two types of messages: Remote and local. All exchanged messages between the Head and Proxy coordinator are remote messages (shown in red); hence, they are usually measured in range of milliseconds to seconds. On the other hand, all other messages are local (shown in blue); hence, they are measured in few microseconds in DCD++, since a processor simply sends a message by inserting it in the unprocessed events queue. To reduce the communication high cost, the DCD++ groups simultaneous events heading to the same destination in one message. For example, as shown in Figure 18-B, the proxy sends two Y messages and D messages for the cost of one message. This

shows huge improvement in performance, particularly, for models with intensive communication overhead. Grouping remote simultaneous events make sense for obvious performance reasons, but also avoid inaccurate simulation results or deadlock in the simulation. This is because P-DEVS messages, as previously mentioned, belong into two categories: (1) Content messages (i.e. Y and X) represent DEVS models communication. These messages must be exchanged within a simulation phase. (2) Synchronization Messages (I, @, * and D) synchronize the start or an end of simulation phase; hence, they mark simulation phases boundaries. Therefore, Content messages must arrive at destination within the correct simulation phase to be executed at the correct virtual time. Further, synchronization messages must arrive at the start or end of the correct simulation phase to ensure correct simulation and to avoid deadlock, since a Coordinator may hang forever waiting for a synchronization message to be able to start a new phase or end the current phase. Of course, we can never guarantee message arrival at destination in the same order of their transmission order. On the other hand, DCD++ guarantees the correct message-order arrival by grouping messages in one XML document, as shown Figure 19.

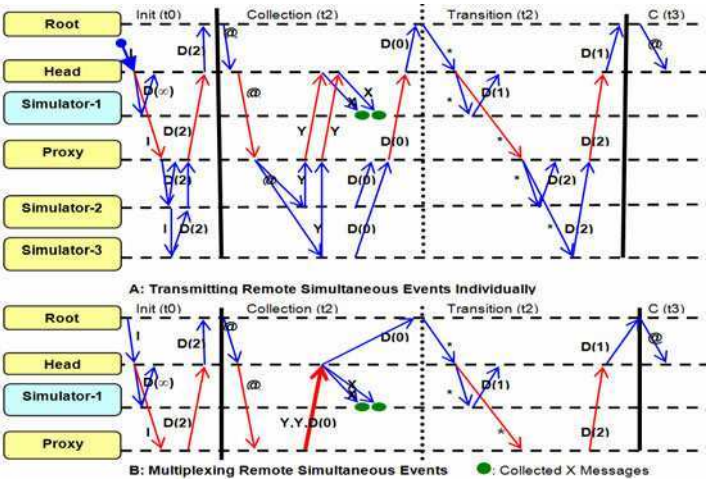


Fig. 18. DCD++ Simulation Phases and Time Advancement

```
<Messages>
   <MessagesCount>2</MessagesCount>
   <Message>
   <MessageType>X</MessageType>
      <Time>08:50:00:00</Time>
      <SrcProcId>2</SrcProcId>
      <PortId>5</PortId>
      <Value>9.0</Value>
      <SenderModelId>3</SenderModelId>
      <DestProcId>1</DestProcId>
   </Message>
   <Message>
      <MessageType>D</MessageType>
      <Time>08:50:00:00</Time>
      <SrcProcId>2</SrcProcId>
      <NextChange>00:00:00:00</NextChange>
      <SenderModelId>3</SenderModelId>
      <Proxy>True</Proxy>
      <DestProcId>1</DestProcId>
   </Message>
</Messages>
```

Fig. 19. Multiplexing Simultaneous Simulation Messages Together

The simulation message contains (at least) the following information (see Figure 19): Message type, simulation time, source processor Id, destination port Id, content value, next change time, sender model Id, and destination Processor Id. DCD++ keeps unique IDs for each model, port and processor (i.e. coupled model coordinator or atomic model simulator) in the DCD++ grid. In this case, simulation managers always organize messages in the order they received from the DCD++ engine, allowing them to be handled in the correct order upon arrival at destination. Simultaneous messages aggregation is accomplished by having message bags in simulation managers to hold content messages to remote processors where those messages are sent with the first synchronization message (i.e. indicates the start/end of a phase) heading to the same processor, according to the shown algorithm in Figure 20. Message aggregation shows clearly that XML message-oriented semantics is much flexible to handle than procedure parameters semantics as in the RPC-style approaches.

```
SendRemoteMessage () {
   If (Remote Processor does not have a message bag) {
      Start Msg bag for Remote Processor;
   }
   Insert Msg in Message bag;
   If (this is a Synchronized Msg) {
      Start XML Document Builder;
      Pack Msgs count in XML Document;
      For (all messages in the Processor's bag) {
         Pack Msg in XML Document;
      }
      Close XML Document;
      Release Processor's Message bag;
      Send XML Document to remote URI;
   }
}
```

Fig. 20. Dispatching Simulation Messages in Single XML Document

5 Distributed Simulation Interoperability Standards

The need for a widely accepted standardized framework is growing necessity nowadays, allowing sharing and reusability across organizations, laboratories and research teams. On the other hand, the specialization of knowledge and fragmentation in the distributed simulation field has also grown than ever. This caused the DEVS simulation community to start the interoperability standardization effort to interoperate various DEVS-based simulation packages together (e.g. CD++ [34]). DEVS standard proposals [30][31][32][33] categorized the standards into two parts: (1) Standardizing DEVS model representation allows a platform-independent DEVS model representation so that it can be executed by a DEVS-based simulator. In this case, a model may be retrieved and executed locally without the need to perform distributed simulation for obvious performance reasons. (2) Since, it is not always possible to run simulation locally on single or multiprocessor machine, the second part deals with Standardizing Interoperability Middleware protocol for interfacing different simulation environments allowing synchronization for the same simulation run across a distributed network regardless of their model representation, as shown in Figure 21. The second part is handled by the distributed simulation middleware, hence our presented topic here. The basic requirements of the interoperability standards are to allow legacy systems to run

their specific model representations, practical software changes (i.e. wrapper to translate messages from/to standardized protocol, see Figure 21), flexible for improvements, independent of any formalism or technology.

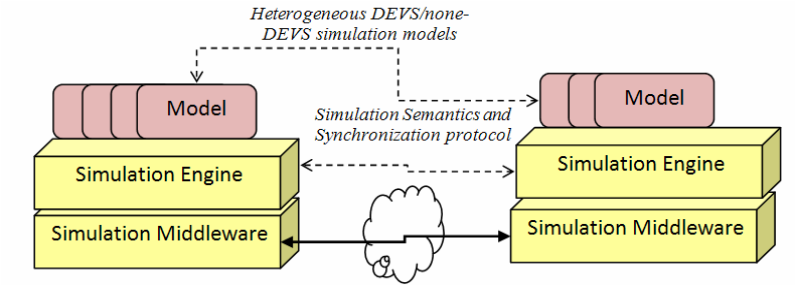


Fig. 21. Concept of Standardized Distributed simulation Middleware

Plugging simulation components into RISE, enabling them to be online, hence accessed via URIs on the Web is one objective of RISE. Further, those components may need to synchronize between each other to simulate a single model within the same simulation model, hence distributed simulation session. In this case, distributed simulation synchronization is still under one team control. Thus, protocols can be customized as needed for specific simulation environment similar to DCD++ previously discussed here. On the other hand, having different independent-developed simulation environment synchronize between each other is another story of complexity. The main complex issue is to bring different teams agree on an interoperability standard. In reality, people do not support standards that require software changes that might affect an existing implementation. The preferred solution is usually by having a wrapper that translates standards from/to local messages. Further, programmers, in practice, do not like to read complex standards, particularly when they are simply evaluating standard proposals without being forced to use it. The lesson learned of the process of having DEVS interoperability standards is that standards should be simple and quickly to understand, fast to support, and without software changes to existing systems. RISE-based standards, presented here, uses the RESTful Web-services plug-and-play and dynamic interoperability style to overcome these issues. The RISE-based proposal details are described in [3] where we discussed all of the submitted proposals by the DEVS community in [30][31][32][33]. The following summarizes the RISE-based proposals in conjunction with the needed wrappers for both CD++ [34] and DEVS/SOA [33] that allow both environments to interoperate.

The RISE-based standards [3][30][31][32][33] divides the entire simulation space into domains. Each domain wraps a DEVS model and DEVS-based simulation engine to simulate that model. Each domain is accessed via three URIs (i.e. the wrapper API in Figure 21) to exchange semantics (i.e. synchronization and configuration) as standardized XML messages. The wrapper API (i.e. URIs) is created at runtime for each experiment setup. The standards completely hide interior implementation domain, avoiding software changes in existing implementation. For example,

RESTful DCD++ performs distributed simulation while DEVS/SOA uses DEVSJAVA [11] engine to perform distributed simulation using SOAP-based WS.

Interoperability is achieved at three levels: (1) the interoperability framework architecture level (API), (2) The model interoperability level, and (3) the simulation synchronization level. These aspects are summarized next.

The interoperability framework architecture level (API) provides the URI template that allows modelers to setup experiment resources across the network, as shown in Figure 22. These resources (URIs) are described as follows: (1) …/{framework}: represents a simulation environment domain. It is named by the modeler upon creation. The modeler uses this URI to submit all necessary information, including RISE XML configuration. (2) …/{framework}/simulation: represents active simulation in a domain, hence used by other domains to exchange simulation messages to synchronize a simulation session. The modeler further uses this URI to start/abort simulation, and to manipulate simulation during runtime or to retrieve online results while simulation is in progress. (3) …/{framework}/results: is automatically created by a domain upon completing the simulation successfully, maintaining simulation results and future results retrieval.

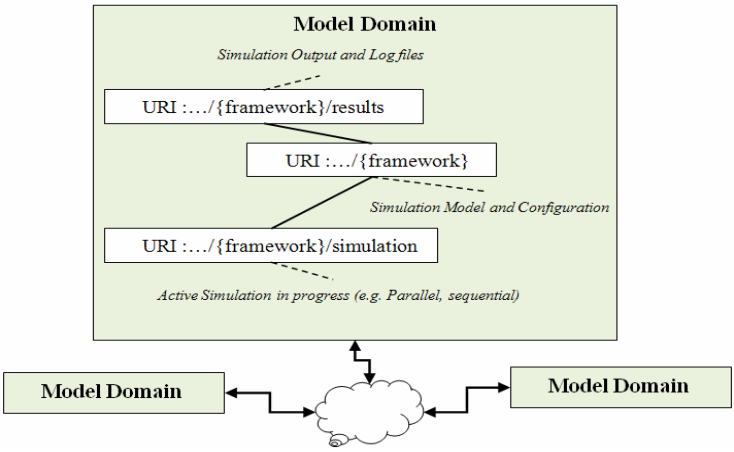


Fig. 22. A Domain Wrapper Application Programming Interface (API)

The model interoperability level provides XML rules for binding different models together. This XML document is provided via PUT channel to URI …/{framework} as part of its initial configuration before a simulation is conducted. However, any dynamic changes during runtime are submitted to URI …/{framework}/simulation. This is mainly when a domain joins/disjoins a simulation session at runtime.

Connecting models across domains is a straightforward step, because of our assumption that each domain contains an entire model with external ports. For example, Figure 23 shows two models placed at two different domains. In this case, the model is wrapped in URI …/{framework}: The first model URI is …/Domain1 and the second model URI is …/Domain2. Each model, in Figure 23, has two external ports connected to the other model ports. This interconnection is shown in the XML

document in Figure 24. For example, Lines 7-10 shows the connection link of port OUT1 (at …/Domain1) to port IN1 (at …/Domain2). The XML document also shows other configuration such as "Type" at Line 3 is set to "O", indicating that the simulation will be synchronized according RISE conservative based algorithm; hence, "Type" attribute can be set to "O" to conduct optimistic synchronization. Line #5 selects the main domain, which is mainly needed to manage the conservative-based simulation. Based on this document Figure 24, each domain needs to build a routing table to identify each of its output port connections so that messages can be transmitted to their destination.

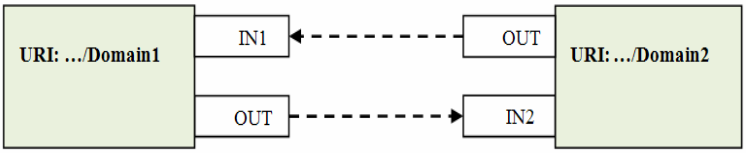


Fig. 23. Models Interconnection across Domains

```
1 <ConfigFramework>
2     …
3     <RISE Version="1.0" Type="C">
4      <Domains>
5       <Main><URI>…/Domain1</URI></Main>
6        <Links>
7         <Link>
8            <From><Port>OUT1</Port><URI>…/Domain1</URI></From>
9            <TO><Port>IN2</Port><URI>…/Domain2</URI></TO>
10        </Link>
11        <Link>
12           <From><Port>OUT2</Port><URI>…/Domain2</URI></From>
13           <TO><Port>IN1</Port><URI>…/Domain1</URI></TO>
14        </Link>
15       </Links>
16      </Domains>
17    </RISE>
18    …
19 </ConfigFramework>
```

Fig. 24. Model Interconnection XML Configuration

The simulation synchronization level provides high-level simulation algorithms (i.e. conservative/optimistic) and synchronization channels in order to carry simulation among different domains. In the optimistic type, XML synchronization messages are sent directly to other domains, since domains should be able to detect errors (e.g. due to straggler messages) and fix them. On the other hand, the conservative type needs to place a Time-Management component (e.g. called here RISE-TM) to synchronize all participants to satisfy local causality constraint via ensuring safe timestamp-ordered processing of simulation events within each domain. Our focus here is on the conservative-type algorithms, since it involves more work from the standards perspective.

RISE-TM executes a simulation cycle in the following steps, as shown in Figure 25: (1) Execute all events in all domains at current time. This starts a new simulation cycle with current or newly calculated RISE time. RISE-TM always starts the first phase with time zero. The domains must always execute all events with

current RISE time, if any, and respond to the RISE-TM with the following information: all external messages generated for other domains stamped with RISE time (or larger), and its next time. The next time is the time of next event in a domain larger than RISE time. (2) Once RISE-TM receives all replies from relevant domains, it calculates the next RISE time and starts a new simulation cycle.

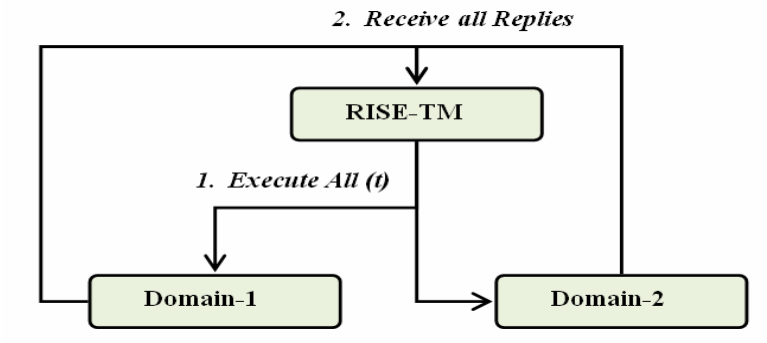


Fig. 25. RISE Conservative-based Simulation Cycle at Time t

```
1 <RISE Version="1.0">
2  <Time>00:00:01:000</Time>
3  <XEvents>
4     <MessagesCount>1</MessagesCount>
5        <XEvent>
6           <Time>00:00:01:000</Time>
7           <Port>IN1</Port>
8           <Value>9</Value>
9           <URI>.../Domain1</URI>
10        </XEvent>
17  </XEvents>
18 </RISE>
```

(A) RISE-TM to Domains XML Request Message

```
1 <RISE Version="1.0">
2 <URI>.../Domain2</URI>
3  <XEvents>
4     <MessagesCount>2</MessagesCount>
5        <XEvent>
6           <Time>00:00:01:000</Time>
7           <Port>IN1</Port>
8           <Value>9</Value>
9           <URI>.../Domain1</URI>
10        </XEvent>
11        <XEvent>
12           ... ... ...
13        </XEvent>
14     <Time>00:00:01:000</Time>
15  </XEvents>
16  <Next>00:00:03:000</Next>
17 </RISE>
```

(B) A Domain to RISE-TM XML Response Message

Fig. 26. RISE-TM and Domains Exchanged Messages Example

Figure 26-A shows an example of messages sent by RISE-TM to a domain (i.e. step #1 in Figure 25). Line #2, in Figure 26-A, specifies the current RISE time, hence every event with this time, in this domain, must be executed in this cycle. Lines 3-17 enclose all collected external messages from all other domains, if any. Figure 26-B shows an example of a domain reply to RISE-TM (i.e. step #2 in Figure 25).

Line #2, in Figure 26-B, indicates the URI of the source domain. Lines 3-15 enclose all of this domain generated external messages to other domains. Line #4 specifies the count of enclosed messages. Lines 5-10 define the first external message. Line #6 specifies the execution time of this message. Line #7 specifies the model destination port (see Figure 23 and Figure 24). Line #8 specifies the message content. Line #9 indicates the destination domain (see Figure 23 and Figure 24). Line #14 specifies the minimum time of all enclosed external messages. RISE-TM must include this time when calculating next RISE time. Line #16 specifies the time of the next event of that domain. RISE-TM must include this time when calculating next RISE time. Further, it is recommended that RISE-TM does not include domains in the

next simulation cycle if they have nothing to do. Note that this value must be set to "-1", indicating infinity, if there is no more events in that domain. This XML document guarantees that all of the domain events stamped with RISE time have executed. This guarantee must be ensured by the RISE-TM by ensuring that the "next" event time (i.e. Line #16 shown in Figure 26-B) is larger than the current RISE time, since it is the time of the next event in a domain. Therefore, domains must only respond once with this XML document.

This method simplifies the synchronization protocol to avoid impractical software changes for a simulation package implementation. It also intended to handle synchronization between DEVS to None-DEVS simulation environments, since it hides all details behind wrappers, including DEVS formalism. The following discusses the changes require to interoperate DCD++ and DEVS/SOA simulation environments to conduct single simulation session. We focus here is on the simulation synchronization level of the standard.

In DCD++, the Simulation manager (see Figure 7) on the main server is the RISE wrapper (Figure 21) of the entire DCD++ domain. It is worth to note that other supportive DCD++ machines are not even aware of being part of a session bridged to another heterogeneous simulation environment. The simulation manager of the main server is extended to act as RISE-TM (i.e. the coordinator of all heterogeneous domains), or as a domain wrapper (i.e. it is being coordinated by other heterogeneous domain), as shown in Figure 25. Therefore, the main simulation manager handles exchanged messages between DCD++ machines according to its specific algorithms, while treat RISE messages according to the standards. Thus, the main DCD++ modifications is in adding new synchronization level between the main simulation manager and its associated CD++ engine. This is done in three parts: (1) having the CD++ engine forward all Y (i.e. output) messages that is intended to other domains to the simulation manager. These are the Y messages received by the Root coordinator (see Figure 18). Regular CD++ considers those messages as output to the environment. (2) Having simulation-manager forward all X messages, received from other domains, to its associated CD++ engine. (3) The CD++ needs to ask the simulation-manager permission before advancing the simulation clock beyond RISE time upon starting new simulation phase (see Figure 18). These parts are discussed in the next paragraphs.

First, the CD++ Root coordinator forwards all Y (i.e. output) messages to its associated simulation manager. This message also includes the simulation timestamp and the model source port. Note that the CD++ does not know where those messages need to be sent; hence, it treats them as output messages to the simulation environment. At this point, the simulation-manager converts those Y messages into X (i.e. external) messages and stores them so that they can be transmitted altogether in single XML document. The simulation-manager also needs to add the destination port and URI. This is easily done based on routing tables constructed based on the configuration document (Figure 24). For example, Y messages received from port OUT1 in Domain-1, at Figure 23, need to be routed to port IN2 of domain-2. Once those messages need to be transmitted, the simulation-manager builds the XML message, shown in Figure 26-B, and sends them to RISE-TM. However, if this simulation manager is the acting as the RISE-TM, it merges them with other domains messages, if any, and sends them back to relevant domains, as shown in Figure 26-A.

Note that this special treatment is only for RISE messages, but messages belong to the DCD++ region need to be handled according to its specific algorithms.

Second, the simulation-manger needs to filter its domain X messages upon their arrival from other domains, and forwards them to the CD++. The simulation manager receives them as the message shown in Figure 26-B, if it is the acting RISE-TM while receives as the message shown in Figure 26-A, if it is not the acting RISE-TM. Subsequently, the CD++ saves them in special queue until the beginning of next simulation cycle where Root coordinator will insert them in the simulation event list similar to any other local events.

Third, the CD++ needs to consult the simulation manager before advancing to new cycle (Figure 18). The entire DCD++ simulation cycles are driven by the Root coordinator, specifically, upon a DONE message arrival, as described in Figure 14. In this case, the Root checks DCD++ next event time against last known RISE time, it then proceeds if they are equal to each other. Otherwise, (1) it requires RISE Time update from simulation manager, (2) Insert any received external messages from other domains into the simulation event list, (3) calculate next event time, and (4) report next time to simulation manager. Based on the next event time and the current RISE time, the simulation manager knows the end of the current simulation cycle. These steps are handled in the following algorithm:

```
While (RISE Time != DCD++ Next Time) {
        Get RISE Time from Simulation Manager;
        Insert Other Domains Collected X messages;
        Calculate new DCD++ Next Time;
        Report Next Time to Simulation manager;
}
```

DEVS/SOA [33] uses DEVSJAVA [11] simulation engine to perform distributed simulation using SOAP-based WS. As illustrated in Figure 27, the DEVS/SOA protocol is executed as following (shown in Figure 27): (Step #1 and #2) the highest coordinator (i.e. Root) requests the next event time of each of its children simulators and coordinators. Messages nextTN and outTN are performed in a single RPC invocation. (2) The Root requests each of its children to compute its output messages to other simulators (i.e. getOut and outTN). (3) Finally, each simulator executes its ApplyDeltFunc method, which computes the combined effect of the received messages and internal scheduling on its state.

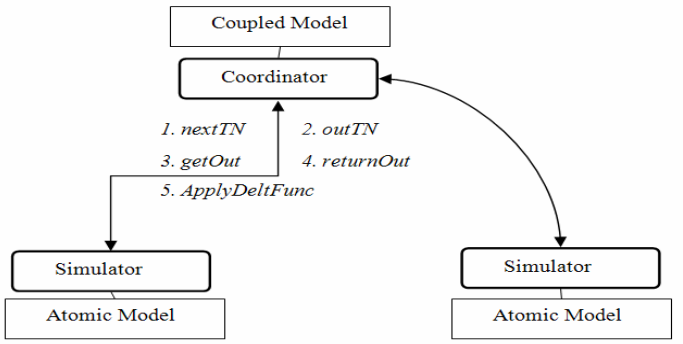


Fig. 27. DEVS/SOA Distributed Simulation Protocol

DEVS/SOA needs to have wrapper (see Figure 21) to translate internal DEVS/SOA commands, shown in Figure 27, into RISE messages. This wrapper, similar to DCD++ simulation manager, needs to exchange RISE XML messages in HTTP envelopes. Further, the Root coordinator should not advance beyond current RISE time. The major requirements of this wrapper is to translate DEVS/SOA RPC internal commands into RISE XML messages (Figure 26) and vice versa, as follows.

RISE XML message (Figure 27-A) corresponds to DEVS/SOA "nextTN", "getOut", "ApplyDelta" calls. Upon this message arrival from RISE-TM, all DEVS/SOA simulators must execute all internal/external events at this cycle time (i.e. element <Time>). Further, RISE-TM forwards previous output messages from previous cycles, if any, in this message.

RISE XML message (Figure 27-B) corresponds to DEVS/SOA "returnOut" and "OutTN" calls: (1) returnOut (i.e. output message) is RISE external message, defined in element <XEvent>. (2) OutTN (i.e. next time) defined in element <Time>.

References

[1] Al-Zoubi, K., Wainer, G.: Performing Distributed Simulation with RESTful Web-Services Approach. In: Proceedings of the Winter Simulation Conference (WSC 2009), Austin, TX, pp. 1323–1334 (2009)

[2] Al-Zoubi, K., Wainer, G.: Using REST Web Services Architecture for Distributed Simulation. In: Proceedings of Principles of Advanced and Distributed Simulation (PADS 2009), Lake Placid, New York, USA, pp. 114–121 (2009)

[3] Al-Zoubi, K., Wainer, G.: RISE: REST-ing Heterogeneous Simulation Interoperability. In: Proceedings of the Winter Simulation Conference (WSC 2010), Baltimore, Maryland, USA (2010)

[4] Al-Zoubi, K., Wainer, G.: Managing Simulation Workflow Patterns using Dynamic Service-Oriented. In: Proceedings of the Winter Simulation Conference (WSC 2010), Baltimore, Maryland, USA (2010)

[5] Amazon Web-services: Security Best Practices (2010), `http://awsmedia.s3.amazonaws.com/Whitepaper_Security_Best_Practices_2010.pdf` (accessed June 2010)

[6] Boer, C., Bruin, A., Verbraeck, A.: A survey on distributed simulation in industry. Journal of Simulation 3(1), 3–16 (2009)

[7] Boukerche, A., Zhang, M., Xie, H.: An Efficient Time Management Scheme for Large-Scale Distributed Simulation Based on JXTA Peer-to-Peer Network. In: Proceedings of the IEEE/ACM Distributed Simulation and Real-Time Applications (DS-RT 2008), Vancouver, BC, Canada (2008)

[8] Bryant, R.E.: Simulation of packet communication architecture computer systems. Massachusetts Institute of Technology, Cambridge (1977)

[9] Chandy, K.M., Misra, J.: Distributed Simulation: A Case Study in Design and Verification of Distributed. Programs. IEEE Transactions on Software Engineering SE-5(5), 440–452 (1979)

[10] Cheon, S., Seo, C., Park, S., Zeigler, B.P.: Design and Implementation of Distributed DEVS Simulation in a Peer to Peer Network System. In: Proceedings of the Advanced Simulation Technologies Conference, Arlington Virginia (2004)

[11] DEVSJAVA, http://www.acims.arizona.edu/SOFTWARE/software.shtml (accessed June 2010)
[12] Erl, T., Karmarkar, A., Walmsley, P., Haas, H., Yalcinalp, L.U., Liu, K., Orchard, D., Tost, A., Pasley, J.: Web Service Contract Design and Versioning for SOA. Prentice-Hall, Englewood Cliffs (2008)
[13] Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine (2000), http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (accessed October 2008)
[14] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, http://www.w3.org/Protocols/rfc2616/rfc2616.html (accessed October 2008)
[15] Fujimoto, R.M.: Parallel and distribution simulation systems. John Wiley & Sons, New York (2000)
[16] Gregorio, J.: URI Templates, http://bitworking.org/projects/URI-Templates/ (accessed October 2008)
[17] Henning, M.: The Rise and Fall of CORBA. Communications of the ACM 51(8) (August 2008), http://queue.acm.org/detail.cfm?id=1142044 (accessed March 2010)
[18] Henning, M., Vinoski, S.: Advanced CORBA programming with C++. Addison–Wesley, Reading (1999)
[19] IBM Mashup Center, http://www-01.ibm.com/software/info/mashup-center/ (accessed June 2009)
[20] IBM Software Group: Why Mashups Matter, ftp://ftp.software.ibm.com/software/lotus/lotusweb/portal/why_mashups_matter.pdf (accessed June 2009)
[21] Khul, F., Weatherly, R., Dahmann, J.: Creating Computer Simulation Systems: An Introduction to High Level Architecture. Prentice-Hall, Englewood Cliffs (1999)
[22] Mandel, L.: Describe REST Web services with WSDL 2.0, http://www.ibm.com/developerworks/webservices/library/ws-restwsdl/ (accessed May 2009)
[23] Mittal, S., Risco-Martín, J.L., Zeigler, B.P.: DEVS-based simulation web services for net-centric T\&E. In: Proceedings of the 2007 Summer Computer Simulation Conference, San Diego, California, USA (2007)
[24] NetBeans IDE, http://www.netbeans.org/ (accessed June 2009)
[25] O'Reilly, T.: What Is Web 2.0 (2005), http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html (accessed May 2009)
[26] Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media, Inc., Sebastopol (2007)
[27] Shahbazian, E.: Introduction to DF: Models and Processes, Architectures, Techniques and Applications. In: Multisensor Fusion, pp. 71–97. Kluwer Academic Publishers, Dordrecht (2000)

[28] Strassburger, S., Schulze, T., Fujimoto, R.: Future trends in distributed simulation and distributed virtual environments: results of a peer study. In: Proceedings of Winter Simulation Conference (WSC 2008), Miami, FL, pp. 777–785 (2008)
[29] Wainer, G., Madhoun, R., Al-Zoubi, K.: Distributed Simulation of DEVS and Cell-DEVS Models in CD++ using Web Services. Simulation Modelling Practice and Theory 16(9), 1266–1292 (2008)
[30] Wainer, G., Al-Zoubi, K., Mittal, S., Risco Martín, J.L., Sarjoughian, H., Zeigler, B.P.: DEVS Standardization: Foundations and Trends. In: Wainer, G., Mosterman, P. (eds.) Discrete-Event Modeling and Simulation: Theory and Applications, ch. 15. CRC Press, Taylor and Francis (October 2010) (expected publication)
[31] Wainer, G., Al-Zoubi, K., Mittal, S., Risco Martín, J.L., Sarjoughian, H., Zeigler, B.P.: An Introduction to DEVS Standardization. In: Wainer, G., Mosterman, P. (eds.) Discrete-Event Modeling and Simulation: Theory and Applications. CRC Press, Taylor and Francis (October 2010) (expected publication)
[32] Wainer, G., Al-Zoubi, K., Mittal, S., Risco Martín, J.L., Sarjoughian, H., Zeigler, B.P.: Standardizing DEVS Model Representation. In: Wainer, G., Mosterman, P. (eds.) Discrete-Event Modeling and Simulation: Theory and Applications, ch. 17. CRC Press, Taylor and Francis (October 2010) (expected publication)
[33] Wainer, G., Al-Zoubi, K., Mittal, S., Risco Martín, J.L., Sarjoughian, H., Zeigler, B.P.: Standardizing DEVS Simulation Middleware. In: Wainer, G., Mosterman, P. (eds.) Discrete-Event Modeling and Simulation: Theory and Applications, ch. 18. CRC Press, Taylor and Francis (October 2010) (expected publication)
[34] Wainer, G.: Discrete-Event Modeling and Simulation: A Practitioner's Approach. CRC press, Taylor & Francis Group, Boca Raton, Florida (2009)
[35] Wainer, G., Al-Zoubi, K.: An Introduction to Distributed Simulation. In: Banks, C., Sokolowski, J. (eds.) Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domains, ch. X. Wiley, New Jersey (2010)
[36] Web Services AXIS, `http://ws.apache.org/axis/` (accessed October 2008)
[37] Web Application Description Language (WADL), `https://wadl.dev.java.net/` (accessed October 2008)
[38] WSDL 2.0: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, `http://www.w3.org/TR/wsdl20/` (accessed July 2010)
[39] Zeigler, B.P., Doohwan, K.: Distributed supply chain simulation in a DEVS/CORBA execution environment. In: Proceedings of Winter Simulation Conference (WSC 1999), Phoenix, Arizona, USA (December 1999)

Chapter 7
Agile Net-Centric Systems Using DEVS Unified Process

Saurabh Mittal

DUNIP Technologies
PO Box 26218, Tempe AZ 85285 USA
saurabh.mittal@duniptechnologies.com
http://www.duniptechnologies.com

Abstract. Industry and government are spending extensively to transition their business processes and governance to Service Oriented Architecture (SOA) implementations for efficient information reuse, integration, collaboration and cost-sharing. SOA enables orchestrating web services to execute such processes using Business Process Execution Language (BPEL). Business Process Modeling Notation (BPMN) is another method that outputs BPEL for deployment. As an example, the Department of Defenses (DoD) grand vision is the Global Information Grid that is founded on SOA infrastructure. The SOA infrastructure is to be based on a small set of capabilities known as Core Enterprise Services (CES) whose use is mandated to enable interoperability and increased information sharing within and across Mission Areas, such as the Warfighter domain, Business processes, Defense Intelligence, and so on. Net-Centric Enterprise Services (NCES) is DoDs implementation of its Data Strategy over the GIG. However, composing/orchestrating web services in a process workflow (a.k.a Mission thread in the DoD domain) is currently bounded by the BPMN/BPEL technologies. With so much resting on SOA, their reliability and analysis must be rigorously considered. The BPMN/BPEL combination neither has any grounding in system theoretical principles nor can it be used in designing net-centric systems based on SOA in its current state. In this work we present a system theoretical framework using the DEVS Unified Process (DUNIP) that allows bifurcated model-continuity based life cycle process for simultaneous development of the executable system using web-services (including the model) and the automated generation of Test-suite for Verification and Validation. The entire net-centric system, which includes artifacts like the model, the simulation and the real system, is deployed on SOA. The simulation system is made possible on a recently developed DEVS-based service framework called DEVS/SOA. We will show the design of DEVS-agents based on WSDLs and how they are composed towards the systems specification. We will demonstrate how agility is an inherent characteristic of such a system founded on DUNIP. We will also present the case of Department of Defense Architecture Framework (DoDAF) and how agility can be applied to the design and evaluation process.

Keywords: DEVS, DUNIP, DoDAF, SOA, WSWF, NCES, GIG.

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 159–199.
springerlink.com

1 Introduction

Industry and government are spending extensively to transition their business processes and governance to Service Oriented Architecture implementations for efficient information reuse, integration, collaboration and cost-sharing. Service Oriented Architecture (SOA) enables orchestrating web services to execute such processes using Business Process Execution Language (BPEL) [4]. Business Process Modeling Notation (BPMN) [5] is another method that outputs BPEL for deployment. As an example, the Department of Defense's (DoD grand vision is the Global Information Grid that is founded on SOA infrastructure. As illustrated in Figure 1, the SOA infrastructure is to be based on a small set of capabilities known as Core Enterprise Services (CES) whose use is mandated to enable interoperability and increased information sharing within and across Mission Areas, such as the Warfighter domain, Business processes, Defense Intelligence, and so on) [16]. Net-Centric Enterprise Services (NCES) [30] is DoD's implementation of its Data Strategy over the GIG. NCES provide SOA infrastructure capabilities such as service and metadata registries, service discovery, user authentication, machine-to-machine messaging, service management, orchestration, and service governance.

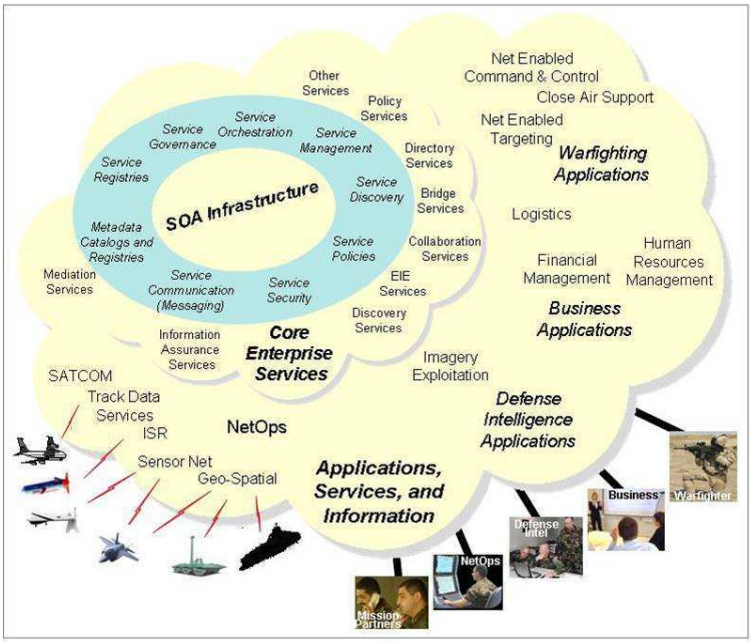


Fig. 1. Core enterprise services in Global Information Grid [16]

However, composing/orchestrating web services in a process workflow (a.k.a Mission thread in the DoD domain) is currently bounded by the BPMN/BPEL technologies. Moreover, there are few methodologies to support such composition/orchestration. Further, BPMN and BPEL are not integrated in a robust manner and different proprietary BPMN diagrams from commercial tools fail to deliver the same BPEL translations. Today, these two technologies are by far the only viable means whereby executives and managers can devise process flows without touching the technological aspects. With so much resting on SOA, their reliability and analysis must be rigorously considered. The BPMN/BPEL combination neither has any grounding in system theoretical principles nor can it be used in designing net-centric systems based on SOA in its current state.

In this research work we provide a proof of concept of how Discrete Event System Specification (DEVS) Formalism can deliver another process work flow mechanism to compose web services in a SOA. A DEVS System is composed of events and components/systems that produce and consume those events. An event is any change in state that merits attention from self/other systems. These systems can be either a simple atomic black box that perform a single task only or they may be a complex system of systems that receive the event and delegate it to one of its sub-components. We will employ DEVS Formalism to a net-centric system deployed using Web Services. Such an architecture where events work along with web services is aptly termed as Service Oriented Architecture (SOA). During this process of designing the net-centric system, we will propose Web Service Work Flow (WSWF) formalism and show how it is executed on the recently developed DEVS/SOA [28] distributed modeling and simulation framework.

In addition to supporting SOA application development, the framework enables verification and validation testing of application. We will also describe how WSWF can be mapped to high level system descriptive frameworks like Department of Defense Architecture Framework (DoDAF) [9],[10],[11], and System Entity Structure (SES). We will demonstrate the execution of WSWF in a complete case-study in which a workflow is composed and executed using DEVS/SOA framework.

Finally, this paper will establish that the DEVS Unified Process inherently is agile and that when deployed on SOA makes it a truly interoperable and testable framework. The paper is organized as follows. Section 2 presents the related technologies. Section 3 describes the underlying technologies that include DEVS, DUNIP, Web Services, Abstract DEVS Service Agent, and DEVS/SOA framework. Section 4 presents layered architecture of Agent-based Test Instrumentation System on/using Global Information Grid using SOA (GIG/SOA). Section 5 deals with Abstract DEVS Service wrapper in detail and also discusses how statistics gathering is integrated with the wrapper design. Section 6 presents the workflow composition and how high-level specifications, as specified by frameworks like DoDAF, can be reduced to WSWF formalism. It is discussed using ontology based System Entity Structure (SES) framework that is targeted to modeling, simulation, systems design and engineering. Section 7 presents a

complete case study demonstrating the usage of WSWF. Section 8 presents some ideas on agility inherent in the DEVS Unified Process. Finally, Section 9 lists conclusions and future work.

2 Related Technologies

In 2003 there were more than 10 recognized groups defining standards for BPM related activities. 7 of these bodies were working on modeling definitions so its no wonder that the whole picture got very confused [31]. Fortunately there has been a lot of consolidation, and currently only 3 key standards to really take notice:

1. BPMN
2. XPDL
3. BPEL

The Business Process Modeling Notation (BPMN) is a standardized graphical notation for graphically representing business processes workflows in a Business Process Diagram (BPD). The BPD is based on a flowcharting technique that is similar to UML Activity diagrams. BPMNs primary goal is to provide a standard notation that is readily understandable by all business stakeholders. Stakeholders in this definition include business analysts, technical developers and business managers. BPMN is primarily constrained to support only the concepts related to business processes. Consequently, there is no support for modeling organizational structure, hierarchical functional breakdowns, data schemas and various other sorts of mapping that are needed for a systems specification. Needless to say, apart from the BPMN graphical elements that are visually appealing and easier to communicate among business users, it provides limited tangible system requirements that are hard to trace back to the constituent systems. With the advent of SOA, these business processes have taken the shape of services. While it shows the orchestration of services, it does not mandate any execution platform or testing platform to build a system. It works in conjunction with Business Process Execution Language (BPEL), that is a standard in itself, to deliver executable code for mockups and concept validation. This is a needed feature, however plagued with lack of roundtrip engineering, mismatch between transformations from BPMN to BPEL and vice-versa, ambiguities and confusion with multiple tool vendors leading to isolated BPMN models.

BPEL is an "execution language" the goal of which is to enable definition of web service orchestrations. It's actually an acronym for Web Services Business Process Execution Language (WS-BPEL) and has no standard graphical notation. Ultimately, BPEL is all about bits and bytes being moved from place to place and manipulated. BPEL code is normally generated from BPMN specification as BPMN is the overarching specification that uses BPEL as its execution platform. The fundamental difference between the BPMN and BPEL specifications makes it very difficult to generate human-readable BPEL code.

The XML Process Definition Language (XPDL) is a format to interchange business process definitions between different workflow products. It defines an XML Schema for specifying the declarative part of workflow/business processes. It is described not an executable programming language like BPEL, but specifically a process design format that literally represents the "drawing" of the process definition, such as the X and Y position of the nodes. XPDL is effectively the file format or "serialization" of BPMN. More generally, it can also support any design method or process model that uses the XPDL meta-model. XPDL is a proven format for process design interchange, and it is the most practical standard for establishing a Process Design Ecosystem.

BPEL has largely been promoted by tool vendors and articles like [57], [58] argue that shortcomings of BPEL outweigh its benefits and simple flash tools can be made that could render BPMN using XPDL formats and execute the process model. Summarizing, currently there is no popular means other than BPMN/BPEL to design a web service workflow orchestration and supposedly no system theoretical foundation to build a net-centric system.

3 DEVS Unified Process with DEVS/SOA

3.1 Discrete Event Systems Specification

Discrete Event System Specification (DEVS) [39] is a formalism, which provides a means of specifying the components of a system in a discrete event simulation. In DEVS formalism, one must specify *Basic Models* and how these models are connected together. These basic models are called *Atomic Models* (Figure 2) and larger models which are obtained by connecting these atomic blocks in meaningful fashion are called *Coupled Models* (Figure 3). Each of these atomic models has *inports* (to receive external events), *outports* (to send events), set of *state variables*, *internal transition*, *external transition*, and *time advance functions*. Mathematically it is represented as 8-tuple system:

$$M =< X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta >$$

where

X is the set of input values

S is the set of states

Y is the set of output values

$\delta_{int} : S \rightarrow S$ is the internal transition function

$\delta_{ext} : Q \times X_b \rightarrow S$ is the external transition function,

where $X_b$is a set of bags over elements in X, Q is the total state set

$\delta_{con} : S \times X_b \rightarrow S$ is the confluent transition function,

subject to $\delta_{con}(s, \phi) = \delta_{int}(s)$

$\lambda : S \rightarrow Y_b$ is the output function

$ta : S \rightarrow R_{(0+,inf)}$ is the time advance function

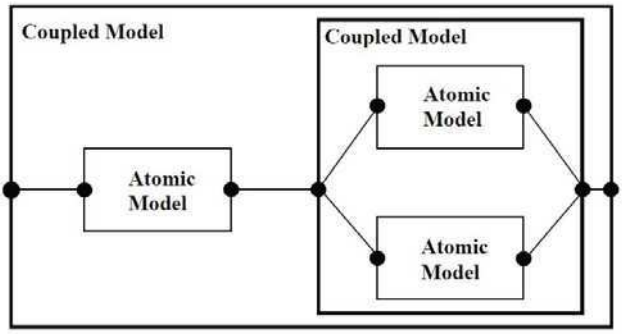


Fig. 2. Hierarchical components at two levels

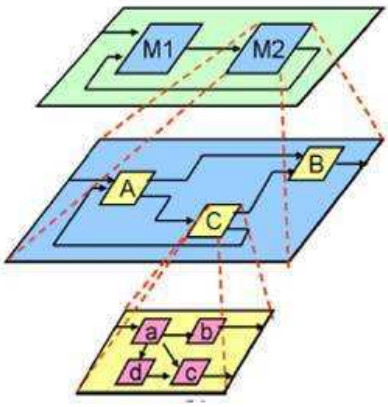


Fig. 3. Hierarchical components for multi-level systems

The models description (implementation) uses (or discards) the message in the event to do the computation and delivers an output message on the outport and makes a state transition. A DEVS-coupled model designates how atomic models can be coupled together and how they interact with each other to form a complex model. The coupled model can be employed as a component in a larger coupled model and can construct complex models in a hierarchical way. The specification provides component and coupling information. The coupled DEVS model is defined as follows.

$$M =< X, Y, D, M_{ij}, I_j, Z_{ij} >$$

where

X is a set of inputs

Y is a set of outputs

D is a set of DEVS component names

for each $i \in D$,
M_i is a DEVS component model
I_i is the set of influences for I
for each $j \in I_i$,
Z_{ij} is the *i-to-j* output translation function.

A Java-based implementation of DEVS formalism, DEVSJAVA [40], can be used to implement these atomic or coupled models. DEVS formalism consists of models, the simulator and the Experimental Frame as show in Figure 4. We will focus our attention to these two types of models i.e. atomic and coupled.

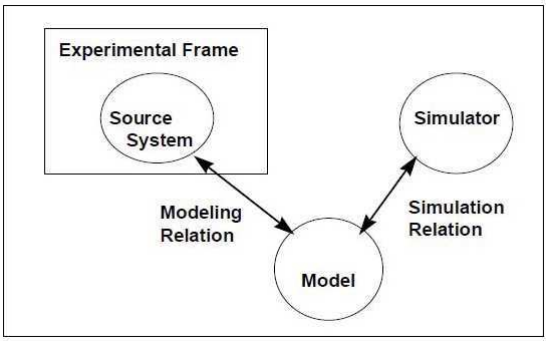


Fig. 4. DEVS separation of the model, the simulation and the Experimental Frame

3.2 Web Services and Interoperability Using XML

The Service oriented Architecture (SOA) framework is a framework consisting of various W3C standards, in which various computational components are made available as services that interact in an automated manner achieving machine-to-machine interoperable interaction over the network. The interface is specified using Web Service Description language (WSDL) [38] that contains information about ports, message types, port types, and other relating information for binding two interactions. It is essentially a client server framework, wherein client requests a service using a SOAP message that is transmitted via HTTP protocol in the XML format. A Web service is published by any commercial vendor at a specific URL is consumed/requested by another commercial application on the Internet. It is designed specifically for machine-to-machine interaction. Both the client and the server encapsulate their messages in SOAP wrappers.

The fundamental concept of web services is to integrate software application as services. Web services allow the applications to communicate with other applications using open standards. To offer DEVS-based simulators as web services, they

must have the following standard technologies: communication protocol (Simple Object Access Protocol, SOAP [35]), service description (Web Service Description Language, WSDL), and service discovery (Universal Description Discovery and Integration, UDDI).

3.3 An Abstract DEVS Service Agent

As a crucial part of our workflow, we have designed an Abstract DEVS Service Agent to link DEVS models with Web Services and to generate statistics regarding remote method calls and response times.

Figure 5 depicts an illustrative example. Our proposed model consists of two DEVS atomic models. The DEVS Web Service Consumer invokes the remote operation provided by means of an external transition. When the operation is processed, this atomic model calculates the round-trip-time (RTT) taken by such operation and directs both the RTT and the received response from the Web Service to the DEVS Logger atomic model. At the end of the simulation, the DEVS Logger provides statistics such as operations executed successfully, the RTT consumed per operation, etc.

The DEVS Web Service Consumer needs to be configured by means of: (a) the URL of the Web Service, (b) name of the operations offered by the web service, and (c) the parameters needed by these operations. This information is specified in the WSDL document. In order to avoid to the user to extract this information by hand, we have implemented a wrapper which automatically generates the DEVS Web Service Consumer for a Web Service. Thus, given a WSDL address, our framework is able to generate the corresponding DEVS Service Agent. Details on how this wrapper is built are given in Section 5.

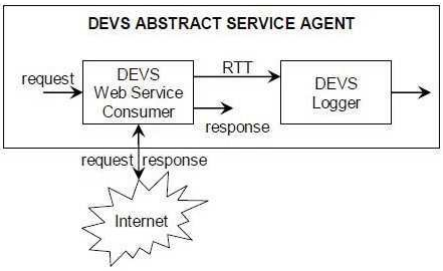


Fig. 5. Schematic showing the architecture of DEVS Agent Service Model

3.4 DEVS/SOA Framework for Net-Centric Modeling and Simulation

DEVS Modeling Language (DEVSML) is a way of representing DEVS models in the XML language [22]. The DEVSML is built on JAVAML [3], which is an XML representation of JAVA. DEVSML takes its power from the underlying JAVAML that is needed to specify the behavior logic of atomic and

coupled models. The DEVSML models are transformable to JAVA in both forward and reverse directions. It is an attempt to provide interoperability between various models and create dynamic scenarios. The layered architecture of this capability is shown in Figure 6. At the top is the application layer that contains model in DEVSJAVA or DEVSML. The second layer is the DEVSML layer itself that provides seamless integration, composition and dynamic scenario construction resulting in portable models in DEVSML that are complete in every respect. These DEVSML models can be ported to any remote location using the net-centric infrastructure and be executed at any remote location. Another major advantage of such capability is total simulator transparency. The simulation engine is totally transparent to model execution over the net-centric infrastructure. The DEVSML model description files in XML contains meta-data information about its compliance with various simulation builds or versions to provide true interoperability between various simulator engine implementations. This has been achieved for at least two independent simulation engines as they have an underlying DEVS protocol to adhere to. This has been made possible with the implementation of a single atomic DTD and a single coupled DTD that validates the DEVSML descriptions generated from these two implementations. Such run-time interoperability provides great advantage when models from different repositories are used to compose bigger coupled models using DEVSML seamless integration capabilities. More details about the implementation can be seen at [22].

The DEVS/SOA framework [28] is analogous to other DEVS distributed simulation frameworks like DEVS/HLA, DEVS/RMI and DEVS/CORBA [32],[13], [36],[7],[17],[41]. The distinguishing mark of DEVS/SOA is that it uses SOA as the network communication platform and XML as the middleware and thus acts as a basis of interoperablity using XML [27]. Furthermore, it uses web-services as the underlying technology to implement the DEVS simulation protocol.

The complete setup requires one or more servers that are capable of running DEVS Simulation Service, as shown in Figure 7. The capability to run the simulation service is provided by the server side design of DEVS Simulation protocol supported by the latest DEVSJAVA Version 3.1 [1].

The numerous modes of DEVS model generation are beyond the scope of this paper (the interested reader is referred to [24]. Once a DEVS model package is developed, the next step is simulation as illustrated in Figure 7.The DEVS/SOA client (Figure 8) takes the DEVS models package and through the dedicated servers hosting DEVS simulation services, it performs the following operations:

- Upload the models to specific IP locations i.e. partitioning (Figure 9)
- Run-time compile at respective sites
- Simulate the coupled-model
- Receive the simulation output at clients end

This section has laid the foundation of net-centric DEVS framework called DEVS/SOA that allows deployment of DEVS models to specific IP addresses

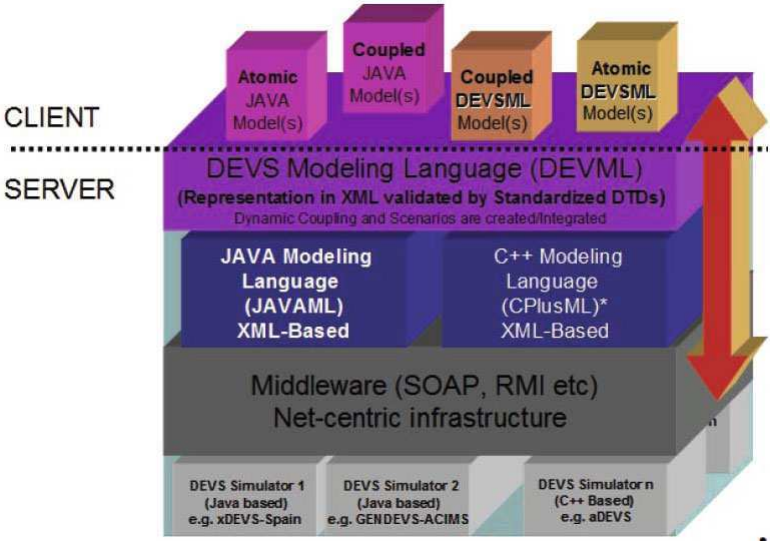


Fig. 6. Layered architecture of DEVSML towards transparent simulators in net-centric domain

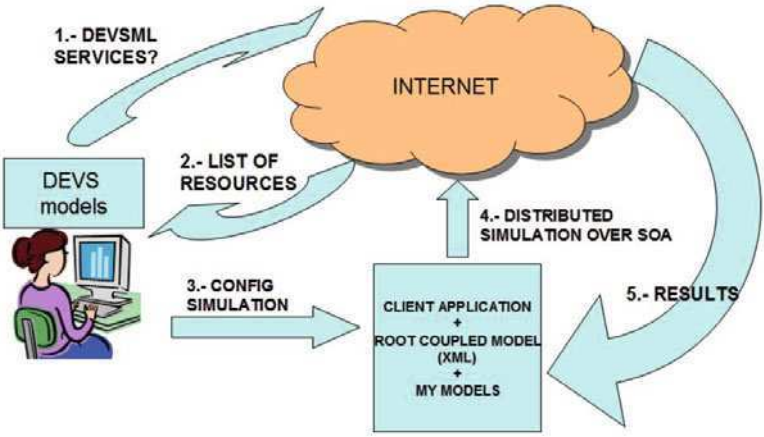


Fig. 7. Execution of DEVS models using DEVS/SOA framework

and allows interoperability between DEVS models using DEVSML. It provides a layered framework in which the models are transparent to their simulators. In the next section we will see how the net-centric DEVS is applicable to testing of Global Information Grid based on Service Oriented Architecture (GIG/SOA). A sample movie of DEVS/SOA in action is available at [34].

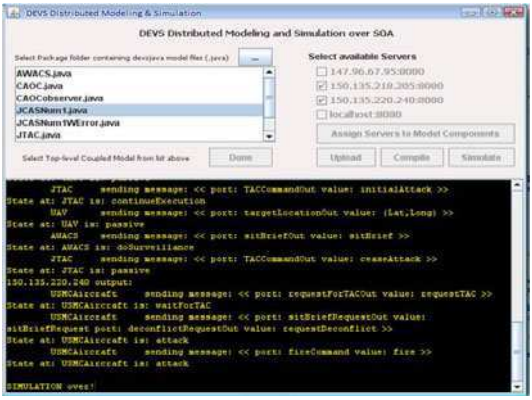


Fig. 8. DEVS/SOA client hosting a distributed simulation

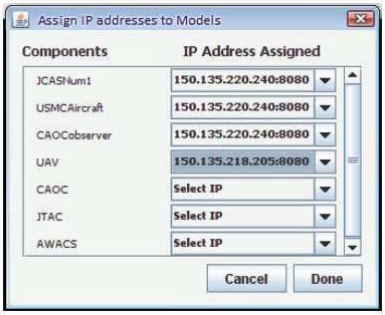


Fig. 9. Server assigned to models using manual model partitioning

3.5 DEVS Unified Process a.k.a DUNIP

This section describes the bifurcated Model-Continuity process [24] and how various elements like automated DEVS model generation, automated test-model generation (and net-centric simulation over SOA are put together in the process, resulting in DEVS Unified Process (DUNIP) [12],[24]. The DEVS Unified Process (DUNIP) is built on the bifurcated Model-continuity based life-cycle methodology. The design of simulation-test framework occurs in parallel with the simulation-model of the system under design. The DUNIP process consists of the following elements:

1. Automated DEVS Model Generation from various requirement specification formats
2. Collaborative model development using DEVS Modeling Language (DEVSML)

3. Automated Generation of Test-suite from DEVS simulation model
4. Net-centric execution of model as well as test-suite over SOA

Considerable amount of effort has been spent in analyzing various forms of requirement specifications, viz, state-based, Natural Language based, UML-based, Rule-based, BPMN/BPEL-based and DoDAF-based, and the automated processes which each one should employ to deliver DEVS hierarchical models and DEVS state machines [15],[24]. Simulation execution today is more than just model execution on a single machine. With Grid applications and collaborative computing the norm in industry as well as in scientific community, a net-centric platform using XML as middleware results in an infrastructure that supports distributed collaboration and model reuse. The infrastructure provides for a platform-free specification language DEVS Modeling Language (DEVSML) [22] and its net-centric execution using Service-Oriented Architecture called DEVS/SOA [23]. Both the DEVSML and DEVS/SOA provide novel approaches to integrate, collaborate and remotely execute models on SOA. This infrastructure supports automated procedures for test-case generation leading to test models.

Using XML as the system specifications in rule-based format, a tool known as Automated Test Case Generator (ATC-Gen) was developed which facilitated the automated development of test models [18],[19],[42]. DUNIP (Figure 10) can be summarized as the sequence of the following steps:

1. Develop the requirement specifications in one of the chosen formats such as BPMN, DoDAF, Natural Language Processing (NLP) based, UML based or simply DEVS-based for those who understand the DEVS formalism.
2. Using the DEVS-based automated model generation process, generate the DEVS atomic and coupled models from the requirement specifications using XML
3. Validate the generated models using DEVS W3C atomic and coupled schemas to make them net-ready capable for collaborative development, if needed. This step is optional but must be executed if distributed model development is needed. The validated models which are Platform Independent Models (PIMs) in XML can participate in collaborative development using DEVSML.
4. From step 2, either the coupled model can be simulated using DEVS/SOA or a test-suite can be generated based on the DEVS models.
5. The simulation can be executed on an isolated machine or in distributed manner (using SOA middleware if the focus is net-centric execution). The simulation can be executed in real-time as well as in logical time.
6. The test-suite generated from DEVS models can be executed in the same manner as laid out in Step 5.
7. The results from Step 5 and Step 6 can be compared for verification and validation process.

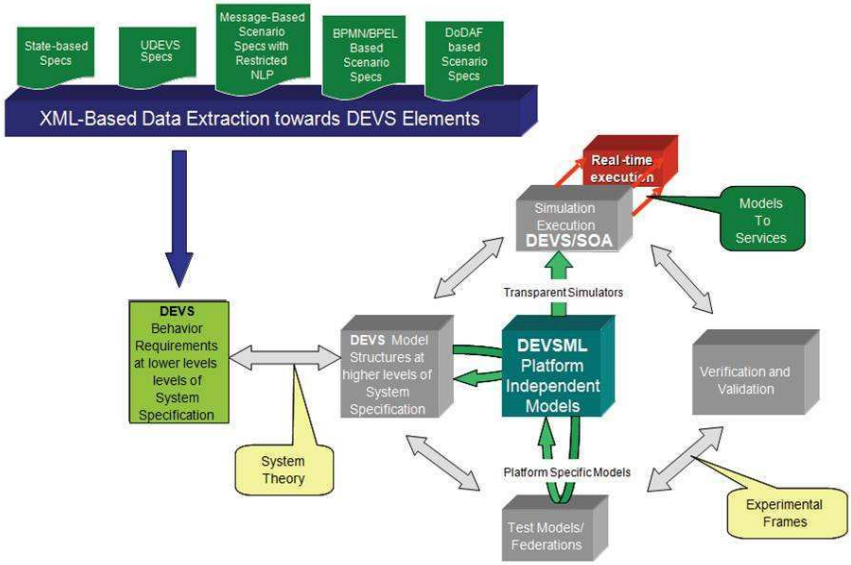


Fig. 10. The complete DEVS Unified Process

4 Multi-layered Agent-Based Test Instrumentation System Using GIG/SOA

A DEVS distributed federation is a DEVS coupled model whose components reside on different network nodes and whose coupling is implemented through middleware connectivity characteristic of the environment, e.g., SOAP for GIG/SOA, The federation models are executed by DEVS simulator nodes that provide the time and data exchange coordination as specified in the DEVS abstract simulator protocol. The DEVS Agent Monitoring System or Test Instrumentation System (TIS) is a DEVS coupled system that observes and evaluates the operation of the DEVS coupled system model. The DEVS models used to observe other participants are the DEVS test-agents. The TIS should provide a minimally intrusive test capability to support rigorous, ongoing, repeatable and consistent testing and evaluation (T&E). Requirements for such a test implementation system include ability to

1. deploy agents to interface with SoS component systems in specified assignments
2. enable agents to exchange information and coordinate their behaviors to achieve specified experimental frame data processing
3. respond in real-time to queries for test results while testing is still in progress
4. provide real-time alerts when conditions are detected that would invalidate results or otherwise indicate that intervention is required

5. centrally collect and process test results on demand, periodically, and/or at termination of testing.
6. support consistent transfer and reuse of test cases/configurations from past test events to future test events, enabling life-cycle tracking of SoS performance.
7. enable rapid development of new test cases and configurations to keep up with the reduced SoS development times expected to characterize the reusable web
8. service-based development supported on the GIG/SOA.

Many of these requirements are not achievable with current manually-based data collection and testing. Instrumentation and automation are needed to meet these requirements. Net-centric Service Oriented Architecture (SOA) provides a currently relevant technologically feasible realization of the concept. As discussed earlier, the DEVS/SOA infrastructure enables DEVS models, and test agents in particular, to be deployed to the network nodes of interest. [26],[43] provides complete detail on how such observers can be autogenerated and be executed using DEVS/SOA.

4.1 Deploying Test Agents over the GIG/SOA

Figure 11 depicts a logical formulation test federation that can observe a System Under Test (SUT) to verify the message flow among components as derived from information exchange requirements. In this context, a mission thread is a series of activities executed by operational nodes. In playing out this thread, DEVS test models are informed of the current activities (or see to detect their onset) as well as the operational nodes that execute these messages. These test models watch messages sent and received by the components that host the participating operational nodes. The test models check whether such messages are the ones that should be sent or received under the current function.

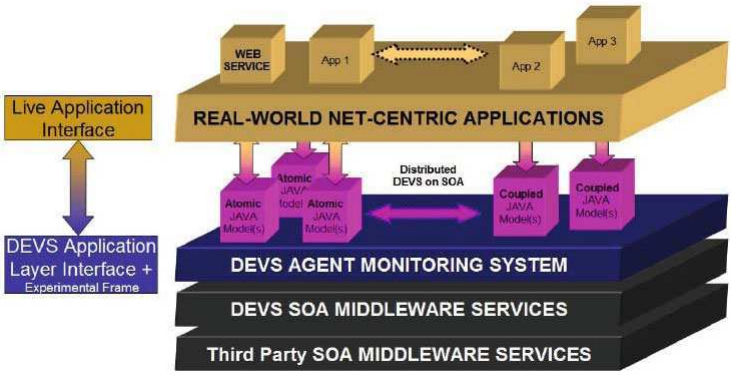


Fig. 11. Multi-layered Agent-based Test Instrumentation Framework

The test-agents are contained in DEVS Experimental Frames (EF) are implemented as DEVS models, and distributed EFs are implemented as DEVS models, or agents as we have called them, reside on network nodes. Such a federation, illustrated in Figure 12, consists of DEVS simulators executing on web servers on the nodes exchanging messages and obeying time relationships under the rules contained within their hosted DEVS models. This DEVS Agent Monitoring System that contains DEVS models interacts with real world web services, as we shall in Section 7 case study.

4.2 Implementation of Test Federations

A test federation observes an orchestration of web-services to verify the message flow among participants adheres to information exchange requirements. A good way to specify these requirements is through Department of Defense Architecture Framework (DoDAF) that have specific documents (OV-3 and SV-6) to localize these information exchanges [9]. These documents very well define the input and output messages for the constituent system and operational components. As derived from DoDAF inputs, a mission thread is a series of activities executed by operational nodes and employing the information processing functions of web-services. As discussed in [26],[43], test agents watch messages sent and received by the services that host the participating operational nodes. Depending on the mode of testing, the test architecture may, or may not, have knowledge of the driving mission thread under test. If thread knowledge is available, DEVS test agents can be aware of the current activity of the operational nodes it is observing. This enables it to focus more efficiently on a smaller set of messages that are likely to provide test opportunities. A DEVS distributed federation is a DEVS coupled model whose components reside on different network nodes and whose coupling is implemented through middleware connectivity characteristic of the environment, e.g., SOAP for GIG/SOA, The federation models are executed by DEVS simulator nodes that provide the time and data exchange coordination as specified in the DEVS abstract simulator protocol.

To help automate set-up of the test we use a capability to inter-covert between DEVS and XML. DEVSML allows distributing DEVS models in the form of XML documents to remote nodes where they can be coupled with local service components to compose a federation [23],[24]. The layered middleware architecture capability is shown earlier in Figure 6. Such run-time interoperability provides great advantage when models from different repositories are used to compose models using DEVSML seamless integration capabilities. Finally, the test federation is illustrated in Figure 12 where different models (federates) in DEVSML collaborate for a simulation exercise over GIG/SOA.

This section has laid out the framework on the creation and execution of a DEVS-based test instrumentation system. More details on the TIS design aspects can be seen in [26]. In the next section we will demonstrate how it can be applied to web services framework.

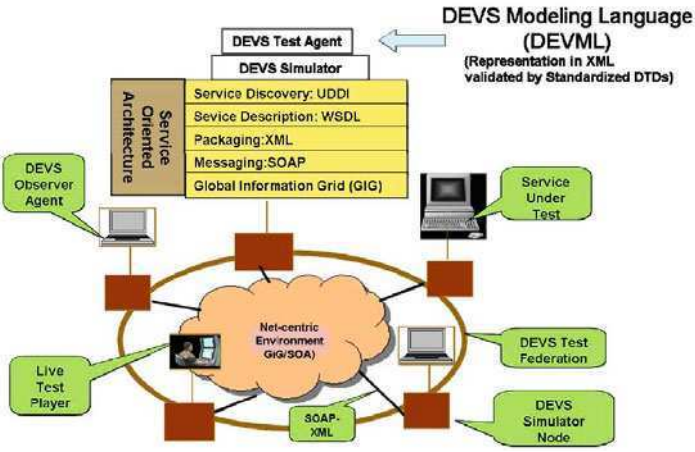


Fig. 12. Protypical DEVS Test Federation

5 Abstract DEVS Service Wrapper

This section will provide details about the role of DEVS interface with a live web service. This is the most crucial step as it links a live web service with a modeling and simulation framework. It is the seat of model-continuity [52] where a DEVS atomic model performs the dual role of a model as well as a wrapper for a real software application utilizing web services.

Web services are utilized using web service clients that are created by various open source and commercially available tools such as Eclipse Web Service Toolkit (WST), Netbeans IDE, Websphere etc.. All of them use the WSDL as the input to generate the web service client. In our implementation we utilize the Axis2 framework to generate clients. Our choice of Axis2 plugin is driven by the implementation platform of DEVS/framework which is Axis/Java. However, it doesnt matter which method is used to generate the client.

A DEVS model has two modes of operation: an internal behavior representation and an external behavior representation. In developing a DEVS wrapper, which would be effectively a DEVS web service client, we will implement the external behavior. The concept is shown in the top half of Figure 13. The detail is shown in the lower half of the same Figure 13. It shows the mapping between the Axis layers, specifically the Axis binding layer and the DEVS elements. It describes the external event that is triggered whenever there is message exchange through the Axis client. This triggered event informs the DEVS atomic model that wraps this Axis client. Such an arrangement does not create any bottleneck or any pipe between the actual Axis client and the network. The DEVS wrapper is informed of the round-trip-time (RTT) when the actual service has been executed its completion. Consequently, it is a passive observer and offers no interference to the true communication between the client and the live web

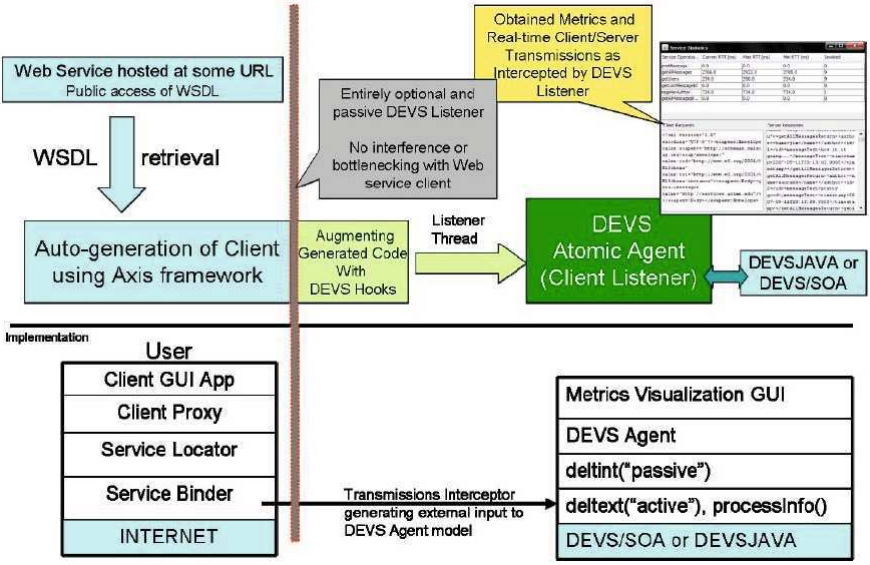


Fig. 13. DEVS Wrapper implementation over an Axis Web Service client

service. By inserting a specific set of code in any Axis generated client, we can create a DEVS wrapper that is ready to become a part of a test-agent federation coupled system, as described in the previous section.

Having described the basic DEVS Web service wrapper, the next task in line is the creation of a coupled model, a web service workflow to be more specific to actually utilize the DEVS modeling and simulation capabilities.

6 Workflow Composition and DoDAF-Based Mission Threads

Web service workflows and orchestration is generally done using BPEL or BPMN or hard-coded in a language specific platform implementation such as Java or .NET. However, to create a DEVS coupled model there are numerous ways [24]. For example the most recent XML-Based Finite Deterministic DEVS (XFD-DEVS) [25] uses XML as the preferred means to develop a Platform Independent Model for both atomic and coupled models. Providing another method to create a web service workflow is beneficial for both the communities. Not only does it provide modeling and simulation capabilities to the existing Web Service architecture, it also establishes DEVS as a production environment that can effectively create application level code using system theoretical concepts.

Another mode of system level design is made possible by System Entity Structure (SES) [44]. The SES is a high level ontology framework targeted to modeling, simulation, systems design and engineering. Its expressive power, both in

strength and limitation, derive from that domain of discourse. An SES is a formal structure governed by a small number of axioms that provide clarity and rigor to its models. The structure supports hierarchical and modular compositions allowing large complex structures to be built in stepwise fashion from smaller, simpler ones. Tools have been developed to transform SESs back and forth to XML allowing many operations to be specified in either SES directly or in its XML guise. The axioms and functionality based semantics of the SES promote pragmatic design and are easily understandable by data modelers. Together with the availability of appropriate tool support, it makes development of XML Schema transparent to the modeler. Finally, SES structures are compact relative to equivalent Schema and automatically generate associated executable simulation models.

The most recent Department of Defense Architecture Framework (DoDAF) application to GIG/SOA is another contender to compose web service workflows for mission-thread design and evaluation. DoDAF, as applicable to mission-thread testing, consists of three views: Operational View (OV), Systems View (SV) and Technical View (TV). It comprises of 26 documents to describe a mission thread. Wrapping head around such documents require sufficient level of understanding and experience with C4ISR frameworks. The main documents are listed in Table 1.

For more detailed analysis of DoDAF, refer [20],[21]. Figure 14 shows the various DoDAF views map into the SES framework.

Operational and System perspectives are considered two different decompositions of the system under consideration. They are represented by corresponding nodes called aspects labeled by the names, Operational View and System View, respectively. The Operational View aspect has entities labeled opNodes (operational nodes) and activities. The various operational views of DoDAF

Table 1. Relevant DoDAF products

Description	**DoDAF Type**
Overview and Summary Information	AV-1
High-Level Operational Concept Description	OV-1
Operational Node Connectivity Description	OV-2
Operational Information Exchange Matrix	OV-3
Organizational Relationships	OV-4
Operational Activity Model	OV-5
Operational Event Trace Description	OV-6b,c
Systems Interface Description	SV-1
Communication Description	SV-2
Systems to Systems Matrix	SV-3
Functionality Description	SV-4
Operational Activity to Function Traceability Matrix	SV-5
Data Exchange Matrix	SV-6
Technical Standards Profile	TV-1

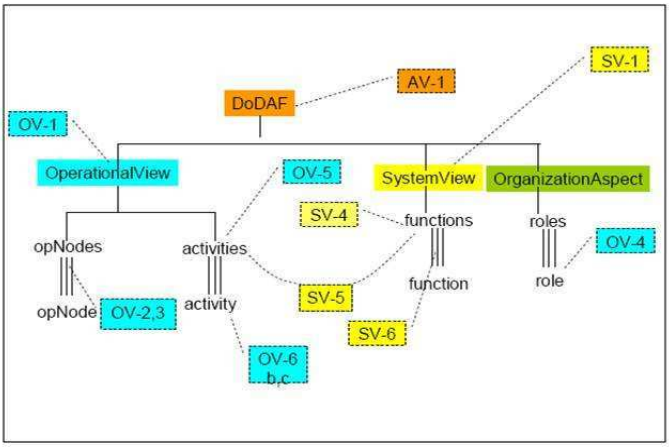


Fig. 14. Mapping of DoDAF documents to System Entity Structure (SES)

(other than OV-4) are easily interpreted as describing the entities and their interactions. Likewise, the System View aspect has entities labeled functions with DoDAF views that are associated with the functions and their interactions. The one exception is SV-5 which is a relation between the functions of the System View and the activities of the Operational View. This view describes how the activities are implemented via executable functions supplied by the system.

Although the current DoDAF specification provides an extensive methodology for system architectural development, it is deficient in several related dimensions absence of integrated modeling and simulation support, especially for model-continuity throughout the development process, and lack of associated testing support [20]. To overcome these deficiencies, we described an approach to support specification of DoDAF architectures within a development environment based on DEVS-based modeling and simulation. The authors [20],[45] enhanced the DoDAF specification to incorporate M&S as a means to develop executable architecture [2] from DoDAF specifications and provided detailed DoDAF to DEVS mapping leading to simulation, and feasibility analysis.

6.1 Web Service Work Flow Formalism

So, after providing an overview of various frameworks that can compose a web service workflow, or simply a process workflow based on certain goals, objectives or requirements, we can deduce the information we need to compose a workflow and develop an automated procedure towards DEVS based design and analysis.

The information set for a Web Service workflow formalism can be described in a four element tuple as:

$$WSWF :< W, M, F, X >$$

where,

W: Set of Web service definitions (WSDLs) or Agents each with a valid URL

M : Set of web service methods

F : defined as $< C, L, D >$

where,

C is a set of W-M pairs with each pair as a source or destination

L is a set of partner links with each link containing a src and dest pair defined in C

D is a type of workflow mode which can either be a sequence, while, holdSend or concurrent type which are corresponding to the BPEL specifications

X is a Set of messages,

where,

each Message contains Data and is defined by time of entry in system, rate, whether it is periodic or stochastic and can be either an Input message or an Output message

The WSWF is expressed in SES as shown in Figure 15 and Figure 16:

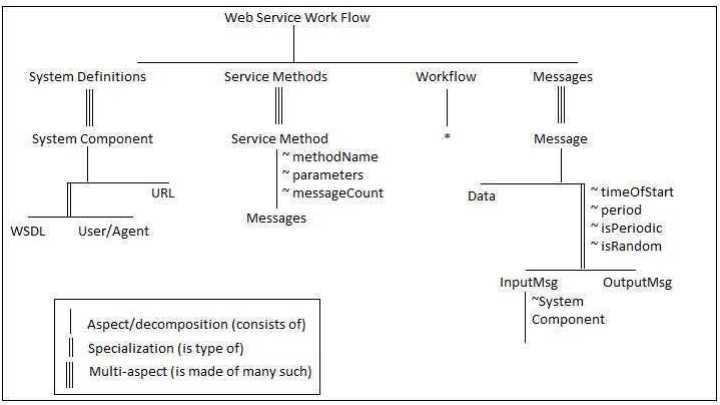


Fig. 15. SES representation of Web Service Work Flow Formalism

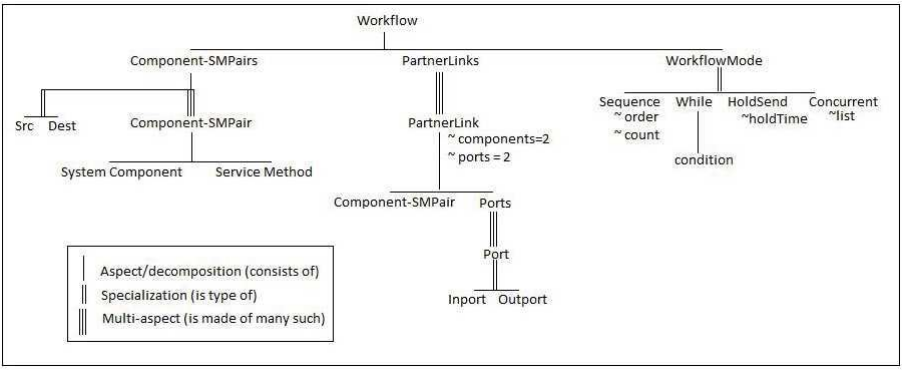


Fig. 16. SES representation of Workflow entity from Figure 15

The WSWF is represented using natural language as shown below:

```
From FORMALISM perspective, WSWF is made of SystemDefinitions,
 ServiceMethods, and Workflow!
From INFO perspective, WSWF is made of messages!
From SystemContainter perspective, SystemDefinitions is made
 of more than one SystemComponent!
From MethodContainer perspective, ServiceMethods is made of
 more than one ServiceMethod!
From MessageContainer perspective, Messages is made of more
 than one Message!

From SystemStructure perspective, SystemComponent is made of URL!
From MessageStructure perspective, Message is made of DATA!
From WorkflowStructure perspective, Workflow is made of
 WorkflowMode, ComponentServiceMethodPairs, and PartnerLinks!
From InfoStructure perspective, ServiceMethod is made of
 InfoExchanges!

From INFO perspective, InfoStructure is like MessageContainer!

SystemComponent can be WSDL, or USERAGENT in SystemType!
Message can be InputMsg, or OutputMsg in MessageType!
Message has timeOfStart, period, is_Periodic, and is_Random!
InputMsg has SystemComponent!

ServiceMethod has methodName, parameters, and messageCount!

WorkflowMode can be Sequence, While, HoldSend, or Concurrent in
 WorkflowType!
```

```
From WhileStructure perspective, While is made of Condition!
Sequence has order, and count!
HoldSend has holdTime!
Concurrent has List!

From ComponentSMPairContainer perspective, ComponentService-
 -MethodPairs is made of more than one ComponentServiceMethodPair!
From PartnerLinkContainer perspective, PartnerLinks is made of
 more than one PartnerLink!
PartnerLinks has ComponentCount and PortCount!
the range of PartnerLink's ComponentCount is RANGE with
 values(2,2)!
the range of partnerLink's PortCount is RANGE with value(2,2)!

From PartnerLinkStructure perspective PartnerLink is made of
 ComponentServiceMethodPair, and Ports!
ComponentServiceMethodPairs can be Src, or Dest in ComponentType!
From PortContainer perspective, Ports is made of more than one
 Port!
Port can be Inport, or Outport in PortType!
```

By expressing the SES for WSWF formalism in restricted natural language, it is made executable using SES-DEVS methodology as elaborated in Zeiglers recent book [43]. Using the SES builder [33], we can very well extract the DTD and/or schema for WSWF. The generated DTD for WSWF is available at [59].

6.2 Mapping of DEVS, BPEL and DoDAF Artifacts with WSWF Formalism

The WSWF information set can very well be extracted from the DoDAF information set. WSWF formalism has also been mapped to XML-Based Finite Deterministic DEVS (XFDDEVS) [25],[46] atomic and coupled models. XFDDEVS is defined by the following tuple:

$$\begin{aligned}\text{Atomic } XFDDEVS = \; & < incomingMessageSet, \\ & outgoingMessageSet, \\ & StateSet, \\ & TimeAdvanceTable, \\ & InternalTransitionTable, \\ & ExternalTransitionTable, \\ & OutputTable >\end{aligned}$$

$$\begin{aligned}\text{Coupled } XFDDEVS = \; & < ModelSet, \\ & CouplingSet >\end{aligned}$$

The table below shows the mapping with BPEL as well. Although mapping to WSWF to BPEL is in early stages, WSWF does have the information set that is required to generate a BPEL file and the associated WSDL file as well. The code to DEVS models has been autogenerated using technologies like JAXB and XSLT. The autogenerated code provides us the DEVS skeleton in platform independent implementation in XML which could be transformed to platform specific implementation in Java, C++ or C#. More information on platform independent DEVS model generation can be seen at [25]. This skeleton can be easily augmented for any run-time capabilities. Providing detailed code implementations have been retained for brevity.

Table 2. WSWF Mapping with DoDAF, XFDDEVS, and BPEL

WSWF		DoDAF	XFDDEVS	BPEL
W		OV-2, OV-4, SV-4	ModelSet	Process
M		OV-5, OV-6	StateSet, ExternalTransitionTable	Basic Activities, PartnerLink-PortType definitions
F				
	C	W, M, OV-2, OV-8	ExternalTransitionTable paramss, InternalTransitionTable params	PartnerLink params, source and target specs in both basic and structured activities
	L	SV-2	CouplingSet	PartnerLinks
	D	SV-5, OV-5, OV-6	ExternalTransitionTable, InternalTransitionTable	StructuredActivities
X		SV-6, OV-3	ExternalTransitionTable, OutputTable	Messages in accompanying WSDL

The WSWF formalism is a new way to compose web service workflows that is expressed in SES-XML methodology. Since it is expressed in XML, it can be mapped easily to XPDL and possibly BPEL too. Since it is largely textual, it can retrieve information from static DoDAF documents as per Table 1. This detailed mapping, however, is not the focus of the current research and will be reported in our forthcoming publication. Going further in our development and execution of this workflow, the following sequential process provide the needed steps in order to do performance evaluation using DEVS test models [26],[24] or execution using DEVS/SOA framework as a real application. In terms of net-ready capability testing, what is required is the communication of live web services with those of test-models designed specifically for them. The approach is:

1. Specify the scenario using WSWF
2. Develop the DEVS model (or generate DEVS workflow)
3. Auto-generate the test-models from DEVS models (using DUNIP as described in [26])

4. Run the model and test-model over SOA (as per DUNIP)
5. Execute as a real-time simulation with live web services embedded in DEVS
6. Execute the test-models with real-world web services (live)
7. Compare the results of steps 5 and 6 to perform verification and validation.

7 Case Study

This case study is divided into two parts:

1. The first study demonstrates the execution of a web service encapsulated in a DEVS wrapper Agent and the associated obtained statistics.
2. The second study extends the first study by developing a workflow that utilizes more than one web services in a workflow manner. It demonstrates the following:
   - Observe user activity with DEVS Agent via WSDL-based access to collaborative service
   - Deploy DEVS virtual user models to simulate humans in collaboration scenario with human user in the loop
   - Show how DEVS agent observers communicate with other DEVS agent via DEVS/SOA infrastructure.

7.1 DEVS Wrapper Agent

In this most basic demonstration, we use only one web service. This web service executes a chat session between two users. The schematic is shown in Figure 17. In our example, we execute the session with a live person and a DEVS agent. The live person here is Jim Client that connects to the CHAT service via an Internet browser at [6]. The chat session is executed using the GUI as shown in Figure 18.

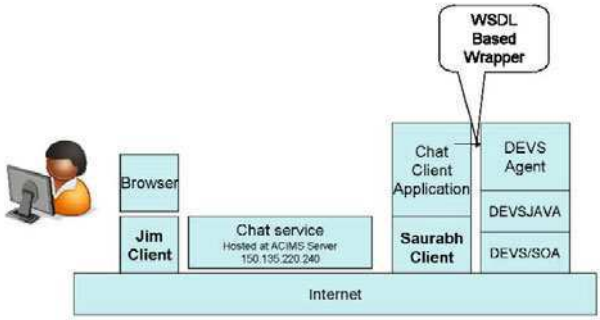


Fig. 17. Schematic showing basic execution of DEVS Wrapper Agent

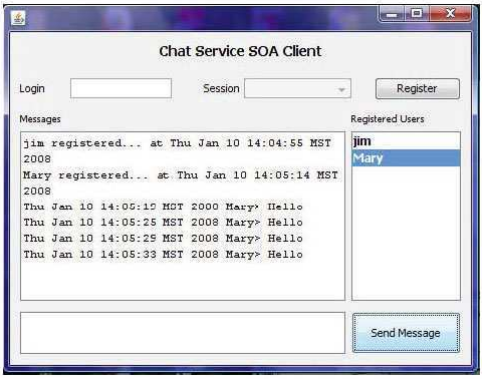


Fig. 18. Chat service client engaged with another chat particpant

The DEVS agent is defined according to the WSWF formalism as follows:

```
<W>: CHAT:
<W1:CHAT>:http://150.135.220.240:8080/ChatServiceCollaboration
/services/ChatService?wsdl
<A1:Jim>: Jim:localhost:8080
<M>: Methods:
<M1> postMessage()
<M2> getAllMessages()
<M3> getLastMessageId()
<M4> registerAuthor()
<M5> getUsers()
<M6> getAllMessagesForAuthor()
<F>:"Flow specifications"
<C>
<C1:Src>A1-M1
<C2:Src>A1-M2
<C3:Src>A1-M4
<C4:Src>A1-M5
<C5:Dest>W1-M1
<C6:Dest>W1-M2
<C7:Dest>W1-M4
<C8:Dest>W1-M5
<L>
<L1>C1-C5
<L2>C2-C6
<L3>C3-C7
<L4>C4-C8
<D>
<D1>M1-HoldSend
```

```
<D2>M2-While-infinity
<D3>M4-HoldSend
<D4>M5-While-infinity
<X>: Set of Messages
<InputMsg>
<I1>W1-M1{string:T1:0:false:false}
<I2>W1-M4{string:T0:0.1:true:false}
<OutputMsg>
<O1>M2{string:T2:1:true:false}
<O2>M5{string:T2:1:true:false}
```

`<W>` tag contains description of the Chat Web Service as `W1` and the agent description as `A1` along with their URL. `<M>` contains the list of service methods that may be used in the process flow. `<F>` contains the flow description categorized into `<C,L,D>` as per the WSWF. `<C>` provides the source and destination specification for a W/A defined in `<W>` with `<M>`. `<L>` specifies the coupling between the sources and destinations as defined in `<C>`. `<D>` contains the execution of methods in `<M>` in logic implementation. For example, `<D1>M1-HoldSend` implies that the method `M1` is to executed in `HoldSend` manner. Similarly, `<D2>M2-While- infinity` implies that `M2` will be executed indefinitely when invoked or bounded by any condition. `<X>` specifies the input and output message structures in `<InputMsg, OutputMsg>` tags. The structure of `<InputMsg>` as specified in `WSWF` SES is `<SystemComponent-Method{Data: time of Start: R+: isPeriodic: isRandom>`. For example, the specification `<I1>W1-M1string:T1:0:false:false` implies that the message `I1` is an input to `W1`, method `M1` with data as `string`. It starts at `T1` with period `0`. Any non-zero value means that the message will be incoming at a periodic rate. The next boolean variable `false` implies that it is not periodic. The last variable `false` implies that it is not random either. Similary, `<I2>W1-M4string:T0:0.1:true:false` implies that `M4` at `W1` is to be invoked by string data message with a periodic rate of `0.1`. The `<OutputMsg>` has a similar structure except the fact that it does not contain any information about the system component. It only contains information about the method in `<M>` as it is just an output message. Whenever method `<Mx>` is invoked, it returns with the parametric details as in `<O1>M2string:T2:1:true:false`.

It is worth stressing here that the messages flow through the linkages as specified in `<L>`. This acts as a coupling for the DEVS models. There are two DEVS models in the WSWF instance described above, viz. `W1` and `A1`. Based on the coupling information for ex. `<L4>C4-C8` implies that the source is Agent `<C4:Src>A1-M5` and the destination is Web service `<C8:Dest>W1-M5`. The source sends a message invoking method `M5` at the destination. If there is a specification on how `M5` should be invoked in `<InputMsg>` listing, then the source has to ensure that it conforms to that specification. In this example there is no specification for `M5`. This implies that there are no parameters to be passed, but just the invocation. At the destination side, `M5` has a

specification `<O2>M5string:T2:1:true:false`, which implies that whenever `M5` returns a value, it will according to this `<OutputMsg>` specification.

The statistics for each of the methods in `<M>` is gathered according to the autogenerated agent GUI monitor at the agents end. The statistics are largely the round trip time (RTT) for each of `<M>`. The GUI in Figure 19 also shows the SOAP messages that are exchanged between the pairs as specified in `<W>`.

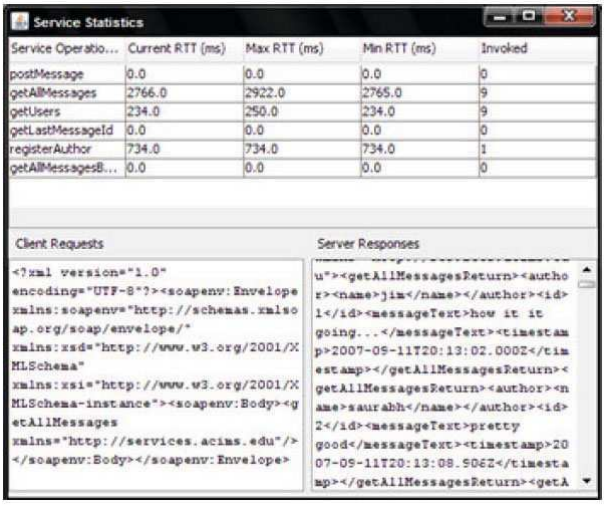


Fig. 19. Associated statistics GUI for an encapsulated Web Service in DEVS Atomic Model

7.2 Workflow Design, Analysis and Execution

The previous demonstration has established that we can encapsulate a live web service within a DEVS atomic model using an XML based formalism such as WSWF. It also establishes that we can create virtual users as DEVS agents that input and communicate with live users. Having such capability allows us to build upon the advances of DEVS hierarchical component based modeling and simulation. In the next demonstration, we will build a workflow with two live web services and all the clients as virtual users.

DEVSJAVA Execution on a Single Machine The first service is the same CHAT service and the second service is a weather service [37]. In this demonstration, we will show that virtual users are engaged in chat session and one user requests weather from another user. The second user (Jim Client) shown in Figure 20 requests the weather from the Weather web service and reports it back to the first user using the CHAT service. We will then also execute the entire scenario as a self-contained coupled model on DEVS/SOA with these virtual agents deployed at different IP addresses. The schematic is shown in Figure 20.

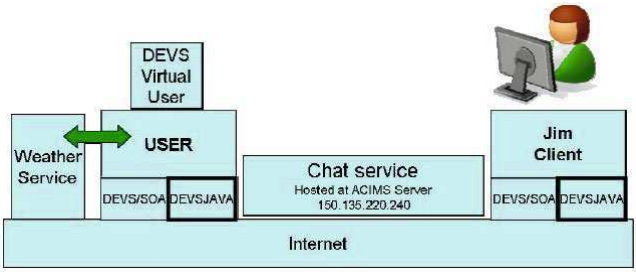


Fig. 20. Schematic of Workflow scenario with two virtual DEVS agents

The workflow according to WSWF formalism is defined as follows:

```
<W>: CHAT-and-WEATHER:
<W1:CHAT>:<http://150.135.220.240:8080/ChatService Collaboration/
services/ChatService?wsdl>
<W2:WEATHER>: http://www.webservicex.net/WeatherForecast.asmx.
<A1:JIM>: Jim:localhost:8080
<A2:USER>: User:localhost:8080
<M>: Methods:
<M1> postMessage()
<M2> getAllMessages()
<M3> getLastMessageId()
<M4> registerAuthor()
<M5> getUsers()
<M6> getWeather()
<F>:"Flow specifications"
<C>
<C1:Src>{A1,A2}-M1
<C2:Src>A1-M2
<C3:Src>{A1,A2}-M4
<C4:Src>A1-M5
<C5:Src>A2-M6
<C6:Dest>W1-M1
<C7:Dest>W1-M2
<C8:Dest>W1-M4
<C9:Dest>W1-M5
<C10:Dest>W2-M6
<L>
<L1>C1-C6 //notice that both A1 and A2 are coupled to W1-M1
<L2>C2-C7
<L3>C3-C8 //notice that both A1 and A2 are coupled to W1-M4
<L4>C4-C9
<L5>C5-C10
<D>
```

```
<D1>M1-HoldSend:5
<D2>M2-While-infinity
<D3>M4-HoldSend
<D4>M5-While-infinity
<D5>M6-HoldSend
<X>: Set of Messages
<InputMsg>
<I1>W1-M1{string:T1:5:true:false}
<I2>W1-M4{string:T0:0.1:true:false}
<I3>W2-M6{string:T3:0:false:false}
<I4>A2-M1(string:T4:0:false:false)
<OutputMsg>
<O1>M2{string:T2:1:true:false}
<O2>M5{string:T2:1:true:false}
<O3>M6{string:T3:0:false:false}
where, T0 > 0, T1&T2 > 0, and T3,T4>T1,T2.
```

The description of WSWF instance above is on the same lines as of previous example. However, instead of just one, there are two services in this instance as specified by `<W1>` and `<W2>`. The two services are: the `Chat Service` and the publically available `Weather service`. There is an addition method `<M6>` that invokes the `Weather service`. There are two agents viz., `Jim` and `USER`. The `USER` is a virtual user and is modeled as a DEVS Agent and `Jim` is a live person. A DEVS agent is a computer program implemented as a DEVS model.

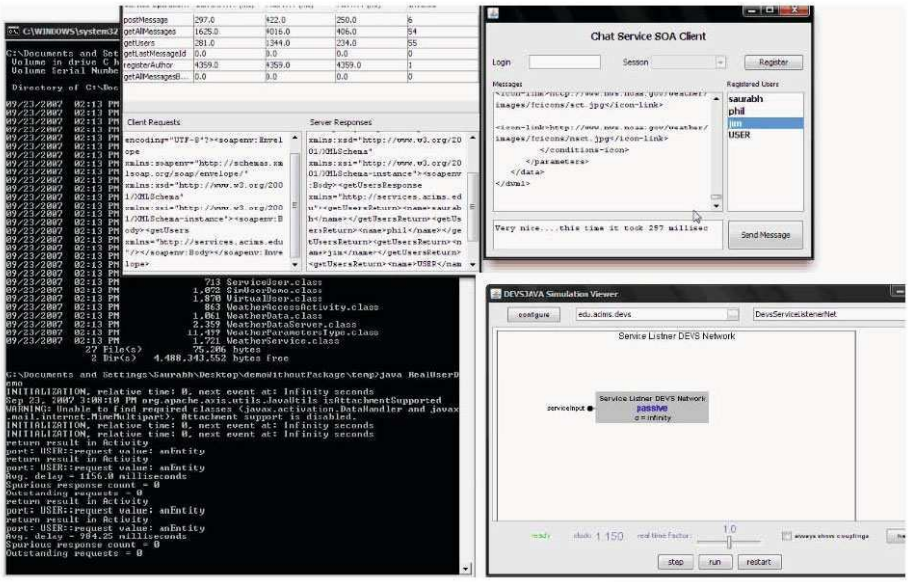


Fig. 21. Snapshot of execution of workflow case study as depicted in Figure 21

It is engaged in chat session with `Jim` and reports back the results of `Weather service` when the request to invoke comes from `Jim`, the real user.

The demonstration has been structured in a manner that it be executed in a single machine. To execute it on remote machines we will be using DEVS/SOA which is described in the next sub-section. To execute it on a single machine, DEVSJAVA platform is sufficient. Figure 21 shows the virtual user USER in black console and the `Jim` real user in the Chat window. Notice that the Jim client also has the monitor running that invokes method `<M4>` and `<M5>` at the `Chat Web service`. The GUI also shows the DEVSJAVA simulation viewer which shows that DEVSJAVA is being used to run the scenario. The Jim client requests weather from USER client. The `USER` invokes `<W2>` web service, and reports back the result by method `<M1>` to the `Chat Service`.

To provide complete performance analysis of the workflow as per the GUI in Figure 21 is outside the scope of the paper and has been retained.

Execution on DEVS/SOA Framework. The scenario remains the same as in preview sub-section. However, the execution is made on DEVS/SOA platform (Figure 22). The real user Jim has now been replaced by another virtual client. The only modification in the WSWF instance is the following:

```
<W>: CHAT-and-WEATHER:
<W1:CHAT>:<http://150.135.220.240:8080/ChatService Collaboration/
services/ChatService?wsdl>
<W2:WEATHER>: http://www.webservicex.net/WeatherForecast.asmx.
<A1:USER1>: User1:150.135.218.205:8080
<A2:USER2>: User2:150.135.220.240:8080
```

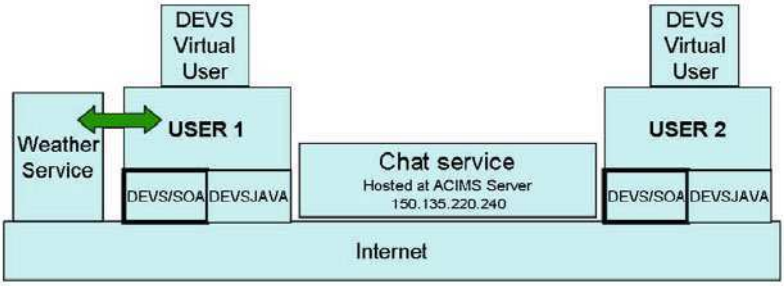


Fig. 22. Execution of Workflow scenario with DEVS/SOA framework

The generated Java code is fed to the DEVS/SOA client GUI as reproduced again in Figure 23. `USER2` in the generated code is given the Class name `PerfMonitor` for differentiation. The Class VirtualUser is `USER1`. The `USER1` is assigned an IP address `150.135.218.205:8080` and `USER2` at `150.135.220.240: 8080`. These

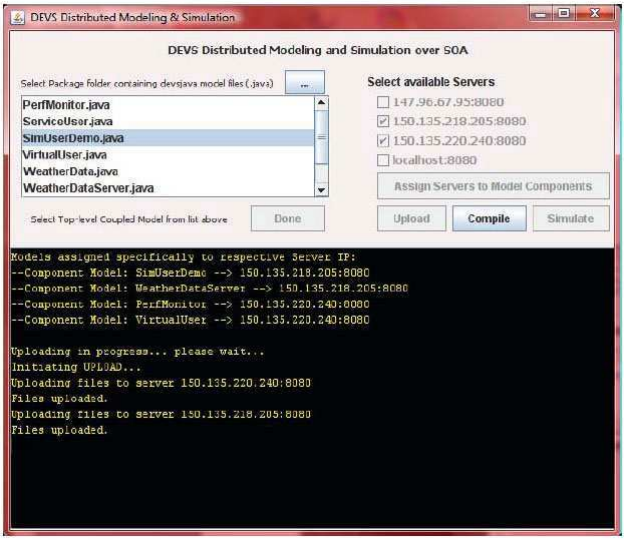


Fig. 23. Models package being executed using DEVS/SOA client

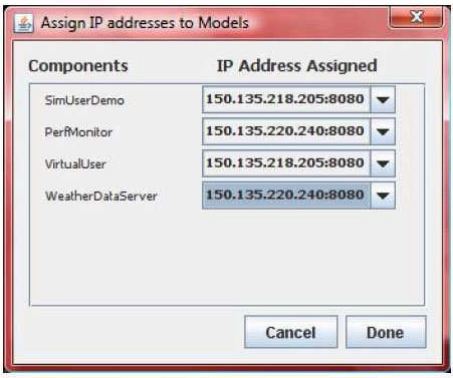


Fig. 24. IP assignment of models for Workflow scenario

virtual users are then sent to these respective IP addresses. These IP addresses provide the DEVS Simulation service and Apache Tomcat servers are used as containers at these IPs. The other dependent files are also uploaded at corresponding IPs. The assignment can be done manually as shown in Figure 24. Once uploaded, the files are compiled at run-time at the servers end and a distributed simulation is executed between these remote machines. Once the simulation is over, the result is communicated back into the console window as shown in Figure 23. The detailed output of the simulation run is shown below. As can be seen from the output below, VirtualUser sent three requests

and got responses with different delays. The responses are communicated by the other USER2 after invoking the Weather service. The result is also sent back to VirtualUser, as,

```
Place Name:HILLSIDE MANOR;MinTempC:-4;MaxTempC:5.

Models assigned specifically to respective Server IP:
--Component Model: SimUserDemo --> 150.135.218.205:8080
--Component Model: VirtualUser --> 150.135.218.205:8080
--Component Model: PerfMonitor --> 150.135.220.240:8080
--Component Model: WeatherDataServer --> 150.135.220.240:8080
Uploading in progress... please wait...
Initiating UPLOAD...
Uploading files to server 150.135.220.240:8080
Files uploaded.
Uploading files to server 150.135.218.205:8080
Files uploaded.
Compilation in progress....please wait....
Starting compilation at remote servers.....
Compiling project at 150.135.220.240:8080...
Success: true
Project compiled.

Compiling project at 150.135.218.205:8080...
Success: true
Project compiled.
Waiting to start SIMULATION....
Simulation in Progress....please wait...
Running simulation ...
21 iterations.
Simulators output:
150.135.220.240 output:
VirtualUser: sent a request
Avg. delay = 375.0 milliseconds
Spurious response count = 0
Outstanding requests = 0
VirtualUser: response length 48
Place Name:HILLSIDE MANOR;MinTempC:-4;MaxTempC:5
VirtualUser: sent a request
Avg. delay = 355.25 milliseconds
Spurious response count = 0
Outstanding requests = 0
VirtualUser: response length 48
Place Name:HILLSIDE MANOR;MinTempC:-4;MaxTempC:5
SIMULATION over!
```

Hybrid Execution Using DEVS/SOA and DEVSJAVA. Once we have such DEVS coupled workflow system, we can extend this system by replacing any virtual user with a live user. Figure 25 below shows the schematic of such operation and a demonstration is made available as an .avi file at [8]. In the schematic below the DEVS coupled system is augmented with other DEVS agents for reporting statistics etc, basically the idea being, such DEVS enabled workflows can now participate in live modeling and simulation exercises in real-time.

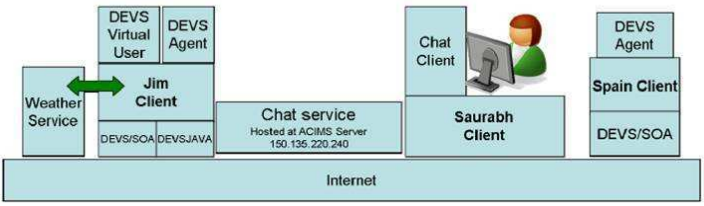


Fig. 25. WSWF formalism based workflow using DEVS as middleware for live modeling and simulation exercises

8 Agility in DEVS Unified Process

Agile software methodologies have taken quite a notice these recent years primarily due to the factors such as volatile ever-changing requirements, dynamic technological landscape, high employee turnover, and most importantly, satisfying the business needs. [50] summarizes it as:

> Agile development is not defined by a small set of pratices and techniques but defines a strategic capability, a capability to create and respond to change, a capability to balance flexibility and structure, a capability to draw creativity and innovation out of a development team, and a capability to lead organizations through turbulence and certainty.

There is a fundamental shift in the approach of delivering the product by hard-line requirements specifications supported by methodologies like Capability Maturity Model (CMM) and CMM Integration (CMMI) and the Agile practices. While the former delineates defined repeatable processes so that the performance can be measured within very close tolerances, the latter is more geared towards employing the latest advancement in technologies, to explore, and to deliver the product as soon as possible. The key point of agile practices is the inclusion of software engineering life cycle in each iteration so that the features delivered are production ready at the end of each iteration. While the vision of most projects are clear, what remains fuzzy are the exact requirement specifications that the developers are faced with. With agile practices, a constant dialogue with the customer, repeatable testing procedures, incremental development and using the latest technology, the requested feature can be delivered in the next

Table 3. Agile Methodology and DEVS Unified Process

Phase	Agile Methodology	DEVS Unified Process (DUNIP)
Model	Identify the domain and business use-case requirements and specify in domain specific languages such as UML, etc.	DUNIP begins by taking requirements in different formats like DoDAF, UML, State-based, NLP and transform them into platform independent XML models
Implementation	Transform your models into executable code with running unit-tests	From PIMs, the DUNIP engine generates code in platfrom specific models (PSMs) such as Java, C++, C# etc. With strong DEVS theory underlying each of the atomic models, the models can be verified mathematically, as well as graphical with various DEVS toolsets such as DEVSJAVA. Unit-testing for each transition or an event is inherent in DEVS.
Test	Identify defects, ensure quality and verify requirements	With DUNIP, the development of test suite is done in parallel with that of the model. Test models are generated from the XML-based PIMs. The test models verify the atomic model's operation at various levels of system specifications, such as I/O pair, I/O function, etc. The Experimental Frames are also designed at this stage that ensure the requirements are met through the test models.
Deployment	Plan the delivery and make it available to end users	With ready deployment capabilities per model-continuity principles to SOA infrastructure, and zero transition times, the model is the actual software and is readily moved to the production servers
Configuration Management	Managed access to project artifacts	DUNIP is very well positioned to reuse and contribute to Model repository. PIMs are a strong contender for such tracking and version management. PSMs can very well be source versioned using tools like Subversion etc.
Project Management	Manage people, project, iterations and budget	These qualities are universal and due to the component nature of DEVS technology, the project plan can very well be partitioned into iterative cycles and milestones
Environment	Ensure that proper process, guidance, and tools are available for the team	DEVS has been in existence for over 30 years and there is a large community that is available for support in basic theory and toolsets.

iteration without changing the entire project vision. The DEVS Unified Process, similarly is based on agile methodology. Table 3 lists the similarities with each phase of agile development methodology [51].

Table 3 establishes that DEVS Unified Process has all the needed phases of being agile and the model continuity [52] enables any DEVS artifiact to be a real software. With DUNIP's SOA edge, we have any DEVS model that is available as a web service. Modeling and Simulation in today's world is more than just a software. It is an enabling technology that has far reaching impact on any nations' progress and advance the forefront of various technologies in many domains such as biology, chemistry, phyiscs, space science, etc. While there are customized M&S software for different problem sets and different domains, an agile methodology is another ace that when employed could incorporate the latest advancements in software engineering discipline and apply it to the M&S solution at hand.

9 Conclusions and Future Work

Service Oriented Architecture (SOA) have come a long way and many of the businesses are seriously considering migration of their IT systems towards SOAs. DoDs initiative towards migration of GIG/SOA and NCES requires reliability and robustness, not only in the execution but in the design and analysis phase as well. Web service orchestration is not just a research issue but a more practical issue for which there is dire need. Further, Service Oriented Architecture must be taken as another instance of system engineering for which there must be a laid out engineering process. Modeling and Simulation provides the needed edge. Lack of methodologies to support design and analysis of such orchestration (except BPEL related efforts) cost millions in failure. This research has proposed that Discrete Event Formalism can be used to compose and analyze Web service workflows. The DEVS theory, which is based on system theoretic concepts, gives solid grounding in the modeling and simulation domain.

The prime motivation of applying DEVS system theoretical principles to these emerging net-centric systems comes from an editorial by Carstairs [53] that demands a M&S framework at higher levels of system specifications where System of systems interact together using net-centric platform. At this level, model interoperability is one of the major concerns. The motivation for this work stems from this need of model interoperability and the characterists of net-centric systems that are easier to simulate, test and deploy with an underlying foundation of systems engineering principles. DEVS, which is known to be component-based system, based on formal systems theoretical framework is the preferred means. Table 4 outlines how it could provide solutions to the challenges in net-centric design and evaluation.

The net-centric DEVS framework as laid out in this chapter required enhancement to these basic DEVS capabilities. Furthermore, this work describes distributed simulation using the web service technology. After the development of World Wide Web, many efforts in the distributed simulation field have been

Table 4. Solutions provided with DEVS Technology in support of M&S for T&E

Desired M&S capability for Test and Evaluation (T&E)	Solutions provided by DEVS M&S technology
Support for DoDAF need for executable architectures using M&S such as mission based testing for GIGSOA	DEVS Unified Process [24],[29] provides methodology and SOA infrastructure for integrated development and testing, extending DoDAF views [20].
Interoperability and cross-platform M&S using GIG/SOA	Simulation architecture is layered to accomplish the technology migration or run different technological scenarios [54]. Provide net-centric composition and integration of DEVS 'validated' models using Simulation Web services [22].
Automated test generation and deployment in distributed simulation	Separate a model from the act of simulation itself, which can be executed on single or multiple distributed platforms [39]. With its bifurcated test and development process, automated test generation is integral to this methodology [42].
Test artifact continuity and traceability through phases of system development	Provide rapid means of deployment using model-continuity principles and concepts like 'simulation becomes reality' [52].
Real-time observation and control of test environment	Provide dynamic variable structure component modeling to enable control and reconfiguratin of simulation on the fly [55],[56]. Provide dynamic simulation tuning, interoperability testing and benchmarking

made for modeling, executing simulation and creating model libraries that can be assembled and executed over WWW. By means of XML and web services technology these efforts have entered upon a new phase. The proposed DEVS Modeling Language (DEVSML) is built on eXtensible Markup Language (XML) as the preferred means to provide such transparent simulator implementation. A prototype simulation framework called DEVS/SOA has been implemented using web services technology. It is currently at the forefront of DEVS net-centric research platform [47]. The central point resides in executing the simulator as a web service. The development of these kind of frameworks will help solve large-scale problems and guarantees interoperability among different networked systems and specifically DEVS-validated models. This chapter focused on the overall approach, and the symmetrical SOA-Based architecture that allows for DEVS execution as a Simulation SOA.

We have shown how a web service can be encapsulated into a DEVS atomic model and can be put towards a coupled DEVS system with other live web services as well as other DEVS models. We also have demonstrated the proposed use of Web Service Work Flow (WSWF) formalism in composing SOA, much like the same functionalities of BPEL. We have also described how WSWF can be mapped to DoDAF elements using the System Entity Structure (SES) and could achieve creation of DEVS net-centric coupled systems based on SOA. We have also shown how the developed DEVS coupled system can be simulated using

the basic DEVSJAVA framework as well as distributed DEVS/SOA framework. Further, on the basis of our earlier work on DEVS/SOA we have basis for:

- Agent-Implemented Test Instrumentation
- Net-centric Execution using Simulation Services
- Distributed Multi-level Test Federations
- Analysis that can help optimally tune the instrumentation to provide confident scalability predictions.
- Mission Thread testing and data gathering:
  - definition and implementation of military-relevant mission threads to enable constructing and/or validating models of user activity.
  - Comparison with current commercial testing tools shows that by replicating such models in large numbers it will be possible to produce more reliable load models than offered by conventional use of scripts.

We have taken the challenge of constructing net-centric systems as one of designing an infrastructure to integrate existing Web services as components, each with its own structure and behavior with DEVS components and agents. The net-centric system is analogous to a System of System (SoS) where in hierarchical coupled models could be created. Various workflows can be integrated together using component based design. The net-centric system can be specified in many available frameworks such as DoDAF, SES, BPMN/BPEL, UML, or by using an integrative systems engineering-based framework such as DEVS. The proposed Web Service Work Flow formalism binds various frameworks like SES, BPEL, DoDAF and DEVS.

In this research, we have discussed the advantages of employing an M&S-integrated framework such as DEVS Unified Process (DUNIP) and its supporting DEVS/SOA infrastructure. We illustrated how M&S can be used strategically to provide early feasibility studies and aid the design process. We have established the capability to develop a live workflow example with complete DEVS interface. In this role, DEVS acts as a full net-centric production environment. Being DEVS enabled, it is also executable as a system under test (SUT) model towards various verification and validation analysis that can be performed by coupling this SUT with other DEVS test models. Last but not the least, the developed DEVS system can be executed by both real and virtual users to the advantage of various performance and evaluation studies. We also summarized how DUNIP is agile and each of its modules fit to the agile practices.

As components comprising SoS are designed and analyzed, their integration and communication is the most critical part that must be addressed by the employed System of System (SoS) M&S framework. We discussed DoDs Global Information Grid (GIG) as providing an integration infrastructure for SoS in the context of constructing collaborations of web services using the Service Oriented Architecture (SOA). The DEVS Unified Process (DUNIP), in analogy to the Rational Unified Process based on UML, offers a process for integrated development and testing of systems that rests on the SOA infrastructure. The DUNIP perspective led us to formulate a methodology for testing any proposed SOA-based integration infrastructure, such as DISAs Net-Centric Enterprise Services.

The present research is being considered and refined for standardization with the DEVS Standardization group [47],[48],[49]. Clearly, the theory and methodology for such net-centric SoS development and testing are at their early stages.

10 Acronyms

ATC-Gen Automated Test Case Generator
BPEL Business Process Execution Language
BPMN Business Process Modeling Notation
C4ISR Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance
CES Core Enterprise Services
CMMI Capability Maturity Model Integration
CORBA Common Object Request Broker Architecture
DEVS Discrete Event System Specification
DEVSML DEVS Modeling Language
DoD Department of Defense (US)
DoDAF Department of Defense Architecture Framework
DTD Document Type Definition
DUNIP DEVS Unified Process
GIG Global Information Grid
HLA High Level Architecture
HTTP Hyper Text Transfer Protocol
JAXB Java Advanced XML Binding
NCES Net-Centric Enterprise Services
NLP Natural Language Processing
OV Operational View (in DoDAF)
PIM Platform Independent Model
PSM Platform Specific Model
RMI Remote Method Invocation
RTT Round Trip Time
SES System Entity Structure
SOA Service Oriented Architecture
SOAP Simple Object Access Protocol
SUT System Under Test
SV System View (in DoDAF)
T&E Test and Evaluation
TIS Test Instrumentation System
TV Technical View (in DoDAF)
WSDL Web Service Description Language
WSWF Web Service Workflow Formalism
XFDDEVS XML-Based Finite Deterministic DEVS
XML eXtensible Markup Language
XPDL XML Process Definition Language
XSLT Extensible Stylesheet Language Transformations
UDDI Universal Description Discover and Integration
UML Unified Modeling Language

References

1. ACIMS software site, http://www.acims.arizona.edu/SOFTWARE/software.shtml (last accessed September 2010)
2. Atkinson, K.: Modeling and Simulation Foundation for Capabilities Based Planning. In: Simulation Interoperability Workshop (Spring 2004)
3. Badros, G.: JavaML: a Markup Language for Java Source Code. In: Proceedings of the 9th International World Wide Web Conference on Computer Networks, pp. 159–177 (2000), http://www.badros.com/greg/JavaML/
4. Business Process Execution Language, http://www.ibm.com/developerworks/library/specification/ws-bpel/
5. Business Process Modeling Notation, http://www.bpmn.org
6. CHAT SOA web service, http://www.saurabh-mittal.com/demos/ChatClient
7. Cheon, S., Seo, S., Park, S., Zeigler, B.P.: Design and Implementation of Distributed DEVS Simulation in a Peer to Peer Networked System. In: Advanced Simulation Technologies Conference, Arlington, VA (2004)
8. Chat-Weather Service Demo as.avi file, http://duniptechnologies.com/training/demos/DEVS_CHAT_Weather_RealUserDemo.avi
9. DoDAF Working Group, DoD Architecture Framework Ver. 1.0 Vol. III: Deskbook, DoD (August 2003)
10. DOD Instruction 5000.2 Operation of the Defense Acquisition System (May 12, 2003)
11. DoD Architecture Framework Working Group 2004, DOD Architecture Framework Ver. 1.0, vol. 1 Definitions and Guidelines, Washington, D.C. (February 9, 2004)
12. DUNIP: A Prototype Demonstration, http://duniptechnologies.com/training/demos/dunip.avi
13. Fujimoto, R.M.: Parallel and Distribution Simulation Systems. Wiley, Chichester (1999)
14. Joint Interoperability Test Command, a Defense Information Systems Agency, http://jitc.fhu.disa.mil/
15. Martin, J.L.R., Mittal, S., Zeigler, B.P., Manuel, J.: From UML Statecharts to DEVS State Machines using XML. In: IEEE/ACM Conference on Multi-Paradigm Modeling and Simulation, Nashville (September 2007)
16. Department of Defense GIG Architectural Vision, Ver. 1.0, prepared by DoD CIO (June 2007), http://www.defenselink.mil/cio-nii/docs/GIGArchVision.pdf
17. Kim, K., Kang, W.: CORBA-Based Multi-threaded Distributed Simulation of Hierarchical DEVS Models: Transforming Model Structure into a Non-hierarchical One. In: International Conference on Computational Science and Its Applications, ICCSA, Italy (2004)
18. Mak, E.: Automated Testing using XML and DEVS. MS Thesis, University of Arizona (2006), http://www.acims.arizona.edu/PUBLICATIONS/PDF/Thesis_EMak.pdf
19. Mak, E., Mittal, S., Hwang, M.H.: Automating Link-16 Testing using DEVS and XML. Journal of Defense Modeling and Simulation 7(1), 39–62 (2010)
20. Mittal, S.: Extending DoDAF to Allow DEVS-Based Modeling and Simulation. Journal of Defense Modeling and Simulation JDMS, Special Issue on DoDAF 3(2), 95–123 (2006)
21. Mittal, S., Mak, E., Nutaro, J.J.: DEVS-Based Dynamic Modeling & Simulation Reconfiguration using Enhanced DoDAF Design Process. Journal of Defense Modeling and Simulation, Special Issue on DoDAF 3(4), 239–267 (2006)

22. Mittal, S., Martin, J.L.R., Zeigler, B.P.: DEVSML: Automating DEVS Simulation over SOA using Transparent Simulators. In: DEVS Syposium (2007)
23. Mittal, S., Martin, J.L.R., Zeigler, B.P.: DEVS-Based Web Services for Net-centric T&E. In: Summer Computer Simulation Conference (2007)
24. Mittal, S.: DEVS Unified Process for Integrated Development and Testing of Service Oriented Architectures. Ph. D. Dissertation, University of Arizona (2007), http://acims.arizona.edu/PUBLICATIONS/PDF/Thesis_Mittal.pdf
25. Mittal, S., Hwang, M.H., Zeigler, B.P.: XFD-DEVS: An Implementation of W3C Schema for XML-Based Finite Deterministic DEVS, in progress, Demo, http://duniptechnologies.com/research/xfddevs
26. Mittal, S., Zeigler, B.P., Martin, J.L.R., Sahin, F., Jamshidi, M.: Systems of Systems Engineering for 21st Century. In: Modeling and Simulation for System of systems Engineering (2008)
27. Mittal, S., Zeigler, B.P., Martin, J.L.R.: Implementation of Formal Standard for Interoperability in M&S/System of Systems Integration with DEVS/SOA. International Command and Control C2 Journal, Special Issue: Modeling and Simulation in Support of Network-Centric Approaches and Capabilities 3(1) (2009)
28. Mittal, S., Martin, J.L.R., Zeigler, B.P.: DEVS/SOA: A Cross-Platform Framework for Net-Centric Modeling and Simulation in DEVS Unified Process. SIMULATION: Transactions of SCS 85(7), 419–450 (2009)
29. Mittal, S., Zeigler, B.P.: DEVS Unified Process for Integrated Development and Testing of System of Systems. In: Critical Issues in C4I, AFCEA-George Mason University Symposium (May 2008)
30. Net-Centric Enterprise Service, http://www.disa.mil/nces/ (last accessed September 2010)
31. Jon, P.: XPDL: The Silent Workhorse of BPM (April 2007), online article, http://www.bpm.com/FeatureRO.asp?FeatureId=232 (last accessed September 2010)
32. Sarjoughian, H.S., Zeigler, B.P.: DEVS and HLA: Complimentary Paradigms for M&S? Transactions of the SCS 17(4), 187–197 (2000)
33. SESBuilder, An Integrated Tool to utilize System Entity Structure (2007), http://www.sesbuilder.com/
34. DEVS/SOA sample demonstration in.avi format, http://duniptechnologies.com/training/demos/demoSOADEVS.avi
35. Simple Object Access Protocol, http://www.w3.org/TR/soap/
36. Seo, C., Park, S., Kim, B., Cheon, S., Zeigler, B.P.: Implementation of Distributed High-performance DEVS Simulation Framework in the Grid Computing Environment. In: Advanced Simulation Technologies Conference (ASTC), Arlington, VA (2004)
37. Weather web service at, http://www.webservicex.net/WeatherForecast.asmx
38. Web Services Description Language, http://www.w3.org/TR/wsdl
39. Zeigler, B.P., Kim, T.G., Praehofer, H.: Theory of Modeling and Simulation. Academic Press, New York (2000)
40. DEVSJAVA, http://www.acims.arizona.edu/SOFTWARE/devsjava_licensed CBMSManuscript.zip
41. Zhang, M., Zeigler, B.P., Hammonds, P.: DEVS/RMI-An Auto-Adaptive and Reconfigurable Distributed Simulation Environment for Engineering Studies. ITEA Journal (July 2005)
42. Zeigler, B.P., Fulton, D., Hammonds, P., Nutaro, J.: Framework for M&SBased System Development and Testing In a Net-Centric Environment. ITEA Journal of Test and Evaluation 26(3), 21–34 (2005)

43. Zeigler, B.P., Hammonds, P.: Modeling& Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange. Academic Press, New York (2007)
44. Zeigler, B.P., Zhang, G.: The System Entity Structure: Knowledge Representation for Simulation Modeling and Design. In: Widman, L.E., Loparo, K.A., Nielsen, N.R. (eds.) Artificial Intelligence, Simulation and Modeling, pp. 47–73. Wiley, New York (1989)
45. Zeigler, B.P., Mittal, S.: Enhancing DoDAF with DEVS-Based System Life-cycle Process. In: IEEE International Conference on Systems, Man and Cybernetics, Hawaii (October 2005)
46. Martin, J.L.R., Mittal, S., Mendel, J., Zeigler, B.P.: eUDEVS: Executable UML Using DEVS Theory of Modeling and Simulation. Invited paper to SIMULATION 85(11-12), 750–777 (2009)
47. Wainer, G., Al-Zoubi, K., Dalle, O., Hill, D., Mittal, S., Martin, J.L.R., Sarjoughian, H., Touraille, L., Traore, M., Zeigler, B.P.: Discrete Event Modeling and Simulation: Theory and Applications. In: DEVS Standardization: Ideas, Trends and Future. CRC Press, Boca Raton (2010)
48. Wainer, G., Al-Zoubi, K., Dalle, O., Hill, D., Mittal, S., Martin, J.L.R., Sarjoughian, H., Touraille, L., Traore, M., Zeigler, B.P.: Discrete Event Modeling and Simulation: Theory and Applications. In: Standardizing DEVS Model Representation. CRC Press, Boca Raton (2010)
49. Wainer, G., Al-Zoubi, K., Dalle, O., Hill, D., Mittal, S., Martin, J.L.R., Sarjoughian, H., Touraille, L., Traore, M., Zeigler, B.P.: Discrete Event Modeling and Simulation: Theory and Applications. In: Standardizing DEVS Simulation Middleware. CRC Press, Boca Raton (2010)
50. Highsmith, J.: What is Agile Software Development? STSC Crosstalk, Journal of Defense Software Engineering (2002)
51. Ambler, S.W.: The Agile Unified Process, `http://www.ambysoft.com/unifiedprocess/agileUP.html`
52. Hu, X., Zeigler, B.P.: Model Continuity in the Design of Dynamic Distributed Real-Time System. IEEE Transactions on Systems, Man and Cybernetics - Part A 35(6), 867–878 (2005)
53. Carstairs, D.J.: Wanted: A New Test Approach for Military Net-centric Operations. Guest Editorial, ITEA Journal 26(3) (2005)
54. Sarjoughian, H., Zeigler, B.P., Hall, S.: A Layered Modeling and Simulation Architecture for Agent-Based System Development. Proceedings of IEEE 89(2), 201–213 (2001)
55. Mittal, S., Zeigler, B.P., Hammonds, P., Veena, M.: Network Simulation Environment for Evaluation and Benchmarking HLA/RTI Experiments. JITC report, Fort Huachuca (December 2004)
56. Hu, X., Zeigler, B.P., Mittal, S.: Dynamic Configuration in DEVS Component-based Modeling and Simulation. SIMULATION: Transaction of the Society of Modeling and Simulation International 81(2), 91–102 (2005)
57. Vigneras, P.: Why BPEL is not the holy grail of BPM. InfoQ (October 2008), `http://www.infoq.com/articles/bpelbpm`
58. Swenson, K.: BPEL: Who needs it anyway?, `http://www.bpm.com/bpel-who-needs-it.html`
59. Mittal, S.: Document Type Definition (DTD) for Web Service Workflow Formalism (2009), `http://duniptechnologies.com/binding/wswf.dtd`

Chapter 8
Systems Engineering and Conversational Agents

James O'Shea, Zuhair Bandar, and Keeley Crockett

School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Chester St., Manchester M15GD, United Kingdom
{j.d.oshea,z.bandar,k.crockett}@mmu.ac.uk

Abstract. This chapter describes Conversational Agents (CAs) in the context of Systems Engineering. A CA is a computer program which interacts with a user through natural language dialogue and provides some form of service. CA technology has two points of interest to systems engineers: the use of systems engineering techniques in CA research and the application of CAs in project development. CAs offer the opportunity to automate more complex applications than are feasible with conventional web interfaces. Currently such applications require a human expert in the domain to mediate between the user and the application. The CA effectively replaces the human expert. This chapter reviews the current capabilities of various CA technologies, outlines a development methodology for systems engineering practitioners interested in developing real world applications and suggests a number of directions for systems engineers who wish to participate in CA research.

Keywords: Conversational agent, systems engineering, dialogue, evaluation, methodology, semantic similarity, short text.

1 Introduction

In the early decades of computing ordinary people did not have any interaction with computers at all, any information that was needed for an application was entered by specialised clerks. This was followed by a period (largely driven by the internet) in which users could drive simple applications by entering simple facts and figures directly. Recently an explosion of internet use has demanded that ordinary people interact with complex applications, and the trend is for these to be of increasing complexity. Such applications would normally require a domain expert to interview the user to obtain the necessary information.

Automating such processes represents a serious challenge to computing practitioners. This chapter describes a possible way forward, Conversational Agents (CAs). CA technology has two points of interest to systems engineers: the use of systems engineering techniques in CA research and the application of CAs in project development.

A CA is a computer program which interacts with a user through natural language dialogue and provides some form of service.

Typically this dialogue system serves a business goal such as providing information, advice or selling. A suitably-designed CA plays the role of the human expert and is

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 201–232.
springerlink.com

generally in charge of the conversation; such CAs are fundamentally intelligence-based systems.

The idea of a computer taking the role of a human in conversation was first proposed by Alan Turing [1], as a test of machine thought. Although there has been substantial philosophical debate about the Turing Test [2-4], it has no impact on the validity of CAs. Practical CAs aspire to provide the user with the kind of advice or services that would come from a knowledgeable or experienced human, but in a purely behavioural sense. This form of CA presents with "Intentionality", that is it displays beliefs, desires and intentions concerning objects, events and states of affairs in the real world [5] – but it is not required to have a "mind."

The applications that have been proposed for practical CAs include health care dialogue systems [6], real estate sales [7], phone call routing [8], intelligent tutoring [9] and natural language interfaces to databases [10, 11].

Probably the most effort has been expended on CAs for online customer self-service, which provide the user with the kind of services that would come from a knowledgeable or experienced human. In 2005 there were at least 10 major companies operating in this area, including IBM and strategic partners of Microsoft [12]. At least 28 patents have been registered concerning CAs and closely related technologies. Despite this effort, success has been mixed and more research will be required to achieve the goal of a functional CA which can fill the role of a human [6].

2 The Scope of CAs

The term "CA" can have a very broad scope including:

- Spoken Dialogue Systems
- Chatterbots
- NLP-based Dialogue Management Systems
- Goal-Oriented CAs
- Embodied CAs

2.1 Spoken Dialogue Systems

Spoken Dialogue Systems (SDSs) are concerned with the conversion of speech into text. The average user might expect to interact with a CA by speaking to it directly and having the speech interpreted by SDS algorithms. In fact the field is insufficiently developed for this to be practical for anything but trivial applications. This is due to the relatively high error rates involved in converting the audio input into text. The performance of SDSs is usually measured as the Word Error Rate (WER) which takes account of the numbers of insertions, deletions and substitutions needed to correct a transcribed segment of speech [13].

Consequently, work on SDS systems falls into two categories.

The first covers systems which can convert speech from members of the general population and the second covers systems which are trained to recognise speech from a particular speaker.

Systems which cover the population split into two further categories, small vocabulary and large vocabulary.

Small vocabulary systems, in use since the 1990s for applications like paying bills, need to recognise the digits plus a few other words such as "account". These systems are capable of recognising tens of words and can achieve WERs of less than 1% under ideal conditions [14].

Large vocabulary systems contain tens of thousands of words [15] and represent the ideal interface for a CA. Unfortunately such systems have high word error rates, examples being a range of 18.4% -35.5% [16] and a particular WER of 25% [17].

Although small vocabulary systems are in routine use the individual words are simply matched as symbols and no real conversation takes place. More research is needed to improve the WER of larger vocabulary systems to make SDSs truly useful for CAs.

2.2 Chatterbots

Chatterbots are the direct outcome of attempts to create a system that would pass the Turing Test and are also stimulated by the Loebner prize which offers a substantial cash prize for passing a version of the test. The objective is to pass as a human for a limited period of time. Consequently chatterbots are programs that engage a human in social conversation and attempt to prolong the conversation for as long as possible.

Chatterbot development is driven by a "cat and mouse" game between developers and judges. The chatterbot is considered successful if it can prolong a conversation, no matter how banal or purposeless, for the time period without being detected as a machine by a judge.

The dominant technology in chatterbots is Pattern Matching. This approach requires scripts that define the conversation to be executed by a pattern-matching engine. The scripts contain rules which in turn contain patterns. The chatterbot responds to a user utterance based on the best match to one of its patterns.

Chatterbot developers program tricks into their systems to convince the user (as a substitute for real thought) and when users are in a judging mode they indulge in unnatural antisocial behaviour to "out" the chatterbot [18].

Although the technology behind chatterbots may be useful as a component of CAs, the chatterbot in itself is too limited to have use as a practical CA.

2.3 Natural Language Processing Based Dialogue Management Systems

Natural Language Processing (NLP) is largely concerned with document retrieval, information extraction, and text categorisation [19].

There is an established interest [20] in applying established NLP procedures such as using parsing, keyword extraction and formation of a structured lexicon to systems which engage in dialogue.

In initial work there was a lack of substance when it came to reasoning about the meaning of user utterances and the production of relevant responses. Modularisation (or compartmentalisation) of NLP based systems leads to these problems being lumped together as Natural Language Understanding (NLU).

The dominant approach to NLU is the frame-based system [21-23]. This is effective for simple applications such as making bookings for journeys or theatre seats. A related approach is the use of state-based systems, popular in healthcare [6].

These undergo state transitions triggered by the content of user utterances. Some success has been achieved with limited systems in which tight constraints are placed on the utterances that the users can produce. This can be done with forced choice questions (e.g. yes or no answers) or the detection of a very restricted set of highly salient speech fragments; however the dialogue may be unnatural. More flexible dialogue is possible, using more powerful grammars and probabilistic/empirical techniques, but is not trusted when high accuracy of understanding of the user intent is required [6].

The most promising NLP-based approach (used within a CA) currently being investigated, at the University of Cambridge, uses phrasal grammar rules to extract the dialogue act type and a list of attribute/value pairs from each utterance and a template-based response generator [24, 25]. However, this approach has only been evaluated in the laboratory, with a simple domain, Towninfo, which recommends restaurants, museums and similar destinations of interest to tourists.

Despite the considerable effort put into NLP, it has a number of problems for use in real-time CAs. The first is whether the chains of computationally intensive processes involved will scale up to real-word applications deployed on the web, especially when large numbers of users are involved. Secondly, each process has a particular error rate and the cumulative effect of these may affect the classification of user utterances. Thirdly, NLP relies on grammatically correct sentences, yet most user utterances are not properly-formed. Repair processes to remedy this incur a further computational overhead. Fourthly, research into NLP is fragmentary in nature. For example recent work has focussed on monitoring the human's engagement [26, 27], interaction control [28, 29] or determining if a party is being addressed [30]. What is really required is a concerted effort to produce a less sophisticated, but functional, system.

2.4 Goal-Oriented CAs

A Goal-Oriented CA has a deep strategic purpose in holding the conversation and its design incorporates mechanisms that enable it to focus the conversation on achieving a goal. This is what distinguishes it from a Chatterbot.

The original design objective of chatterbots was to prolong social chit-chat, thus they are easily de-railed by human users when used for practical applications.

A Goal-Oriented CA (GO-CA), on the other hand, is specifically designed to interact with a human, using natural language dialogue, to achieve a particular business goal - such as providing information, advice or selling. It plays the role of an empowered human in a productive application or task. Thus the GO-CA [31] may spend more time leading the conversation and asking questions than the human.

In general terms the human approaches the GO-CA with a problem or need. In current implementations [31] a pattern matching dialogue front end is combined with a rule-based system, which contains a model of the problem domain that is expressed in terms of a set of attributes. Through the process of dialogue, appropriate attributes are captured to model the particular problem experienced by the user and identify the appropriate solution.

The GO-CA is a mixed-initiative system (from time to time either the human or the agent may take control of the conversation). Due to the goal-oriented nature of the

agent it will take the initiative in the first instance and will always return to the goal after the human has diverted the conversation (for example to ask for a clarification of something said by the agent).

Figure 1 shows the generic architecture for a typical GO-CA [31]. This is intended to take on challenging real-world applications in which the human user may present adversarial, disruptive or deceptive behaviour at times during the conversation.

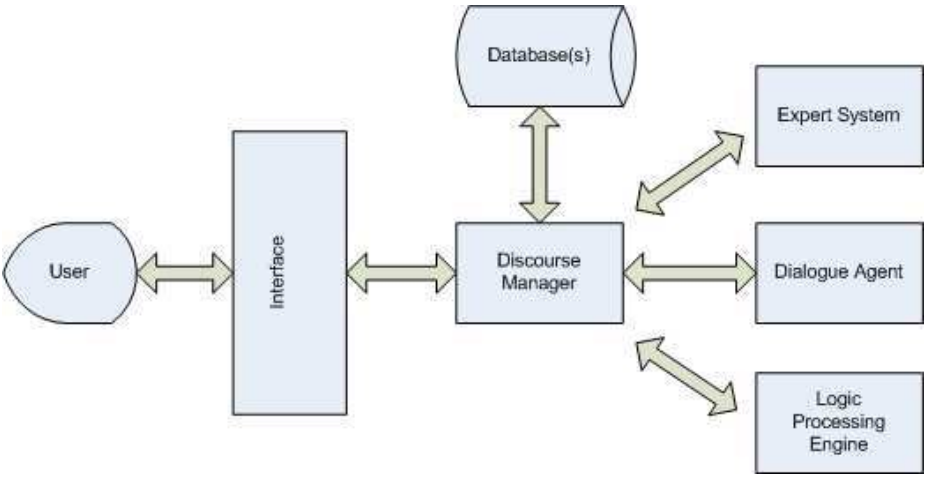


Fig. 1. Goal-Oriented CA architecture

Modularisation is an important element of the architecture. New modules can be added for extended functionality or existing ones omitted if not required. In this particular CA the rule-based system takes the form of an expert system. In another the application required an interface that allowed the agent and the users to communicate by SMS text messaging.

The architecture is best described by considering a dialogue transaction between the agent and a user.

1. The first step in a dialogue transaction is for the expert system to identify the attribute (or next attribute) whose value is to be captured.
2. The expert system passes this requirement to the Discourse Manager (DM).
3. The DM invokes the dialogue agent (DA) which produces a suitable utterance for the agent and returns this to the DM.
4. The DM passes the agent's utterance to the web interface which sends it to the user.
5. The user replies to the agent and the web interface passes the user utterance to the DM.
6. The DM invokes the DA to determine if it contains the required attribute.
7. If the attribute has not been captured, go to step 3.
8. When the attribute has been captured, the discourse agent passes the attribute to the expert system, which updates its model.

9. If further attributes are required by the expert system, go to step 1.
10. The solution to the problem is communicated to the user.
11. At this stage the user may end the conversation or continue by asking clarification questions.

The interface can consist of a text-based, instant messaging-style system which is very familiar to users of social networking applications. It is also possible to use a speech recognition system (currently this would need to be trained for a specific user; large vocabulary word recognition may be feasible in the future).

The database is used for long-term storage of user attributes (e.g. start date with employer) and the logic processing engine provides domain-specific computational tasks (e.g. date calculations).

The common feature between GO-CAs and chatterbots is the prevalence of the technique of pattern matching. However a GO-CA will engage in extended dialogue, during the course of which it will appear to have mental states that are directed at states of affairs in the world e.g. beliefs, desires, hopes, perception, intention etc. Whereas chatterbot-based systems typically present a business's FAQ list with a human face, GO-CAs are intended to give sophisticated advice on topics such as bullying and harassment in the workplace [31].

2.5 Embodied CAs

An Embodied CA (ECA) is characterised by a multimodal interface which includes a facial display, hand gestures, posture etc., interaction with a human (or representation of a human in a computer environment) and a dialogue system where both verbal and nonverbal devices advance and regulate the dialogue between the user and the computer [32].

The degree of embodiment can vary considerably. At its simplest it involves a graphic representation of the agent capable of facial expressions, where the intention is to provide a generally heightened sense of realism. One example of this approach is a virtual museum guide used to investigate the kind of dialogue that embodied agents provoke from humans [33]. The most advanced view of embodiment encompasses facial expressions and gestures by the agent coupled with the reading of gestures from the user [32, 34, 35]. This extends to the modelling of emotions on the part of the agent [35].

Whilst there is clear potential for embodiment to improve GO-CAs, for example through disambiguating pronouns such as *this* and *that* using pointing gestures and shared visual space [32, 36], the dialogue tasks attempted remain relatively simple.

The REA agent [37] uses chatterbot-style social engagement combined with a linear attempt to collect a very small number of attributes in order to make a property recommendation. ECAs are generally used with very low-stakes applications, where there is little to lose if the agent fails to operate correctly, for example museum guides [33, 34, 38].

Research continues in this field, but tends to follow fragmentary and specialised interests rather than focussing on the holistic problem of building a GO-CA for real-world applications [27, 29] [39].

3 Practical Applications of CAs

The primary interest of this chapter is the development of CAs for real-world applications. There are two requirements that a CA technology must meet to achieve this:

- It must be capable of handling extended dialogue about complex tasks
- It must be capable of operating in real-time when deployed over the internet.

We begin this section with an overview of the use of the various forms of CA in selling. This is followed by a description of a more challenging application where the stakes are higher, leading to more complex user behaviour.

3.1 CAs for Selling

Selling is an obviously useful activity and customers often have difficulty in making choices about a purchase, requiring assistance from a human sales person.

The move to selling over the internet reduces costs but removes this human element. A CA can rectify this problem. Selling is a challenge that can be met at a number of different levels of complexity, which makes it a useful vehicle for comparing CAs. Also, in its more challenging forms it creates work for practitioner systems engineers which has a clear value for their business clients.

From the systems engineering research point of view it is sufficiently demanding as a test of new theories, processes and tools, and it has been attempted across the scope of CAs described above, from SDSs to embodied CAs.

Influential early work in selling using SDS was stimulated by the DARPA air travel challenge [40]. Attributes collected from the user were origin, destination, date (of flight), time (of flight), airline, trip-type (one-way or return), hotel and car. The attributes were collected in a linear fashion and there was no interaction between them, apart from asking whether ground arrangements were required before asking about the hotel and car requirements [41]. Walker also used a restaurant booking application to investigate multi-attribute decision making. In this case the restaurants were modelled using 6 attributes: *food quality, cost, decor*, *service, food type* and *neighbourhood*. The attributes were scored on a scale and the combination of scores was used to make recommendations [42].

Recent work on SDS introduces more sophisticated processing of the user utterances, for example with more support of user barge-ins (interruptions) [43]. But current work confirms the limited number of attributes that can be managed using an SDS system [44].

The current position is that the SDS approach is limited to transactions that require a small number of factual answers and these are characterised by situations where the user has already made the decision to buy, for example in booking travel arrangements. Real-world systems have been deployed effectively for the payment of utility bills where only recognition of digits and a few additional words is required to make a payment.

SDSs currently fail to meet the requirement to handle more extended dialogue, with the general population, principally because of the high word error rate with large vocabularies. Current SDS technology does not achieve a suitable word error rate for

speech from the general population for the size of vocabulary needed for more sophisticated applications.

A significant number of chatterbot systems have been developed for selling over the past couple of decades. Furthermore, the chatterbot is the only form of agent to have been deployed on large companies' websites for real interaction with the general public. Examples include Hank (Coca Cola), Kate (Ford) and Anna (IKEA) [45]. Of these agents Hank and Kate have failed to prosper. They have been replaced on the websites by FAQ systems with question entry boxes and these do not encourage the user to enter into dialogue. The IKEA bot, Anna, still has a visible presence on the IKEA website. Anna presents as a 2-D cartoon head and shoulders figure. She is a sales assistant who accepts typed input and replies with text, which is also spoken by a female voice synthesiser. The agent blinks, smiles and makes occasional posture changes but can not be considered as truly embodied as these gestures have no significance in the conversation. In terms of dialogue Anna is extremely limited, simply pointing customers to the IKEA catalogue, for example:

```
User: I'm having a barbecue

Anna: I'm really not sure what it is you're trying to
say. Can you please try and re-phrase your question or
statement.

User: do you sell barbecue accessories

Anna: EHere you will find the Decoration Accessories
Category.

User: do you sell plastic glasses

Anna: Please have a look at the Glasses Subcategory.
```

Under these circumstances a human sales agent could have told an anecdote about organising a barbecue, gone through specific examples of the utensils required and prompted the purchase of additional items, so clearly an opportunity has been missed.

Selling chatterbots are still under development, a recent example being Susan, hosted on the Kegel Motorcycles website [46]. Susan is described in the chatbots.org site as ". . . an attractive, smart and bright cowgirl who not only talks with clients but also presents multimedia." Susan appears on the website as more conservatively dressed and a little more sober in manner than the original description. Susan is composed from video clips of a real woman speaking the dialogue. Susan uses a few gestures (pointing to self, pointing to menus, dialogue box etc.) and expresses boredom if there are pauses in the conversation (for example by tapping on the glass of the monitor as if it were a window). Despite appearances, the behaviour is quite superficial and is a long way from qualifying as genuine embodiment.

Furthermore, the interaction hardly qualifies as dialogue because it is so constrained. The opportunity to exploit social chat in the selling has been sacrificed to achieve robustness. Susan repeatedly refers the user to a menu below the dialogue

box, there is an overall menu of topics covered by the side of the agent and during typing an auto-correct style "did you mean . . ." steers the user towards one of Susan's standard replies. For example:

```
User: I'm 55, what sort of bike should I get?

Susan Try the menu buttons on the left
```

The menu displayed to accompany this includes much off-topic information such as company history, financing deals etc.

Real-world applications require the capture and analysis of attributes from the user's utterances. It should be noted that neither Anna nor Susan do this to any noticeable degree and are thus less analytical than the SDS DARPA travel systems.

The one strength of the chatterbot is social chat. The original instantiation of Susan seemed to encourage this as the video clips were of a pretty, flirtatious cowgirl. Whilst this is probably consistent with the brand image required for a motorcycle vendor, it also underlines the weakness of chatterbot technology. Experience in developing CAs shows that users are likely to be attracted to this kind of site for two reasons. The first is to flirt with the chatterbot. The second, particularly with such a human representation, is to test it to breaking point (much like the behaviour of Loebner prize judges).

Social chat creates a paradox at the heart of chatterbot systems. On one hand the social chat capabilities improve the user's experience, which ought to make chatterbots more effective as sales agents. On the other hand purposeless chat does not progress the conversation to achieve the client's objectives.

Conventional chatterbots are not equipped with the necessary mechanisms to achieve both requirements, hence the development of the GO-CA.

Selling has been of interest to the NLP based community for the past two decades as exemplified by the SCHISMA theatre bookings project [47]. Schisma projects began with text-based interfaces and an intention to move to SDS interfaces "in the near future" [48]. Progress may be judged by recent work which combines NLP techniques with an SDS interface. The NLP element is sophisticated, involving semantic parsing, context free grammar and dialogue act recognition. The SDS component however, is only capable of recognising 263 words (trained using data from British speakers).

The theatre booking task is a little more sophisticated than the SDS systems described earlier. Attributes to be captured include actors, authors, performances (category theatre/opera as well as title), dates and venues. This entails the recognition of data types such as number, date, time etc. in the utterance, which is performed with error correction before the parsing stage.

Although this work is interesting, it is only partially implemented and the research still makes use of the Wizard of Oz approach (in which a human simulates the CA) for collecting corpus data.

Of all the techniques, one might have expected NLP to produce working, robust, systems - however these do not exist in the real world. There are two principal reasons for the failure of NLP to produce. First, NLP requires chains of processes such as stemming, pos-tagging, syntactical repair and parsing which can lead to

cumulative errors in recognising a user utterance. Disambiguation is a further problem as the usages and senses of English words are not easily identified. To make things worse, as observed by Donald Michie, "Real chat utterances are mostly unparsable. They are concerned with associative exchange of mental images. They respond to contextual relevance rather than to logical or linguistic links."[49].

The second problem is the high computational complexity of NLP processes which raises serious questions about its scalability for a CA deployed over the web serving multiple users. For example the UK national flu service received 9.3 million hits per hour on the first day of operation (resulting in it crashing even though this was a simple menu-based system)[50].

GO-CAs have also been developed for selling; one such is VSA [51]. The system is described as dialectical and goals are driven through an internal process of argument about logical formalisations of the dialogue. The sales process is divided into 3 phases, before sale (identifying needs and suitable products), sale (negotiation to make a deal) and after sale (which was not dealt with in Morge's study). The scope of the system is not clear, but one example dialogue in which a quilt is sold has 4 attributes, allows the user to barge-in, volunteering information and also allows the user to question the agent. Although the agent was developed using commercial software, there is no indication of any real-world deployment.

Although the chatterbots like Anna and Susan present a human face, these are not genuinely embodied. The distinguishing feature of embodied CAs is that the gestures and expressions are purposeful in contributing channels of information to the dialogue. An early and well-known ECA is REA, the real estate sales agent [32]. REA is indicative of the greater interest in embodiment than dialogue in ECA research. A typical study using REA captured 3 pieces of information in a linear sequence: the city the user wanted to live in, the number of bedrooms desired and acceptable level of rent. The objective of the study was to investigate whether the use of social language fostered trust in the agent on the part of the user. Current work continues to take a strong interest in social behaviour, such as the display of emotions [52], rather than deployable systems.

ECAs are still relatively immature. They build embodiment on top of existing dialogue management techniques and inherit their weaknesses. They have not been used to tackle the kinds of applications of interest in this chapter and consequently their potential advantage (disambiguation through gesture) has not been put to a serious test. The leaves the GO-CA is the most currently promising technology for developing real-world applications. The following section describes a GO-CA which ran for 8 years, advising university students about debt problems.

3.2 A GO-CA Student Debt Advisor

Adam, the student debt advisor, operated from 2002 to 2010. Adam was designed for a very specific application, to assist a student who had received a warning letter about debt to the university. An important distinction between Adam and a chatterbot system is that Adam was not intended to counsel students about their feelings about debt, rather to follow the steps necessary to pay the debt off.

Adam's rule-based system uses 23 different debt-based attributes, as well as collecting some information not used for decision making, such as the student name.

By analysing combinations of the attributes, Adam directs the student to one of 26 different actions to solve the debt problem. Analysis is performed by a decision tree, so only the subset attributes required to traverse from the root to the appropriate leaf need to be collected. There are many different routes through the tree with corresponding dialogue structures.

One outcome of the knowledge engineering phase is that the tree is designed to process and complete the most frequent solutions first, reducing the computational load on servers in the deployed system.

For example, in the previous telephone-based system, one of the most common calls was from students who had paid off the debt and wanted assurance that no further proceedings would be taken. This situation is caught with the attribute Have_paid_already in the first conversational context.

The many conversational contexts within the Dialogue Manager are decomposed into 4 groups:

- Conventional dialogue
- Filter
- Oracular Layer
- Aliza Layer

The contexts making up the Conventional Dialogue put the questions to the user to acquire attributes, perform clarification tasks and provide the instructions (diagnosis).

The filter is executed every time the user types an utterance, regardless of the current context in the main dialogue. It performs two tasks. First, it contains a small number of rules to detect highly obvious statements of the values of any of the 23 attributes. So if a user volunteers additional attributes in the current context, they will still be recognised and captured. Secondly it is used to detect racist or other highly offensive language in the conversation (which results in the conversation being terminated).

The oracular layer is responsible for answering questions put to the system. These include the many possible requests for information or clarification required to supply the attribute values to Adam, as well as general questions.

The Aliza layer (named for its resemblance to the Eliza chatterbot) layer contains general chat. This allows Adam to respond to social remarks included in the conversation. It also includes light-hearted responses to personal, challenging or antisocial remarks by the user.

A user utterance is first passed through the Filter and then (if the desired attribute is not captured) to the Conventional Dialogue layer. If the conventional dialogue layer does not have rules that can process the utterance it is analysed to see if it is a question, if so the utterance is passed to the Oracular layer, if not it is passed to the Aliza layer.

Adam has one simulated emotion, irritation. This can build up or dissipate over a number of dialogue transactions. If a certain level is reached Adam will terminate the conversation. This can happen very quickly (in the case of extremely offensive language from the user) or more gradually with warnings (in the case of the user failing to co-operate with Adam's conversational strategy).

Having illustrated the power of the GO-CA, the following section describes a methodology for developing them.

4 Design Methodology for GO-CAs

The software development methodology for the GO-CA combines elements of the staged approach used in the Waterfall model with elements of prototyping or iterative development. The major stages are shown in figure 2.

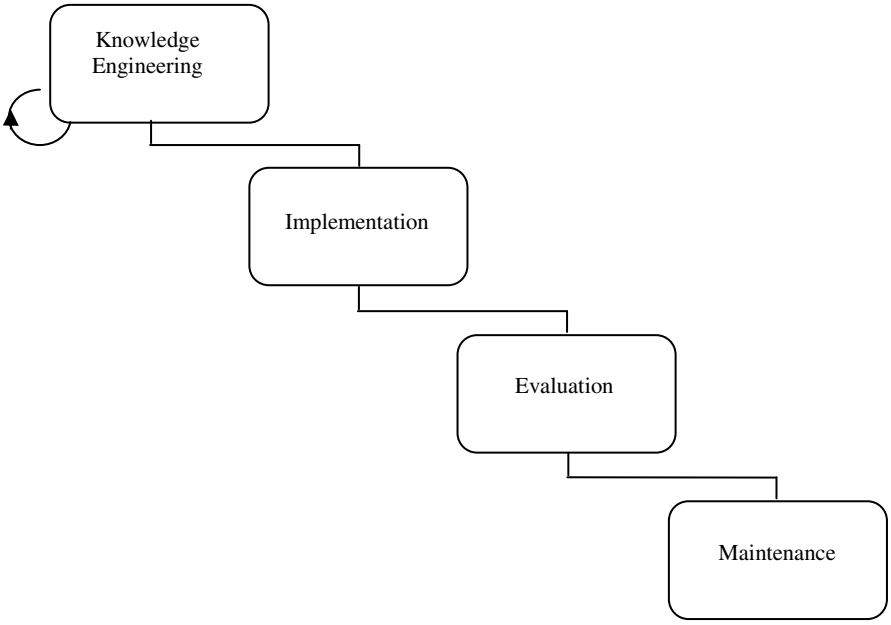


Fig. 2. Design Methodology for GO-CAs

4.1 Knowledge Engineering

Knowledge usually extracts information about a domain from many different sources, including:

- Managers in the client organisation
- Practitioners in the client organisation who interact with the customers who will use the CA being developed
- Documented procedures of the client organisation (e.g. workflow charts)
- 3rd party websites (e.g. government legislation concerning the domain)
- Telephone logs of customer calls related to the domain.

Many organisations do not have formalised processes, instead custom and practice is handed from one generation of employees to another in an oral tradition. This can lead to a reliance on a small number of key individuals in an organisation. Where an organisation has a formalised process for the problem domain, this may not be followed in practice for a variety of reasons e.g. experience and gut feelings,

reluctance to change or simply lack of knowledge. So a highly important aspect of knowledge engineering is to formalise the process in the first case or to establish exactly what process should be followed in the second.

To achieve this, a series of meetings is held with key personnel in the organisation who are most knowledgeable about the process and if appropriate, some meetings with the organisation's customers are held.

The processes are refined and the problem domain is expressed in a structured way such as a rule-based system. Figure 3 shows a section of a decision tree rule-based system for bullying and harassment.

Fig. 3. Bullying and harassment decision tree example

The constructed rule-based system will be validated using a walk-through process and then approved as a true representation of the domain before it is implemented.

When the rule-based system is implemented as an executable program, it is interfaced with the DM and provides the logic for the CA.

A copy of the knowledge base, which includes the rules and other knowledge acquired during the process will be passed on to the scripters, for example particular terminology used in the domain.

The rule base provides information, in terms of its size and complexity, about the overall effort required to construct the CA and is a useful recourse for cost and effort estimation (similar to function point analysis).

4.2 Implementation

All of the components in figure 1 require implementation effort to varying degrees. The user interface is an area where substantial quantities of library code may already exist. In this case effort is restricted to relatively simple tasks in laying out the radio buttons, text boxes etc. Likewise if the application requires a database it is likely to exist already within the organisation (e.g. personnel database), in which case most of the effort involves interfacing it to the system. The logic processing engine may also be able to benefit from previously existing code, for example calculating the number of days expired between two calendar dates is quite a common task.

The rule-based system is the product of the knowledge engineering stage described previously. This is often implemented as high-level code executed on a generic engine. The engine is likely to be a stable piece of code which is re-usable between applications. Rarely, an application may introduce some new demand which requires an engine modification.

Each application will require a customised DM. The DM is the glue that binds the other components of the CA together. Its principal function is to maintain the list of attributes required by the rule-based system. As each new attribute is requested by the system, the DM takes control, continually re-entering the DA until a transaction between the agent and the user captures the required attribute. So the DM is unique to each application, but is relatively small in size.

The bulk of development effort is spent on the DA. The DA is implemented using a scripting language which executes on an interpreting engine.

4.3 Scripting Language

Most scripts use the pattern matching technique, which has been identified as one of the most common and capable methods for developing dialogues that seem to be coherent and intelligent to users [6]. One of the most influential pattern matching approaches was published widely during the 1990s [53] in "The Zen of Scripting Verbots." This stimulated the production of a number of scripting languages [54] in use today.

A script file consists of rules each of which contains a number of patterns and a response, executed by an engine, for example the DA in figure 1. A user utterance is compared with the patterns in a rule and a numerical activation is calculated (the better the match, the higher the activation). When all of the rules have been processed, the rule with the highest activation fires and its response is used to reply to the user. A threshold can be set for the activation, below which a rule can not fire. If no rule exceeds the activation a default rule will recover the thread of discussion. If useful information is generated as a result of the rule firing it is passed to other programs making up the agent for relevant action.

The rules will often be divided into contexts to make them more manageable, corresponding to modularisation of conventional code. An application may require many script files covering the various conversational contexts that may occur and there are also mechanisms for switching contexts.

Every time the user types an utterance, every pattern in every rule in the current context must be tested and each test requires multiple passes through the utterance and the pattern. If a user utterance fires a rule that switches to a new context, then the whole process may have to be repeated until there is a match. Pattern matching is a computationally intensive process and producing scalable CAs depends on skilled context design by the people who write the scripts as well as the software engineers who create the engines. The technique is illustrated by the following example, adapted from Plantec [53]:

Consider a pair of activation-based rules:

```
<what-work>
a:0.5
p:60 What *your*job*
```

```
p:60 How *earn* living*
r:I'm a full time Verbot
+:<explain>

<explain>
a:0.5
p:60 What *you* mean*
p:60 *Eh*
p:60 *explain*
r: I am a computer program that chats with you.
```

Suppose the first user utterance is

```
"What is your job?"
```

The engine will begin by comparing it with the first pattern in the first rule:

What *your* job*

In brief the "What" sections of both strings match, but the following "is" and "your" do not. However the wildcard * is able to absorb the "is" and matching continues for the "your" substring. Wildcards are allowed to match nothing so the terminal * on "your" is ignored and the substring "job" matches. Finally the "?" symbol on the end of "job" in the utterance is absorbed by the terminal * in the pattern. This is a match which generates a positive numerical score. This score also depends on how close the match between the utterance and the pattern is (for example how much of the utterance the wildcards have to absorb).

The second pattern in the first rule is then tested and it falls at the first post, because the "H" in How fails to match the "W" in what and there are no leading wildcards to accommodate the difference.

When the second rule is processed the first pattern will begin by matching but will fail at the point where "job" is compared to "mean." The process continues for the remaining patterns.

So the first rule fires (wins) and the agent will reply

```
"I'm a full time Verbot."
```

At this point the user will make another utterance. If there were no matches with contextually meaningful rules, the script could fall back on a general rule that would reply with something like:

```
"What on earth you mean by that?"
"Eh?"
```

or

```
"Could you explain that for me please?"
```

In a large base of rules the corresponding patterns (or variants on them) could occur many times so a promotion mechanism is used to ensure that the correct rule fires.

The entry +:<explain> at the end of the first rule temporarily boosts the activation for the rule <explain> for the next few utterances that the user makes. This feature is known as promotion and the complementary process which temporarily cuts the activation, -: is known as demotion. Various decay functions can be set to govern

the return to the original activation over successive utterances in the conversation. This gives some feel for the complexity that can be involved in debugging a large CA.

Other features of scripting include the large range of tuneable parameters for example:

a:0.5 sets a base activation value for the rule, which is principally used to allow one of the rules to fire when matched.

p:60 sets an activation strength for an individual pattern; this is used to prioritise patterns within a rule or instances of a pattern when it appears several times in different rules.

Part of the success of pattern matching may be attributed to its abilities in feature extraction and forming associations. Two particular mechanisms allow it to mimic (to a limited extent) properties of human consciousness. The first is the promotion / demotion of selected rules. This simulates the "stream of consciousness" in which a particular thought comes into the foreground of consciousness against a background of other partially-activated thoughts and ideas [55]. The second is the organisation of rules into contexts (so named because they correspond to a particular conversational context). This allows the agent to focus its attention on a particular topic and prevents misfiring of rules that would correspond to a human being distracted. For example, if a user introduces the name Pluto into a conversation the context could be astronomy or it could be Mickey Mouse's dog. If each of these topics has a corresponding context, when one is being executed by the DA the rules in the other (and all other contexts) are inaccessible.

Creating scripts is a highly skilled craft [49], requiring the anticipation of user utterances, generation of permutations of the utterances and generalisation of patterns through the replacement of selected terms by wild cards. Modifications to rules containing the patterns can impact on the performance of other rules and modern pattern matching systems contain many parameters that further modify their behaviour.

The main strengths of the technique described in this section are:

- It works well within its limits and it's about the only technique that currently works at all for extended dialogues.
- The computational engines for such systems are well-developed and robust; they are rarely crashed by unexpected user input.
- The scripting method separates out language skills from coding skills. People with language skills can become scripters without learning a great deal of computer science.

However, it also suffers from a number of weaknesses, some of which are:

- Writing patterns which match user inputs effectively is a labour intensive process and the scripters must be highly skilled at selecting key words or phrases and integrating them with wildcards.
- The CA's responses to the user must also be crafted to maintain the dialogue along predictable lines. Transactions which are plausible in isolation can be stilted or incoherent as a complete conversation.
- The organisation of rules into coherent contexts involves another set of skills, similar to the design of coherent modules in conventional programming. Failure

to do this results in systems that are difficult to test and debug. They are also easily destabilised by the addition of a single rule.

These drawbacks have an impact on development costs, maintainability and scalability.

However, on the important issue of scalability, pattern matching systems do not require pre-processing stages such as stemming, pos-tagging, syntactical repair and parsing. This is an important consideration as a real-world system could have millions of instances of the CA running simultaneously on the organisation's servers.

4.4 Evaluation

It is possible to evaluate the rule-based system before the rest of the CA is constructed. This is performed using a technique known as "Wizard of Oz" in which a human simulates the CA interface and operates the rule-based system [56]. Users believe they are talking to a real CA because the wizard is hidden by the interface. All of the "agent's" dialogue is provided by the wizard, who extracts the attributes for the rule-based system.

A substantial amount of work has been done on evaluating agents as a whole. The seminal work in this area was the creation of the PARADISE framework [57] which was applied to evaluate the DARPA communicator SDS [58]. An important feature of PARADISE is the application of linear regression for deriving abstract, indirect attributes such as User Satisfaction in terms of directly measurable attributes [59].

The PARADISE framework continues to provide the framework for current evaluation of CAs including further development of evaluation methodologies [60], CAs for navigation [61], dialogue management strategies [62], tutoring [56] (in press), human-robot dialogue [63] and companion agents [64].
The metrics used to evaluate CAs can be broken down into 3 categories:

- Aspirational Subjective Measures
- Attempted Subjective Measures
- Objective Measures

Prima facie, the subjective measures are important, but they are also difficult to measure with the scientific rigour one might expect of a physical variable such as voltage.

4.4.1 Aspirational Subjective Measures

A number of publications discuss very high level, abstract and subjective concepts which would be very difficult to measure as a single attribute. The most common attributes are:

- Usability [40, 56, 65-69]
- User satisfaction [56, 65, 68, 70-72]
- Agent credibility [70, 73, 74]

The first two are common but difficult to measure attributes from the field of software engineering [59]. There are many more intangible and vague attributes mentioned in studies, including: "Fun to talk with" [38], "lovely, pleasant, black humorous" [70],

"Intimacy, Benevolence" [37], "Comfort, Solidarity, Familiarity" [75] and "Trust, Uncertainty, Attractive" [66].

4.4.2 Attempted Subjective Measures

Some studies then go on to attempt to measure a subset of subjective attributes. These are largely measured using Likert or Likert-like attitude rating scales. Attributes measured in this way include:

- Ease of use / Task ease [56, 58, 61, 63, 69, 76-78]
- Ease of the user understanding the agent [56, 58, 63, 66, 76, 77]
- The agent's understanding of the user comprehension [56, 65, 66, 77]
- Various cognitive attributes related to comprehension and complexity [56, 61, 65, 67, 73, 76]
- Various attributes related to the reliability of the agent and the ease of correcting misunderstandings [65, 76]
- Various attributes concerning the user's expertise (of the domain or using the agent) [58, 66, 67, 77]
- The efficiency or effectiveness of the agent [56, 63, 65-67, 76, 78]
- Various attributes about command and control of the conversation [65, 73, 76] [66, 67]
- The pace of the interaction [76, 77]
- Whether the agent behaved as expected [56, 58, 61, 64, 65, 77]
- How natural the agent's behaviour seemed [61, 73, 75, 78]
- Various positive emotional attributes (e.g. friendliness, enjoyment) [65, 66] [63, 75, 76]
- Various negative emotional attributes (e.g. boredom, fluster) [63, 65, 76]
- Whether the user would use again [58, 65, 76, 77] or prefer human service [56, 61, 76]

There are also a substantial number of attributes which occur once or twice including "like further help" [65], "narrative skills" [70], "needs improvement" [76], "question answering capability" [70] and "how much willing to pay" [37].

4.4.3 Objective Measures

Most studies include a set of objective measures. Generally speaking, there is a leap of faith that these in some way reflect the aspirational subjective measures that appear at the beginning of published studies. The only systematic and scientific approach was that taken by the PARADISE framework [58]. Attributes measured in this way include:

- Dialogue / Conversation length [38, 40, 58, 61, 63, 65, 72, 73]
- Count of dialogue turns [9, 40, 58, 61, 63, 72, 77-79]
- Various measures of success at utterance or task completion level [58, 61, 63, 65, 68, 72, 79]
- Various counts of errors, corrections or percentage error rates [6, 38, 61, 63, 64, 71, 77, 79]

- Various counts of correct actions by the agent (e.g. answering questions) [58, 66, 67, 70]
- Various speech recognition accuracy measures [9, 74, 76]

Again there are a substantial number of attributes which occur once or twice including "mental workload" [72], "learning gains" (in a tutoring system) [9], count of help messages [76], percentage of time user spent looking at (embodied) agent [67] and user trust of agent (using a standardised measure from psychology) [37].

All recent work makes use of some of the fundamental PARADISE measures whilst adding some application-specific elements. For example, companion agents add more emotional evaluation including the nature of the relationship between the user and the companion, and the appearance, personality, emotion demonstrated and social attitudes of the companion [64].

4.5 Maintenance

CAs require maintenance in the same manner as any other form of software, however their nature dictates that at the current state of the art they require it to a greater degree. There are 3 drivers of the maintenance requirement, these may be termed:

- Conventional bugs
- The user paradigm shift
- Domain stability

4.5.1 Conventional Bugs

Conventional bugs are the same as those in other software; these may be syntax errors or logic errors. They can occur in the three classes of software making up the agent, the rule-based system, the dialogue scripts and the engines. Most errors in the rule-based system will be discovered and removed during the latter stages of the Knowledge Engineering process. However, as with any software development process some will get through.

Once a development team has a set of established engines (such as the DA in figure 1), they will become a relatively infrequent source of bugs and the usual pattern is established of bugs arising largely as a matter of upgrades to the functionality of an engine. As the dialogue scripts are specifically written for each application they differ from previous code and are more likely to provide a source of bugs. Detecting and correcting conventional bugs is a very familiar process for systems engineers, consequently the other two classes pose the greater challenge for CA developers.

4.5.2 The User Paradigm Shift

Knowledge engineering is an excellent way of capturing existing processes for the production of a CA. However, it does capture the *status quo*, in which user behaviours are shaped by the social interaction with a human expert and the understanding that the expert will have specific expectations of the conversation. However, when faced with a web-based CA the human users behave differently.

One example is the student debt advice system developed [31]. This system was designed to steer students through the process of obtaining the money to pay off their debts to the university by chasing late payments from the student loan company,

seeking alternative sources of access funding etc. However, after the system was deployed on the web, logs showed a significant number of students accessing it before starting university to find ways to avoid getting into debt.

This immediately faces the maintainer with decisions, whether to cater for the changed demands from the users, if so to what degree, and how much cost and effort can be justified for implementation.

Considering the goal-oriented architecture, a limited change could be accommodated by adding to the scripts executed by the DM whereas a more extensive change would also require the creation of new attributes processed by the rule-based system. Changing both creates more maintenance effort (modifications to the DM would also be required to communicate the attributes between the DM and rule-based system). Consequently the temptation is to limit the changes to rules in the scripts, but relying too much on scripted rules without the discipline of the rule-based system leads to unstable behaviour of the kind exhibited by poorly-performing chatterbots.

4.5.3 Domain Stability

By far the most serious problems arise from domain instability. Domain instability refers to the rate of change in the environment in which the CA operates. There are a number of sources of domain instability. The client commissioning an agent operates in some form of market. When the agent is completed and released into the market all may be well, but markets change over time affecting the relevance of the agent. Another very important source of change is legislation. Many agents will be required to comply with (or explain) requirements of government legislation. Unless the agent is kept up-to-date again it will become less relevant and will generate fewer satisfactory outcomes from the dialogues.

One particular example of this is the UK student loan system which was fundamental to the student debt advisor. This system has changed virtually every year following changes of policy and changes of government.

How may these maintenance challenges be met? One way to tackle the volume of work in maintaining pattern matching systems is to improve the efficiency of the laborious process of hand-crafting the rules in the scripts. Tools have been developed [31] to automate pattern generation and process standardised dialogue templates by a fill-in-the-blanks approach. Other possible tools under consideration include a conflict detector to find conflicts between similar patterns in different rules. This would solve the common problem of adding new rules which overlap with existing rules and conflict with them in firing. Also, current testing requires manually typing utterances into the agent and checking the responses. A regression testing tool containing its own scripts of user utterances with expected outputs from the agents could make this process more efficient. Again regression testing is important to ensure that changes to an agent have not introduced new problems. Test data for regression testing may be accumulated from logs of earlier user evaluations or from conversational scenarios created to design the contexts.

The second approach is to develop an entirely new method of matching user utterances, which does not require the generation of large numbers of patterns. One promising technique, Short Text Semantic Similarity is described in the following section.

5 Novel Algorithms – Short Text Semantic Similarity

The potential for Short Text Semantic Similarity (STSS) algorithms to improve CAs arises from their replacement of the pattern matching component. Suppose an STSS algorithm produces a numerical measure of semantic similarity, this could be used to make judgments such as:

- A pair of STs is identical in meaning
- A pair of STs is completely unrelated in meaning
- One pair of STs is more similar in meaning than another.

Consequently an incoming user utterance could be compared with a number of prototype statements from the domain and an appropriate action and response chosen based on the value of the best match. Consider the following patterns, taken from a rule in a student debt advisor system:

```
p:15 *can*not *afford *pay*
p:15 *can*not *afford *full amount*
p:15 *<problem>* pay*
p:15 . . . . . .. (many more)
```

These patterns will match utterances such as:

I cannot afford to pay you anything this term
I can't afford the full amount but I could manage to pay a third
There is a difficulty in paying because I was mugged
(amongst many others).

It is clear that even with wildcards for generalisation, many patterns will be needed for good coverage of the overall conversational space. Also there will be a need for skilled scripters who can anticipate user utterances, generate permutations of the utterances, reduce these permutations through generalisation to patterns (use of wild cards) and, very importantly, anticipate interactions between rules.

The alternative offered by STSS is to build the rules from a set of prototype or archetype STs. Suppose, instead of patterns; we had rules containing the following STs:

I can not afford to pay. (ps1)
My money has not come from the Student Loan Company. (ps2)

The user utterance

I cannot afford to pay you anything this term. (u)

would be compared with all of the prototype statements using the STSS algorithms and the highest similarity match would win, as expressed in equation 1:

$$\text{sim}(ps1,u) > \text{sim}(ps2,u) \quad (1)$$

The rule containing ps1 would win and the action specified for the rule would be taken (an attribute set, response to user generated etc.)

Early work on text similarity was concerned with relatively long documents. Consequently similarity was measured using exact matches between words in relatively long vectors of words selected from their respective documents based on their significance [80, 81] Utterances in dialogue are much shorter, and two utterances which convey largely the same meaning may share no common words at all.

5.1 The STASIS Algorithm

The STASIS algorithm [82] was specifically designed to overcome this problem. Its key features are:

- Short vectors derived only from the words in the STs
- Use of function words, specific word forms (no stemming/lemmatisation)
- Exploitation of word order information.

Function words are high-frequency closed-class words e.g. articles and auxiliary verbs. In the two sentences "Could you pass the salt?" and "Did you pass the salt?" a single word changes the speech act [83], the overt meaning and the subtle implications of the basic propositional content.

Following the larger-scale publication of STASIS [82], there has been a flurry of work in the STSS field. The majority of subsequent work in the field is either derivative from or influenced by STASIS [84-94]. Accordingly, the following description of STASIS should prove useful to Systems Engineers wishing to develop STSS algorithms.

STASIS uses two stages to calculate the overall semantic similarity between two Short Texts: construction of two vectors (semantic and word-order), followed by combination of the similarity information obtained by the vectors. This is shown in figure 4.

In the following, taken from [82], the lexical database, corpus and word similarity measure components can be replaced by alternatives, although the word similarity measure used in [95] is recommended.

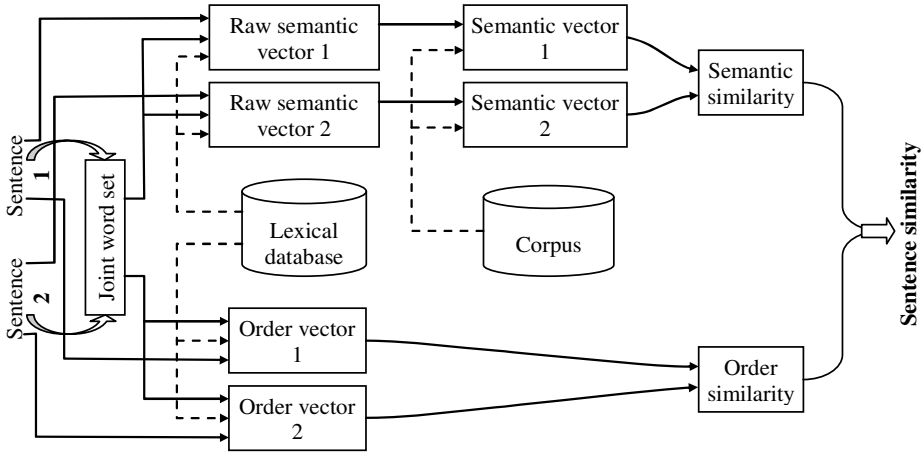


Fig. 4. STASIS sentence similarity computation diagram

In summary the semantic similarity is calculated as follows:

- A joint word set T is derived from all of the distinct words in two short texts, T1 and T2 (equation 2).

$$T = T_1 \cup T_2 = \{w_1 \quad w_2 \quad \cdots \quad w_m\} \tag{2}$$

- A lexical semantic vector *š* is derived from the joint word set for each short text. Each entry, *š*$_i$*(i=1,2,...,m)*, is determined by the semantic similarity of the corresponding word in the joint word set to a word in the short text (where *m* equals the number of words in the joint word set). Semantic similarity between non-identical words is calculated using the Wordnet ontology.

The words are weighted according to their information content [96] using equation 3:

$$s_i = \breve{s} \cdot I(w_i) \cdot I(\tilde{w}_i) \tag{3}$$

where $I(w_i)$ is the information content of a word in the joint word set and $I(\tilde{w}_i)$ is the information content of its associated word in the short text.

- The semantic similarity (S_s) between the two short texts is calculated using a cosine-like measure between the semantic vectors s_1 and s_2 using a cosine-like function (equation 4):

$$S_s = \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|} \tag{4}$$

Word order similarity is calculated as follows:

- Word order vectors, $\mathbf{r_1}$ and $\mathbf{r_2}$ are constructed using the index numbers of words from the joint word set to represent the words in each of the short texts.
- The word order similarity component (S_r) is calculated as the normalised difference in word order (equation 5):

$$S_r = 1 - \frac{\|\mathbf{r}_1 - \mathbf{r}_2\|}{\|\mathbf{r}_1 + \mathbf{r}_2\|} \tag{5}$$

Finally short text similarity is calculated using a weighted sum of the two components using equation 6:

$$S(T_1, T_2) = \delta \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|} + (1-\delta) \frac{\|\mathbf{r}_1 - \mathbf{r}_2\|}{\|\mathbf{r}_1 + \mathbf{r}_2\|} \tag{6}$$

The parameter δ (which adjusts the relative contributions of semantic and word order) is in the range $0.5 < \delta < 1$ and was chosen empirically. Some preliminary experiments have shown potential for significant improvement in performance by optimising δ

through linear regression or by combining the two components using a Neural Network [97]. Recent work on embedding the STASIS algorithm in a CA has shown considerable promise [92].

5.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) offers an alternative approach to STSS measurement [98]. Although it was originally designed for document retrieval, the LSA website has a front-end for calculating the similarity between pairs of sentences [99]. There are a number of implementations available on the web. A full description of LSA can be found in [98].

The first stage of LSA is the construction of a semantic space as follows:

- The collection of documents comprising the space is pre-processed to remove all but the most useful terms
- A term-by-document matrix is constructed (X)
- This large matrix is decomposed by Singular Value Decomposition into 3 other matrices, one derived from the terms (T_0), one from the documents (D_0) and a diagonal matrix (S_0) linking them
- The reduction in the size of S_0, which reduces the size of T_0 and D_0
- An approximation of the original space, $\hat{X}$, is reconstructed according to equation 7:

$$X \approx \hat{X} = TSD' \tag{7}$$

LSA generalises by spreading information from a cell to its related cells; whereas X contains a small number of 1s and a large number of 0s, $\hat{X}$ will contain many more non-zero values. Thus when a text is projected into the semantic space, more cells (and therefore spatial dimensions) contribute to its representation, enabling LSA to calculate a similarity even when no co-occurrence exists between particular terms.

Short text similarity is calculated as follows:

- A row in the semantic space is formed for each ST
- If a text did not appear in the original space it is constructed as a pseudo object using equation 8:

$$D_q = X_q' TS^{-1} \tag{8}$$

The method for this is disclosed in [100].

- Similarity is measured as the angle between the vectors representing the two short texts using equation 9:

$$\hat{X}\hat{X}' = TS^2T' \tag{9}$$

The performance of LSA depends on the semantic space. It is possible to use the best performing semantic space from LSA's creators [98], in which case only the steps for similarity calculation are required.

6 Research Opportunities

There are opportunities for systems engineers to perform research on virtually every aspect of CAs. Starting with SDS, the key problem is the vocabulary. The ultimate research aim must be a universal SDS interface that allows CAs to be deployed over the internet for the general population. However, the state of the art is a high WER with large vocabulary systems or a small vocabulary with more reliable systems [14, 16]. Potential lines of research are:

- The production of multi-classifier systems adding more diverse elements such as neural networks to the current statistically-based techniques
- Development of generic (rather than customised) medium-sized vocabularies that would be re-usable across different projects in the same general domain
- Development of an open SDS protocol that locates the speech recognition in the user's mobile phone (such a system would be trained only once and would act as the SDS front end for any application the phone owner wished to use).

Chatterbot systems still offer a useful starting point for systems engineers to become involved in CA research or development. In particular, the ALICE AIML chatterbot technology is readily available and well documented. Furthermore, ALICE also illustrates that there is potential for introducing new creative techniques to improve chatterbots: the symbolic reduction (SRAI) instruction allows an AIML chatterbot to re-enter itself recursively to decompose complex user utterances. Potential lines of chatterbot research are:

- Development of software tools to improve productivity of scripting
- Development of tools to support debugging and maintenance
- The extension of chatterbot engine functionality.

The chief problem of Natural Language Processing is that CA development has a minor role in the field as a whole, with its main interests being in areas such as information retrieval and machine translation. Undoubtedly many interesting and novel algorithms, architectures and processes have been developed, but in a piecemeal fashion. Therefore the research challenges for NLP systems are:

- The construction and evaluation of real-world systems that can be deployed for use by the general public
- Measurement of the scalability of such systems to realistic numbers of users in real-time.

GO-CAs are a recent development and opportunities for research are plentiful. Some lines include:

- Development of alternative top-level component architectures (for example through the use of alternative intelligent components such as neural networks to drive the goals)
- Development of alternative methods for the DA to communicate with the other components (at present the DA is a pattern matching engine)
- Application of new algorithms such as STSS within the DA
- Development of new knowledge engineering tools and processes
- Development of authoring and maintenance tools

ECAs present the same kinds of opportunities for systems engineers as NLP systems. ECA research interest has shifted from fairly mundane but obviously useful topics such as disambiguation through pointing at objects and co-operative use of objects such as maps in shared visual space, to more abstract topics like measuring engagement in multi-party dialogues and simulation of emotions. Again the immediate research topics of interest to a systems engineer should be:

- Development of ECAs using current properties that are believed to be useful (e.g. pointing, emphatic gestures etc.)
- Applying such ECAs to realistic problems requiring larger numbers of complex attributes
- Objective evaluation of the gain (in terms of metrics such as successful task completion, length of dialogue etc.) obtained from ECA features.

7 Conclusions

CA technology is something that all systems engineers should be aware of. GO-CA technology has reached the point where it is possible to build and deploy real-world applications and some systems engineers in industry will find that this forms part of projects they will work on during their careers. How far CAs penetrate into mainstream computing will depend on research in the short and medium term. This research, particularly involving development of authoring and maintenance tools, and the objective evaluation of tools, algorithms and techniques, will be crucially dependent on the skills of the systems engineer to be successful.

Glossary

ALICE:	Artificial Linguistic Internet Computer Entity
CA:	Conversational Agent
DA:	Dialogue Agent
DARPA:	Defense Advanced Research Projects Agency
DM:	Dialogue Manager
ECA:	Embodied Conversational Agent
GO-CA:	Goal Oriented Conversational Agent
LSA:	Latent Semantic Analysis
NLP:	Natural Language Processing
NLU:	Natural Language Understanding
PARADISE:	a framework for evaluating and comparing the performance of spoken-language dialogue systems
REA:	a Real Estate Agent
SCHISMA:	SCHouwburg Informatie Systeem (a Dutch theatre booking system)
SDS:	Spoken Dialogue System
SMS:	Short Message Service (texting)
SRAI:	Symbolic Reduction Artificial Intelligence
STASIS:	a specific instance of an STSS algorithm
STSS:	Short Text Semantic Similarity
VSA:	Virtual Seller Agent
WER:	Word Error Rate

References

1. Turing, A.M.: Computing Machinery and Intelligence. Mind, New Series 59(236), 433–460 (1950)
2. Gunderson, K.: The Imitation Game. Mind, New Series 73(290), 234–245 (1964)
3. Searle, J.R.: Minds, brains and programs. Behavioural and Brain Sciences 3, 417–424 (1980)
4. Block, N.: Psychologism and behaviourism. The Philosophical Review LXXXX(1), 5–43 (1981)
5. Searle, J.R.: Mind, Language and Society. Weidenfield & Nicholson (1999)
6. Bickmore, T., Giorgino, T.: Health dialog systems for patients and consumers. J. Biomed. Inform. 39(5), 556–571 (2006)
7. Cassell, J., et al.: Embodied CAs (2000)
8. Gorin, A.L., Riccardi, G., Wright, J.H.: How may I help you? Speech Communication 23, 113–127 (1997)
9. Graesser, A.C., et al.: AutoTutor: An Intelligent Tutoring System With Mixed Initiative Dialogue. IEEE Transactions on Education 48(4), 612–618 (2005)
10. Owda, M., Bandar, Z., Crockett, K.: Conversation-Based Natural Language Interface to Relational Databases. In: IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops (2007)
11. Glass, J., et al.: A Framework for Developing Conversational User Interfaces. In: Fourth International Conference on Computer-Aided Design of User Interfaces, Funchal, Isle of Madeira, Portugal, pp. 347–358 (2004)
12. McGeary, Z., et al.: Online Self-service: The Slow Road to Search Effectiveness. In: Customer Relationship Management (2005)
13. Hunt, M.J.: Figures of Merit for Assessing Connected Word Recognisers. Speech Communication 9, 239–336 (1990)
14. Hosom, J.-P.: Automatic Speech Recognition at CSLU (2003), `http://cslu.cse.ogi.edu/asr/` (cited October 20, 2010)
15. Hunt, A.: comp.speech FAQ Section 6 (1997), `http://www.speech.cs.cmu.edu/comp.speech/Section6/Q6.1.html` (cited October 20, 2010)
16. Raut, C.K.: Discriminative Adaptive Training and Bayesian Inference for Speech Recognition. In: Emmanuel College, University of Cambridge (2009)
17. Hillard, D.L.: Automatic Sentence Structure Annotation for Spoken Language Processing. In: Electrical Engineering, University of Washington (2008)
18. Zdenek, S.: Passing Loebner's Turing test: A case of Conflicting Discourse Functions. Minds and Machines 11, 53–76 (2001)
19. Jackson, P., Moulinier, I.: Natural Language Processing for Online Applications, 2nd edn. Natural Language Processing. John Benjamins Publishing Company, Amsterdam (2007)
20. Zdravkova, K.: Conceptual Framework for an Intelligent ChatterBot. In: 22nd International Conference Information Technology Interfaces ITI 2000, pp. 189–194 (2000)
21. Minker, W., Bennacef, S., Gauvain, J.-L.: A stochastic case frame approach for natural language understanding. In: Fourth International Conference on Spoken Language, ICSLP 1996, Philadelphia, PA, pp. 1013–1016 (1996)
22. Farquhar, A., Fikes, R., Rice, J.: The Ontolingua Server: a Tool for Collaborative Ontology Construction. Journal of Human-Computer Studies 46, 707–728 (1997)

23. Sagae, K., et al.: Towards Natural Language Understanding of Partial Speech Recognition Results in Dialogue Systems. In: NAACL HLT 2009, pp. 53–56. Association for Computational Linguistics, Boulder (2009)
24. Young, S., et al.: The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. Computer Speech and Language, Special Issue on Evaluation 24(2), 150–174 (2010)
25. Lefevre, F., et al.: k-Nearest Neighbor Monte-Carlo Control Algorithm for POMDP-Based Dialogue Systems. In: The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue, London, UK, pp. 272–275 (2009)
26. Bohus, D., Horvitz, E.: Learning to Predict Engagement with a Spoken Dialog System in Open-World Settings. In: The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue, London, UK (2009)
27. Bohus, D., Horvitz, E.: Models for Multiparty Engagement in Open-World Dialog. In: SIGDIAL 2009: the 10th Annual Meeting of the Special Interest Group in Discourse and Dialogue, Queen Mary University of London, pp. 225–234 (2009)
28. DeVault, D., Sagae, K., Traum, D.: Can I finish? Learning when to respond to incremental interpretation results in interactive dialogue. In: The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue, London, UK, pp. 11–20 (2009)
29. Skantze, G., Gustafson, J.: Attention and Interaction Control in a Human-Human-Computer Dialogue Setting. In: The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue, London, UK, pp. 310–313 (2009)
30. op den Akker, H., op den Akker, R.: Are You Being Addressed? - real-time addressee detection to support remote participants in hybrid meetings. In: The SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue, London, UK, pp. 21–28 (2009)
31. Crockett, K., et al.: Bullying and Debt: Developing Novel Applications of Dialogue Systems. In: Knowledge and Reasoning in Practical Dialogue Systems (IJCAI), pp. 1–9. IJCAI, Pasadena (2009)
32. Cassell, J., et al.: More Than Just a Pretty Face: Conversational Protocols and the Affordances of Embodiment. Knowledge-Based Systems 14, 55–64 (2001)
33. Robinson, S., et al.: What would you ask a CA? Observations of Human-Agent Dialogues in a Museum Setting. In: Language Resources and Evaluation Conference 2008, Marrakech, Morocco, pp. 1125–1131 (2008)
34. Babu, S., et al.: "What Would You Like to Talk About?" An Evaluation of Social Conversations with a Virtual Receptionist. In: Gratch, J., Young, M., Aylett, R.S., Ballin, D., Olivier, P. (eds.) IVA 2006. LNCS (LNAI), vol. 4133, pp. 169–180. Springer, Heidelberg (2006)
35. Lance, B.J., Marsella, S.C.: A model of gaze for the purpose of emotional expression in virtual embodied agents. In: International Conference on Autonomous Agents, Estoril, Portugal, pp. 199–206 (2008)
36. Gergle, D., Rosé, C.P., Kraut, R.E.: Modeling the Impact of Shared Visual Information on Collaborative Reference. In: The SIGCHI Conference on Human Factors in Computing Systems, pp. 1543–1552 (2007)
37. Bickmore, T., Cassell, J.: 'How about this weather?' Social Dialog with Embodied CAs. In: The American Association for Artificial Intelligence (AAAI) Fall Symposium on "Narrative Intelligence", Cape Cod, MA, pp. 4–8 (2000)

38. Kopp, S., et al.: A Conversational Agent as Museum Guide – Design and Evaluation of a Real-World Application. In: Panayiotopoulos, T., Gratch, J., Aylett, R.S., Ballin, D., Olivier, P., Rist, T. (eds.) IVA 2005. LNCS (LNAI), vol. 3661, pp. 329–343. Springer, Heidelberg (2005)
39. Bevacqua, E., et al.: An expressive ECA showing complex emotions. In: AISB 2007 - Artificial and Ambient Intelligence, Newcastle University, Newcastle upon Tyne, UK (2007)
40. Walker, M.A., Hirschman, L., Aberdeen, J.: Evaluation for Darpa Communicator Spoken Dialogue Systems. In: Language Resources and Evaluation Conference, Athens, Greece (2000)
41. Walker, M.A., Passonneau, R., Boland, J.E.: Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems. In: The 39th Annual Meeting on Association for Computational Linguistics, Toulouse, France, pp. 515–522 (2001)
42. Walker, M.A., et al.: Speech-Plans: Generating Evaluative Responses in Spoken Dialogue. In: International Conference on Natural Language Generation, pp. 73–80 (2002)
43. Giraudo, E., Baggia, P.: EVALITA 2009: Loquendo Spoken Dialog System. In: Evaluation of NLP and Speech Tools for Italian EVALITA 2009, Reggio Emilia, Italy (2009)
44. Rigo, S., et al.: The 2009 UNITN EVALITA Italian Spoken Dialogue System. In: Evaluation of NLP and Speech Tools for Italian EVALITA 2009, Reggio Emilia, Italy (2009)
45. De Angeli, A.: Ethical implications of verbal disinhibition with CAs. PsychNology Journal 7(1), 49–57 (2009)
46. Kegel. Kegel - Oldest Harley Dealer, `http://kegelmotorcycles.com/` (cited March 23, 2010)
47. Hulstijn, H., et al.: Topics in schisma dialogues. In: Twente Workshop on Language Technology 11 (TWLT11), University of Twente (1996)
48. Andernach, T., et al.: Language Analysis for Dialogue Management in a Theatre Information & Booking System. In: 15th International Conference on Language Engineering, AI 1995, Montpellier, pp. 351–362 (1995)
49. Michie, D.: Return of the Imitation Game. Electronic Transactions in Artificial Intelligence 6(B), 205–220 (2001)
50. BBC. Tories criticise flu advice line 2009 06:45 GMT, Friday, July 24 (2009), 07:45 UK, `http://news.bbc.co.uk/1/hi/health/8166444.stm` (cited June 24, 2009)
51. Morge, M., Abdel-Naby, S., Beaufils, B.: Towards a dialectical approach for CAs in selling situations. In: The 9th International Conference on Autonomous Agents and Multiagent Systems, Toronto, Canada, pp. 127–144 (2010)
52. Lance, B., Marsella, S.: A Model of Gaze for the Purpose of Emotional Expression in Virtual Embodied Agents. In: 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, pp. 199–206 (2008)
53. Plantec, P.: The Zen of Scripting Verbots (1998), `http://web.archive.org/web/19991013075513/vperson.com/verbotzen30tt.html` (cited September 28, 2004)
54. Sammut, C.: Managing Context in a CA. Electronic Transactions in Artificial Intelligence 5(B), 191–201 (2001)
55. Dehaene, S., Naccache, L.: Towards a cognitive neuroscience of consciousness: basic evidence and a workspace framework. Cognition 79, 1–37 (2001)

56. Forbes-Riley, K., Litman, D.: Designing and evaluating a wizarded uncertainty-adaptive spoken dialogue tutoring system. Computer Speech and Language, Special Issue on Evaluation 25(1), 105–126 (2011)
57. Walker, M.A., et al.: PARADISE: a framework for evaluating spoken dialogue agents. In: The 35th Annual Meeting of the Association for Computational Linguistics, Madrid, Spain, pp. 271–280 (1997)
58. Walker, M.A., et al.: Darpa communicator dialog travel planning systems: The June 2000 data collection. In: EUROSPEECH 2001 7th European Conference on Speech Communication and Technology 2nd INTERSPEECH Event, Aalborg, Denmark, pp. 1371–1374 (2001)
59. Fenton, N., Pfleeger, S.: Software Metrics: A Rigorous and Practical Approach. PWS (1998)
60. Kato, T., Matsushita, M., Kando, N.: Bridging Evaluations: Inspiration from Dialogue System Research. In: SIGIR 2010 33rd Annual International ACM SIGIR Conference, SIGIR, pp. 3–4 (2010)
61. Dethlefs, N., et al.: Evaluating Task Success in a Dialogue System for Indoor Navigation. In: SemDial 2010 14th Workshop on the Semantics and Pragmatics of Dialogue, pp. 143–146 (2010)
62. Lee, C., et al.: Recent Approaches to Dialog Management for Spoken Dialog Systems. Journal of Computing Science and Engineering 4(1), 1–22 (2010)
63. Foster, M.E., et al.: Evaluating Description and Reference Strategies in a Cooperative Human-Robot Dialogue System. In: The 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, pp. 1818–1823 (2009)
64. Webb, N., et al.: Evaluating Human-Machine Conversation for Appropriateness. In: The 7th Conference on International Language Resources and Evaluation (LREC 2010), Valletta, Malta (2010)
65. Bouwman, G., Sturm, J., Boves, L.: Incorporating confidence measures in the Dutch train timetable information system developed in the ARISE project. In: ICASSP 1999, pp. 493–496 (1999)
66. Semeraro, G., et al.: Evaluation and Validation of a Conversational Agent Embodied in a Bookstore. In: Carbonell, N., Stephanidis, C. (eds.) UI4ALL 2002. LNCS, vol. 2615, pp. 360–371. Springer, Heidelberg (2003)
67. Andersen, V., et al.: A methodological approach for designing and evaluating intelligent applications for digital collections. Applied Artificial Intelligence 17(8-9), 745–771 (2003)
68. Lamel, L., et al.: User evaluation of the MASK kiosk. Speech Communication 38(1), 131–139 (2002)
69. Ortony, A., Clore, G.L., Collins, A.: The Cognitive Structure of Emotions. Cambridge University Press, Cambridge (1990)
70. Yuan, X., Chee, Y.S.: Design and evaluation of Elva: an embodied tour guide in an interactive virtual art gallery. Computer Animation and Virtual Worlds 16(2), 109–119 (2005)
71. McKevitt, P., Partridge, D., Wilks, Y.: Why machines should analyse intention in natural language dialogue. Int. J. Human-Computer Studies 51, 947–989 (1999)
72. Le Bigot, L., Jamet, E., Rouet, J.-F.: Searching information with a natural language dialogue system: a comparison of spoken vs. written modalities. Applied Ergonomics 35, 557–564 (2004)
73. Cassell, J., Vilhjálmsson, H.: Fully Embodied Conversational Avatars: Making Communicative Behaviors Autonomous. Autonomous Agents and Multi-Agent Systems 2(1), 45–64 (1999)

74. Massaro, D.W., et al.: Developing and evaluating CAs. In: Cassell, J., et al. (eds.) Embodied CAs, pp. 286–318. MIT Press, Cambridge, MA (2000)
75. Cassell, J., Bickmore, T.: Negotiated Collusion: Modeling Social language and its Relationship Effects in Social Agents. User Modeling and User-Adapted Interaction 13, 89–132 (2003)
76. Lamel, L., et al.: The LIMSI RailTel System: Field trial of a telephone service for rail travel information. Speech Communication 23(1-2), 67–82 (1997)
77. Litman, D.J., Pan, S.: Designing and Evaluating an Adaptive Spoken Dialogue System. In: User Modeling and User-Adapted Interaction, vol. 12, pp. 111–137 (2002)
78. Sanders, G.A., Scholtz, J.: Measurement and Evaluation of Embodied CAs. In: Cassell, J., et al. (eds.) Embodied CAs. MIT Press, Cambridge, MA (2000)
79. Bouwman, G., Hulstijn, J.: Dialog Strategy Redesign with Reliabilty Measures. In: 1st Int. Conf. on Language Resources and Evaluation, Granada, Spain, pp. 191–198 (1998)
80. Spärck-Jones, K.: A Statistical Interpretation of Term Specificity and its Application in Retrieval. Journal of Documentation 28, 11–21 (1972)
81. Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing. Communications of the ACM 18(11), 613–620 (1975)
82. Li, Y., et al.: Sentence Similarity Based on Semantic Nets and Corpus Statistics. IEEE Transactions on Knowledge and Data Engineering 18(8), 1138–1150 (2006)
83. Austin, J.L.: In: Urmson, J.O. (ed.) How to do things with Words: The William James Lectures delivered at Harvard University in 1955, 2nd edn. Harvard University Press, Cambridge, MA (1975)
84. Ferri, F., Grifoni, P., Paolozzi, S.: An Approach to Multimodal Input Interpretation in Human-Computer Interaction. In: The Nineteenth International Conference on Software Engineering Knowledge Engineering (SEKE 2007), Boston, MA, USA, pp. 664–669 (2007)
85. Tsatsaronis, G., Varlamis, I., Vazirgiannis, M.: Text Relatedness Based on a Word Thesaurus. Journal of Artificial Intelligence Research 37, 1–39 (2010)
86. Min, F., Wenyin, L., Chen, W.: Answer Clustering and Fusion in a User-Interactive QA System. In: Second International Conference on Semantics, Knowledge and Grid, p. 41 (2006)
87. Gacitua-Decar, V., Pahl, C.: Automatic Business Process Pattern Matching for Enterprise Services Design. In: 2009 World Conference on Services - II, Bangalore, India, pp. 111–118 (2009)
88. Huang, J.-J., Changt, S.-T., Hu, S.-Y.: Searching for Answers via Social Networks. In: 5th IEEE Consumer Communications and Networking Conference, CCNC 2008, Las Vegas, NV, pp. 289–293 (2008)
89. Capuano, N., et al.: On-Demand Construction of Personalized Learning Experiences Using Semantic Web and Web 2.0 Techniques. In: Ninth IEEE International Conference on Advanced Learning Technologies, pp. 484–488. IEEE Computer Society, Washington, DC (2009)
90. Inkpen, D.: Semantic Similarity Knowledge and its Applications. Studia Universitatis Babes-Bolyai Informatica LII(1), 11–22 (2007)
91. Achananuparp, P., Hu, X., Shen, X.: The Evaluation of Sentence Similarity Measures. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2008. LNCS, vol. 5182, pp. 305–316. Springer, Heidelberg (2008)
92. O'Shea, K., Bandar, Z., Crockett, K.: Towards a New Generation of CAs Based on Sentence Similarity. Lecture Notes Electrical Engineering, vol. 39, pp. 505–514 (2009)

93. O'Shea, K., Bandar, Z., Crockett, K.: A Novel Approach for Constructing CAs using Sentence Similarity Measures. In: Proceedings of the World Congress on Engineering WCE 2008, London, U.K., pp. 321–326 (2008)
94. Liu, X., Zhou, Y., Zheng, R.: Sentence Similarity based on Dynamic Time Warping. In: International Conference on Semantic Computing, ICSC 2007, pp. 250–256 (2007)
95. Li, Y., Bandar, Z., McLean, D.: An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. IEEE Transactions on Knowledge and Data Engineering 15(4), 871–882 (2003)
96. Resnik, P.: Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. Journal of Artificial Intelligence Research 11, 95–130 (1999)
97. O'Shea, J.: A Framework for Applying Short Text Semantic Similarity in Goal-Oriented CAs. In: Computing and Mathematics. Manchester Metropolitan University, Manchester (2010)
98. Landauer, T.K., Foltz, P.W., Laham, D.: An Introduction to Latent Semantic Analysis. Discourse Processes 25, 259–284 (1998)
99. Laham, D.: Latent Semantic Analysis @ CU Boulder (1998), LSA website, `http://lsa.colorado.edu/` (cited January 20, 2008)
100. Deerwester, S., et al.: Computer information retrieval using Latent Semantic Structure, U.S.P. Office. Bell Communications Research Inc.: United States of America (1989)

Chapter 9
Advanced Concepts and Generative Simulation Formalisms for Creative Discovery Systems Engineering

Levent Yilmaz[1] and C. Anthony Hunt[2]

[1] Auburn University
yilmaz@auburn.edu
[2] University of California at San Francisco
c.hunt@ucsf.edu

Abstract. While M&S has been widely used in engineering and computational sciences to facilitate empirical insight, optimization, prediction, and experimentation, the role of simulation in supporting early foresight phases of creative problem solving received less attention. We advocate models of creative cognition to rethink simulation modeling so that creativity is enhanced rather than stifled. Generative Parallax Simulation (GPS) is introduced as a strategy and a generic and abstract specification for its realization is presented. GPS is based on an evolving ecology of ensembles of models that aim to cope with ambiguity, which pervades in early phases of model-based science and engineering. Besides its contributions as a modeling and simulation methodology in support of creativity, GPS provides a fertile and useful domain as an application testbed for parallel simulation.

1 Introduction

Advances in multimodel formalisms (Zeigler and Oren, 1986; Fishwick and Zeigler, 1992) and exploratory modeling techniques (Davis and Bigelow, 2000) resulted in significant improvement in dealing with uncertainty and enhancing computational productivity in complex system analysis. However, as the use of computational modeling and simulation become pervasive, scientists are moving from the comparably well-understood and safe territory of computational productivity to the more ambiguous domain of discovery, creativity, and innovation (Schneiderman, 2007).

To demonstrate the utility of viewing scientific knowledge generation as an evolving complex adaptive ecology of analogue ensembles, we introduce a strategy to leverage principles of complex systems thinking to foster creative discovery through strategic and context-sensitive creation and evaluation of many mechanistic analogue (model) schemas. Those easily falsified are discarded. Those that survive a round of falsification provide features that can be copied and assembled differently to make new schemas, and a few of the mechanisms represented may be successful in initially non-intuitive ways. We expect those will lead to

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 233–258.
springerlink.com

new insight; some may catalyze creative leaps that otherwise may be slow in coming when only the domain of expensive wet-lab experimentation is available. Mechanism schemas that survive several rounds of falsification will stand as an ensemble of concretized theories of how a morphogenic emerges. The approach has the potential to become a new means of stimulating creativity while advancing biomedical science and engineering.

For the purpose of this article, creativity is interpreted as the production of novel and useful ideas (Amabile, 1996). More specifically, creative discovery is viewed as the product of a process that involves concept (e.g., model) combination, expansion, metaphor, and analogy along with mechanisms acting in concert to expand the frontiers of knowledge and conceptualization in a domain. Hence, creativity can be seen as a system by which processes transform and create structures to produce results that are novel, yet rooted in existing knowledge (Ward et al., 1997).

Generative Parallax Simulation (GPS) views novelty as an emergent phenomenon; it harnesses the principles of self-organization and draws from the science of complexity to enable simulation technology development to enhance discovery in model-based science and engineering. The proposed approach is influenced by and extends on the promise and success of our recent work on Symbiotic Adaptive Multisimulation (Mitchell and Yilmaz, 2008) that views model behavior generation as a complex adaptive system. By leveraging models of creative cognition (Sawyer, 2008), we delineate simulation modeling concepts to improve computational discovery.

Creative processes often involve a broad idea generation phase that is approached from different perspectives, followed by idea evaluation and selection (Amabile, 1996) . Because creativity requires novel yet useful solutions to make creative leaps, appropriate trade offs between constraints and flexibility are needed over the models' representation of the problem. Hence, the effectiveness of simulation systems that support creative discovery will rely heavily on their ability to start behaving robustly across a large number of hypotheses, constraints, and propositions, followed by narrowing toward a limited range of conditions that are found to be plausible in terms of explaining extensible set of attributes. Development of such simulation systems will require progress in addressing the following:

- when does discovery involve exploitation of a problem space, and when does it involve exploring alternative problem representations?
- what are appropriate trade offs between constraints and flexibility in supporting incremental and iterative expansion of the model space to explain expanding sets of mechanisms and attributes underlying the phenomena of interest?
- how can models and theories of creative cognition help us rethink simulation modeling so that creativity is enhanced rather than stifled?

The concepts introduced herein coupled with preliminary explorations are moving us toward answers. In Figure 1, they are shown as the domain of generative

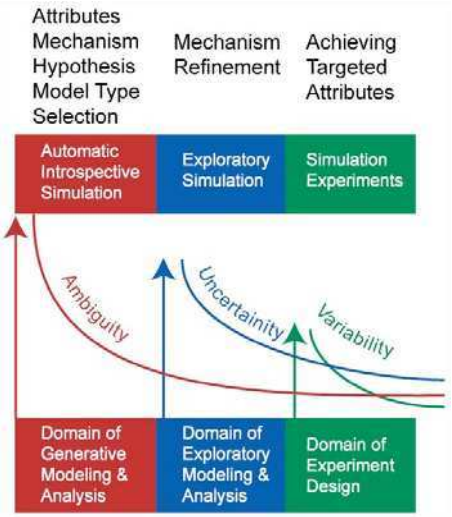


Fig. 1. Generative and Exploratory Simulation

modeling and analysis, which precedes exploratory analysis and computational experimentation. To demonstrate the proposed concepts, we use model-based scientific study of biological systems as a motivating scenario. For clarity, we restrict discussions to problem solving in the biological domain, where uncertainty is ample. Further, our initial focus is on problems requiring deeper insight into causal mechanisms responsible for such biological phenomena as epithelial morphogenesis. Nevertheless, the ideas presented are generalizable. A major thrust of GPS is that by automating (large parts of) generating and exploring simulation models, we can foster creativity and dramatically enhance the generation of foresight into epithelial morphogenesis, our motivating scenario. Achieving measurable scientific progress will motivate extension of the approach into other domains. Following the success of earlier work on Symbiotic Adaptive Multisimulation (Mitchell and Yilmaz, 2008), we leverage principles of complex systems thinking in simulator design to propose GPS as a method based on adaptive evolutionary ecologies of model ensembles.

The rest of the paper is organized as follows. In section 2, a motivating scenario is presented to emphasize the significance of generative modeling approach for improving creativity problem solving. To set the context for desirable characteristics for next generation intelligent simulators and to provide a basis for computational support for creative processes, section 3 overviews confluence models of creativity. Building on these cognitive and social models of creativity, section 3 outlines design principles for creativity support simulation systems by introducing a generic formalism for GPS. Section 4 focuses on meta-simulation study of GPS to better understand its dynamics. In section 5, we discuss limitations of GPS and suggest mechanisms to improve its ability to generate novel and useful models as a creative evolutionary system and recommend potential avenues of

future research. We conclude in section 6 by emphasizing the value of GPS in addressing uncertainty and ambiguity during early phases of model-based science and engineering.

1.1 Motivation

We use the metaphor of traditional cycles of scientific advancement, where hypotheses are formulated from a body of knowledge and their testable consequences are evaluated. To sharpen the focus, the specific system on which we use to motivate proposed concepts is epithelial cells undergoing morphogenesis or repair (Kim et al., 2008, 2009). The analogue (model) systems described are testable hypotheses about mechanisms that may be responsible for specified attributes of epithelial cell cultures. The problem is straightforward yet complex: discover plausible mechanisms, consistent with current, expanding knowledge, for sets of perspective-dependent behaviors. There are no "correct" mechanisms. In many cases there may be multiple, equally valid mechanisms. The approach advanced herein uses computational modeling and simulation in new ways. The proposed work is predicated on the conjecture that theories of creativity and creative cognition, in conjunction with principles of complex adaptive systems thinking facilitate reconsidering the use of simulation so that creative discovery in model-based exploration of complex systems can be enhanced rather than stifled.

The space of behaviors and mechanisms of living systems, even cultured cells in vitro, is much larger than the space of attributes that can be measured using today's technologies (Engelberg et al., 2008; Lam and Hunt, 2008). The scientific method requires focusing on a finite number of observations during each experiment and clamp down onconstrainother system behaviors to limit the influence of unmeasured attributes. Triangulating on the real system by building perspective dependent models of the type described here (i.e., evolving ecology of multiple ensembles of models) will help provide deeper insight. Parallax Simulation is based on that triangulation concept. However, the inherent sequentiality of iterative, one-at-a-time model development limits research progress by limiting the way the generators can be assembled to fill the simulated attribute space. Completely exploring the potential attribute space becomes infeasible with even a relatively small number of operating principles (i.e. abstract mechanisms).

1.2 Scientific Problem Solving with Computational Models

Figure 2 depicts the space of models used in biological research. Like much of biology, epithelial morphogenesis, is complex and a great deal remains obscure. Even when studying a single epithelial cell-line, such as Madin-Darby canine kidney (MDCK) cells, use of two different experimental conditions can change the mix of phenotypic attributes observed. It is as if we are observing the biological system from two different perspectives.

To demonstrate that we understand how molecular level details within mammalian cells interconnect at different levels and emerge as a dynamic,

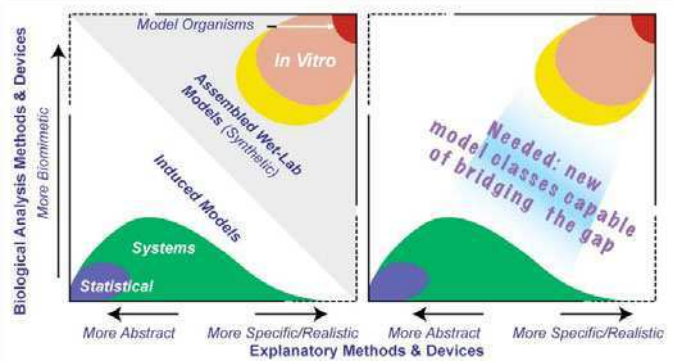


Fig. 2. Bridging the Gap between Model Organisms and Computational Models: Model types are arranged according to abstraction level versus biological character. The Biomimetic axis indicates the degree to which a model resembles its referent. Statistical models are to the lower left (blue). Model organisms are to the upper right (red).

multi-attribute systemic phenotype, we need models and methods that can bridge the gap between wet-lab systems, such as MDCK cell cultures, and the current generation of inductive, mathematical models, such as those used in pharmacokinetics (PK) and pharmacodynamics (PD). We need computer-based models that, although still abstractions, have extant mechanisms that generate emergent properties analogous to how biological counterpart properties emerge during experiments. The current generation of biological systems models (green) are inductive, partially heuristic, partially predictive, biomimetic, functional and structural models used for evaluating canalizing aspects of data obtained from a referent systems located on the other side of the gap. On the other hand, models that can bridge the gap will be heuristic, biomimetic analogues (models) that can be used to evaluate explicit mechanistic hypotheses from different experimental perspectives in the context of many aspects of the referent. The in vitro models such as MDCK cell cultures are fundamentally different from their mammalian referents.

Figure 3 demonstrates phenotype overlap of different model systems. Each circle represents a set of relevant, measurable phenotypic attributes, viewed from related perspectives: epithelial cells within an animal model, MDCK cell cultures, and two somewhat different validated models of MDCK cells (Grant et al., 2007). The white spots within the MDCK phenotype illustrate specific, measurable attributes. The area of overlap between in vivo and MDCK cultures represents situations in which attributes and generative mechanisms are similar. The areas of overlap between the phenotypes of two in silico models and cultured MDCK cells: measurable properties of the analogue during execution are similar to corresponding measures of MDCK cultures. Each model is of a different perspective of the same system. In part B, after multiple rounds of revision and validation, the models in A have evolved. The larger area of overlap (interpenetration)

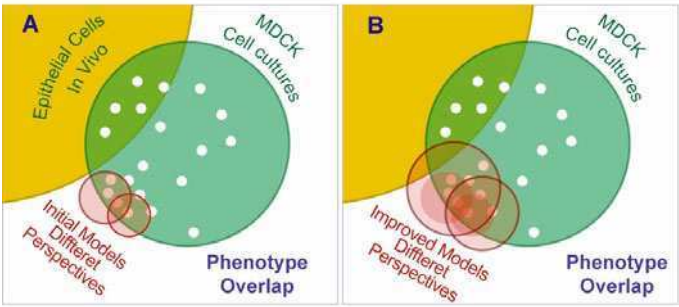


Fig. 3. Phenotype Overlap of Different Model Systems

means that a larger set of each models relevant phenotypic attributes (and generative mechanisms / operating principles) have been judged similar to a target set of MDCK counterparts. It may take multiple, somewhat different analogues to obtain broad coverage of in vitro attributes.

The task of discovering abstract, atomic, cell-level operating principles for in vitro morphogenesis requires multiple models and, perhaps, multiple experimental contexts in which those multiple models live. Initial success has been achieved (Grant et al., 2007; Kim et al., 2008) through the serial development of models, adding and removing various operating principles, executing the model, and falsifying against validation data from the referent. The fundamental method being employed is: a) hypothesize and implement operating principles, which can stand in for biological mechanisms, b) execute those implementations, c) determine a fine-grained falsification for the systemic behavior of the implementation, and d) modify, add, or remove operating principles and iterate. This method is a basic exploration of the forward and inverse maps from generator to phenomena and vice versa. This genotype/phenotype mapping is the most basic multi-scale nature of the research. It is based on falsification of the sequentially, iteratively developed models, not simply positive validation. The model validates only when it can no longer be falsified.

GPS is introduced as an effective means of identifying which analogue component or feature to change to extend the coverage of targeted attributes. That change may require addition of a new component, increasing spatial or temporal granularity, giving new capabilities to components, replacing a basic component with composites, changing experimental conditions etc. Once that has been achieved, it is important to explore and measure the changes that occur in an analogue's behavior within the parameter space close to that of the current parameter vector. Again, automated methods are needed. So doing is more than a form of sensitivity analysis. Some behaviors may suggest specific wet-lab experiments. Others may show a shift to abiotic behaviors (in Figure 2, moving away from a region of overlap). It can be useful to understand what analogue parameter changes move behaviors from a biomimetic region to an abiotic region. Therefore, a combination of divergent and convergent mechanisms in hypotheses generation and evaluation is critical to facilitate creative discoveries that are

meaningful and useful to scientists to better understand the phenomena of interest. Achieving a set of targeted attributes is a procedure for systematically reducing ambiguity while providing a degree of validation. Once that has been achieved, it is important to explore and measure the changes that occur in an analogue's behavior within the parameter space close to that of the current parameter vector. Again, automated methods are needed. So doing is more than a form of sensitivity analysis.

Such methods allow us to approach a biologically inspired form of modeling derived from synergistic integration of evolutionary dynamics and ecological perspective, which we call generative modeling. The ability to instantiate, execute, and if necessary, evolve multiple models, in parallel, all of which take similar but slightly different perspectives on the same referent (biological) system, opens the door to the automatic generation and selection (by falsification) of many somewhat different hypothetical, including non-intuitive mechanisms for that referent. In other words, the above ability would allow us to construct, execute, and falsify, many more hypotheses for the way a biological system works than can be achieved feasibly through our current sequential, iterative, modeling methods. Modeling throughput would increase exponentially. Such an exponential increase in model and hypothesis throughput would promote creative discovery and increase opportunities for creative leaps; that increase is necessary for generative simulation to begin significantly supplanting some of the current trial and error methods in domains like development of new therapeutics.

2 Models and Principles of Creative Problem Solving

There has been extensive research on creativity, discovery, and innovation at different levels spanning from cognition in individuals to groups, organizations, and collective creativity in large-scale community forms of organization. While significant process is achieved in many disciplines, the topic is relatively new for the modeling and simulation discipline.

2.1 Background

To improve creativity and discovery in model-based science and engineering, advanced simulation technologies can be extended to provide facilities and opportunities that go beyond its conventional experimentation capabilities. Principles of creative problem solving help establish the role of evolutionary dynamics in generating novelty.

- As indicated by Gero and Kazakov (1996), evolution is creative in the sense of generating surprising and innovative solutions.
- Analogous to creative and innovative problem solving, evolutionary mechanisms improve solutions iteratively over generations (Gero and Kazakov, 1996).
- Ambiguity and lack of clarity about knowledge about existing relations between the requirements for ideal solution and forms that satisfy these requirements (Rosenman, 1997) are useful opportunities for creativity.

- Exploring a search space in an effective and efficient manner and ability to explore alternative search spaces by redefining the problem representation are critical in creative problem solving. To the extent that evolutionary mechanisms that do not have considerable freedom to vary their representations are clearly not creative (Gero, 1996).
- Creativity requires transfer of knowledge and use of metaphor (Holland, 1998) and analogical reasoning across disciplines (Goldberg, 1999). Hence, evolutionary dynamics coupled with ecological perspective that favors transfer is more likely to be creative.

2.2 Models of Creative Cognition

Bottom-up theories of creativity include generate and explore model (Smith and Blakenship, 1991), systems model of creativity Csikszenthmihalyi (1999) and evolutionary models of creativity (Campbell, 1960; Simonton, 1999). The commonality between these models is two-fold. First, they all hypothesize and substantiate that creativity is influenced by synergistic interaction and confluence of environmental factors. Second, their underlying mechanisms are based on the principles of models of evolutionary systems. That is, creative ideas emerge through a process analogous or similar to natural selection process.

Systems Model of Creativity. According to systems view of creativity introduced by Csikszenthmihalyi (1999), creativity is not the product of an individual, but rather the interaction between the individuals, social context, and the problem domain. For creativity to occur, original and novel contributions submitted by individuals should be evaluated for inclusion as new body of knowledge in the domain. Individual contributions are predicated on the practices and knowledge stored in the problem domain, so that novelty can be produced as a variation in the content of the domain. Figure 4 depicts the components and interactions of this model. The domain is considered as a critical component of creativity because it is impossible to introduce a variation without reference to an existing pattern specified in the domain knowledge. The technical contributions made by individuals that produce creative solutions to domain-specific problems. Such technical contributions induce novel variations in the domains that constitute the context.

Generate and Explore Model. The view of creative problem solving as a bottom-up generative process is the basis for the generate and explore model. Introduced by Smith and Blakenship (1991), the model is based on a two-stage process. In the first stage, the individual *generates* pre-inventive forms, which are ideas and conceptual structures that might be useful for creative production. In the second phase that primarily focuses on *exploration* of the pre-inventive structures to interpret their utility for solving specific problems. The generation phase can leverage various cognitive processes. The individual may retrieve prior

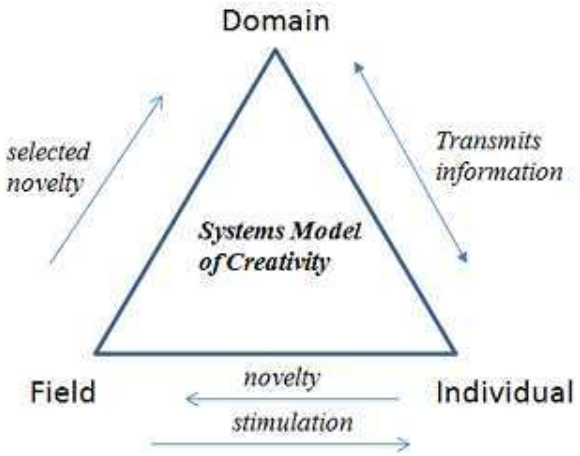


Fig. 4. Systems Model of Creativity

learned concepts for evaluation, or transform concepts in novel ways or configure them in new forms to orient the solution in a new context. Constraints on the final product can be imposed to influence the generation and exploration phases.

Evolutionary Models of Creativity. According to Campbell (1960), three major conditions are necessary for creative thinking: (1) similar to organic evolution, there should be a mechanism to generate new ideas in the form of ideational variation; (2) following the generation of new ideas, specific criteria must be used to select those ideas that are useful for the problem under consideration; (3) the variations that are selected should be retained and used for reproducing new ideas in future generations. The arguments against Campbell's model of creative cognition focus on the need for incorporation of experience and learned behavior. Campbell (1960) extensive quotes Poincare (1908) to signify the role of blind variation at some point in the process to bring true novelty before the production of experience in a particular problem domain.

In elaborating Campbell's view, Simonton (1999) proposes a chance configuration model that applies evolutionary perspective to mental elements, which are fundamental psychological units that are manipulated and combined to generate new forms and complex mental units. Mental elements are elaborated and combined via a process of *chance permutation*, which are carried out in the unconscious. In retaining ideational variations, Simonton (1999) proposes that permutations differ in terms of their stability. Stability is defined as the affinity and coherence of mental elements in the proposed configuration. The greater the stability of a combination, the greater the chance that it will be selected and retained for attention by the consciousness.

3 Generative Parallax Simulation: Basic Concepts

Development of simulation methods that support creative problem solving requires leveraging principles that explain emergence of creativity. The perspective examined in this work is the creative cognition world-view that focuses on bottom-up idea generation and evaluation strategies that enable optimal combinations of explorative and exploitative modes of inquiry.

3.1 An Abstract Model of Creative Cognition

Examination of creative cognition models reveals three main components that interact with each other to produce useful novelty: Domain, Generator, and Evaluator. We define a high-level reference model (see Figure 5) to delineate each component along with its role.

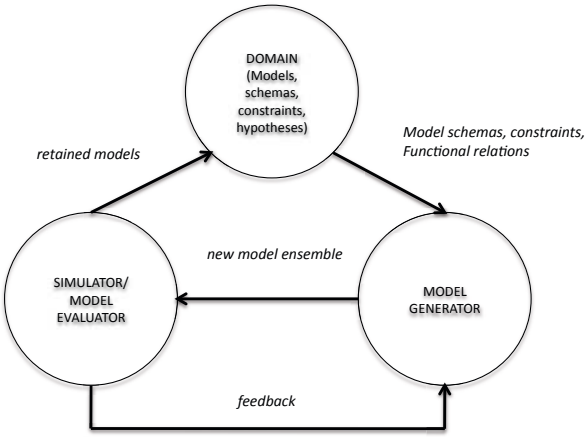


Fig. 5. Generate and Explore Reference Model

Domain: The domain embodies the ensemble of plausible models (problem formulations), hypothesized mechanisms believed to represent referent processes, constraints (e.g., experimental conditions, range of values of known variables), phenomena being explored, plus schemas (e.g., meta-models) used to specify analogues.

Generator: The generation phase of the creative cognition process can be based on any number of novelty generation actions. To be successful in improving creative insight into a problem, a simulation platform and its underlying mechanisms need to be aware of principles and operators underlying the process for generating creative novelty. Sawyer (2008) discusses and illustrates four major operators that often enable creative outcomes:

- *concept elaboration* - extending existing concepts (e.g., analogue) through new features and constraints to obtain more specialized concepts.
- *concept combination* - requires integration of two or more concepts to obtain a new novel concept.
- *concept transfer* - involves establishing a metaphor through analogy to reuse a collection of related concepts in a new context.
- *concept creation* - refers to invention of new concepts that do not exist in the problem domain

Evaluator: Analogue composition is a hypothesis: these components as composed become a mechanism upon execution, and that mechanism will lead to measurable phenotypic attributes that mimic prespecified, targeted, referent attributes. A more interesting analogue is one capable of a greater variety of phenotype mimicry, and for which the mappings from analogue to referent mechanisms can be concretized; conceptual mappings cannot. Improved analogue-to-referent mappings at the mechanism level are expected to lead to deeper insight. Analogues capable of greater mimicry of targeted attributes are retained. Phenomimetic measures are needed to compare phenotype overlap: attribute similarity. Comparative phenometrics will depend on the relative ability of two or more analogues to achieve prespecified measures of similarity to referent. Included will be quantitative validation metrics as well as more qualitative measures, such as behavioral similarities. Comparative phenometrics should also take into account the degree of phenotype overlap and mechanistic similarity between analogues. Substantial multi-attribute similarity coupled with some mechanistic divergence has the potential to catalyze creative leaps. The feedback provided back to the generator improves its effectiveness in selecting the model generation operators through a learning mechanism.

3.2 Abstract Specification of the Structure and Dynamics of GPS

To formalize GPS, we define the structure of the domain of models as a graph of ensembles, $G = (V, R)$, where V is the set of nodes, and each node $v \in V$ denotes an ensemble of models, and R is the set of relations depicting affinity (e.g., similarity in terns of function and form) between the ensembles. Each ensemble E has a neighborhood $N(E)$, which refers to a connected subgraph of G containing E. For our purposes, each ensemble contains a collection of metaobjects, each one of which specifies the schema of a corresponding model. Figure 6 depicts the structure of graph of ensembles. The strength of relations (e.g., $w(i, k)$ or $w(k, i)$) between ensembles signify the degree of similarity analogues in the source ensemble exhibit with respect to phenotypic attributes of the target ensemble's referent.

To evolve model schemas and their metaobjects, we need an encoding scheme. Although the encoding of a schema depends on the purpose of the study and aspects that will be evolved during the process of generative simulation, for simplicity and purpose of demonstration, we may assume for demonstration

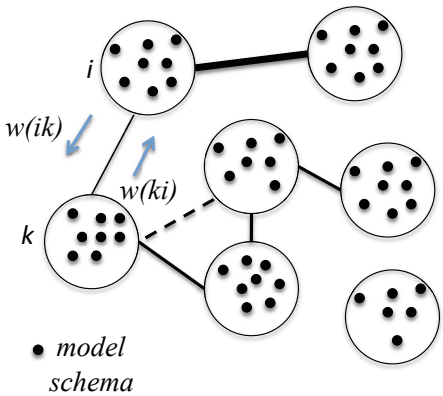


Fig. 6. Graph of Model Ensembles

purposes that each schema is a binary sequence of length n from the space $\{0,1\}^n$. We denote the set of schemas with S and define a population function, P, as

$$P : V \rightarrow S \tag{1}$$

The population function can be extended via a neighborhood function $N : V \rightarrow 2^V$, which returns for a given node $x \in V$, all nodes, $y \in V$ in G, where $(x, y) \in R$. Hence, $P(N(E))$, where $E \in V$, returns all schemas contained in the neighboring ensembles. One can however also access schemas within a given ensemble by taking neighborhood equal to a specific ensemble. At any given round of generation, T, the schema population of the neighborhood of an ensemble, E, is given by $P(N(E), T)$. Similarly, the population of a specific ensemble is defined as $P(\{E\}, T)$, and the number of schemas in ensemble is $N = N_{E;T}$.

Evaluation - Phenometrics: Given the above specification, we need to define evolutionary aspects of the ensemble. Three major factors are of interest within the morphogenesis domain. First, analogues that exhibit behaviors similar to those targeted must be favored, as they generate sufficiently valid behavior, if a similarity measure value exceeds a prespecified threshold. Second, those analogues that use divergent mechanisms yet have a sufficient level of validity may facilitate discovery of novel mechanisms. Consequently, they should be retained. Finally, those analogues that demonstrate success in generating behaviors and features imposed by neighboring ensembles should be favored, as they may be able to extend their usefulness and scope. Analogues that satisfy the constraints of multiple phenotypic attributes will relate to schemas from more than one ensemble: they are expected to have larger impact. The similarity for a given ensemble v is defined as $F_a : S \rightarrow [0, 1]$. $F_a(s)$, where $s \in S$, returns similarity of the analogue with schema s within the ensemble v. For a given schema, its performance is the degree of similarity of corresponding analogue behaviors to targeted phenotypic attributes. The *extent* function, $F_e : N(E) \times S \rightarrow [0, 1]$, measures the degree of relevance of the schema with respect to ensembles in

$N(E)$. The total similarity of schema s in ensemble E is the weighted sum of its validity and extent:

$$f(s) = \alpha_a F_a + \alpha_e F_e \tag{2}$$

where $\alpha_a + \alpha_e = 1$. Adjustment of these parameters enables examination of alternative population evolution strategies. So doing may suggest effective and efficient methods for discovering analogues that are qualified to mimic selected sets of attributes.

Generation: The similarity functions specify how the schemas in the ensemble will be judged. Schema generation uses general schema *transformation operators*. A transformation operator is defined as a function $t : S^m \rightarrow S$ for some integer m. The generative simulation system has a collection of operators, $R = \{t_i, i = 1, 2, \ldots, J\}$, that enable generation of alternative forms and structures. *Elaboration* involves refinement of a schema and is similar to a mutation operation with $m = 1$, while *combination* is analogous to a crossover operation, with $m = 2$. Each transformation operator, t_i, is associated with a weight, w_i, that determines frequency of its application. The generation of new schema involves a stage of schema selection, followed by interactions to create new schemas. At each round, schemas with similarity values less than a predefined threshold are dropped. The remaining population in $P(\{E\}, t)$ representing ensemble E is replaced with a new interim population ϕ consisting of schemas in $P(\{E\}, T)$ with different frequencies. Using the conventional GA fitness proportionate parent selection mechanism, we compute for each schema its probability of selection: $\frac{f(s)}{\sum_{j \in E} f(j)}$. The expected number of copies of each schema (metaobject) in the interim population is then given by $\frac{f(s)}{\overline{f}}$, where $\overline{f} = \frac{1}{N} \sum_{j \in E} f(j)$ the average similarity measure of the population. The frequency of each schema is therefore approximately proportional to its total similarity. The interaction between selected schemas proceeds as follows. A schema is selected from the interim population ϕ. A transformation operator t_j is selected with a probability proportional to its weight w_j. If the arity of the transformation operator is m, then the remaining $m - 1$ schemas are randomly selected from the interim population ϕ. Following application of the transformation operator the produced schema is included in the new population: $P(\{E\}, T+1)$. The interaction process is repeated N times to generate a new full population.

Transfer: Given the set of edges, $R \subseteq V \times V$ of the ensemble graph, G, each edge (E_i, E_j) is associated with two components: w_{ij} and w_{ji}. These components, shown in Figure 6 as the strength of relations, are positive integers that define transfer rates from E_i to E_j and E_j to E_i, respectively, and $Q = \sum_j w_{ij} \leq N$. Each schema s in the ensemble has a propensity to transfer $\mu(s)$, which is a monotonic function of the change of similarity over k iterations. Initially, w_{ij} for each ensemble i is set to a low value. These transfer rates, which emulate conceptual transfer and analogy-based discovery, may change over time. Learning takes place as information about the similarity of copied and transferred models is gathered. If models that are transferred from E_i to E_j improve their average

similarity, the transfer rate for migration from E_i to E_j is updated to increase number of transfers; otherwise, the transfer rate is decreased. At each round of evaluation, for every (E_i, E_j) in the analogue ensembles graph, the population in ensemble i is scanned to locate K schemas with $\mu(s) \geq \gamma_{transferThreshold}$ and from these schemas a subset of size proportional $\frac{m_{ij}}{Q}$ schemas are selected for transfer to E_j.

3.3 Implications of the Ecological Perspective

The generation and transfer mechanisms update the contents and structure of the ensemble graph as an evolving ecology. The network of ensembles enables interaction between models. The boundaries denote separate attributes and targeted objectives in the referent. Ensembles communicate with each other and share models across their boundaries. An overall solution is discovered through continual flows of models so that ensembles in the graph can sustain themselves and improve the impact and usefulness of local solutions. Those models that do not survive after migration to other ensembles are considered as falsified. Exchanges of models in this evolving ecology of ensembles are sustained by pervasive cooperation, because a model that migrates to a new ensemble contributes its traits to its new context. Furthermore, viewing a solution to complex multi-aspect and multi-attribute problem as an evolving ecology achieves stability and resilience through richness and complexity of interaction. Due to synergistic combination of evolution and ecological perspective, the generated solution that is defined in terms of an ensemble of ensembles is flexible due to consequence of multiple feedback loops that keep the solution in a state of dynamic balance. That is, the solution is not biased toward a single targeted attribute; hence, the solution is not optimized toward one specific aspect. Rather, ensembles fluctuate around their optimal forms.

4 Meta-simulation of GPS

Preliminary experiments are conducted to better understand the operating regime and parameter ranges that improve integrated differentiation in a hypothetical model space. To this end, we present a meta-simulation study of evolving ensembles of model schemas. For the purpose of this article, meta-simulation is defined as the simulation study of a simulation method for the purpose of better understanding its behavioral dynamics and patterns that emerge as a result of sensitivity to parameters, which are related to creative cognition model discussed above.

4.1 Conceptual Model for GPS Simulator

As shown in Figure 7, the collection of models is comprised of four ensembles, each constituting a quadrant of the grid. Each ensemble contains a set of model schemas. A model schema is specified as a binary vector of length 40. Each element (i.e., bit) of the vector depicts a trait that a model belonging to associated

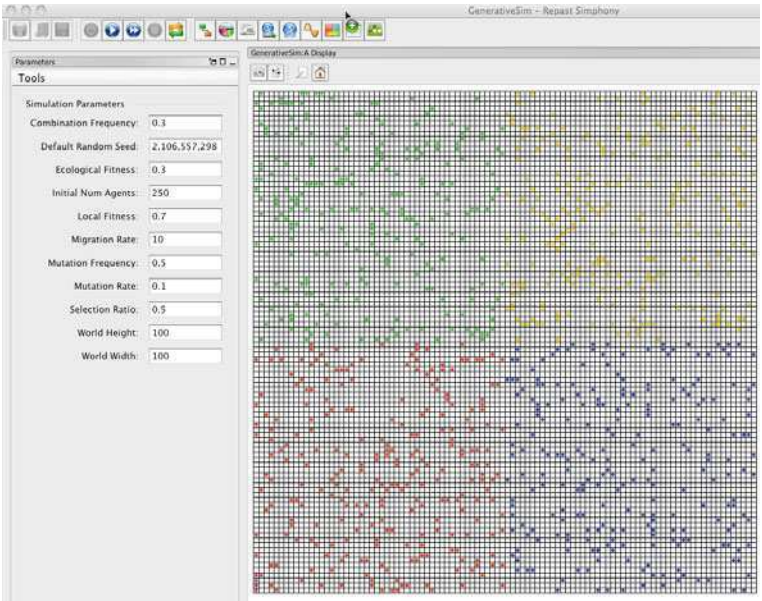


Fig. 7. Initial State of the Meta-simulation

ensemble representing one of the four perspectives or aspects involved in the hypothetical abstract problem. In comparison to the motivating scenario examined in section 2, each ensemble represents a separate targeted attribute.

Each bit in the vector representing a trait is interpreted as a component (e.g., axiom, indicators for existence or lack of a variable, low or high levels of values for a specific variable). Ideally, in realistic problems, a set of components is used as primitive blocks to construct solutions through an evolutionary design mechanism to generate problem representations and configurations to facilitate discovery of novel and useful solutions. The constraints involved in plausible configurations should play a critical role in selective retainment of solutions, but in our simplified scenario, we use a practical solution by favoring those schemas that maximize the value of the bit vector. Given that our goal is simply to examine sensitivity of our evolving ecology framework to parameters discussed below and to interpret emergent patterns pertaining to degree of integrated (useful) differentiation (novelty) in the schema space, as opposed to demonstration of its performance on a specific problem, this simplified treatment for analysis is acceptable for the purpose at hand.

Figure 8 depicts the encoding scheme for elements of each ensemble. There are four ensembles, a, b, c, and d, which are initialized with collections of schemas. Each schema is defined in terms of 40 bits divided into four sections, V_a, V_b, V_c, and V_d, with 10 bits each. At the time of initialization, the binary vector of a schema that belongs to ensemble k is set in such a way that each bit in V_k is set randomly to either 0 or 1. The bits in the remaining sections are set

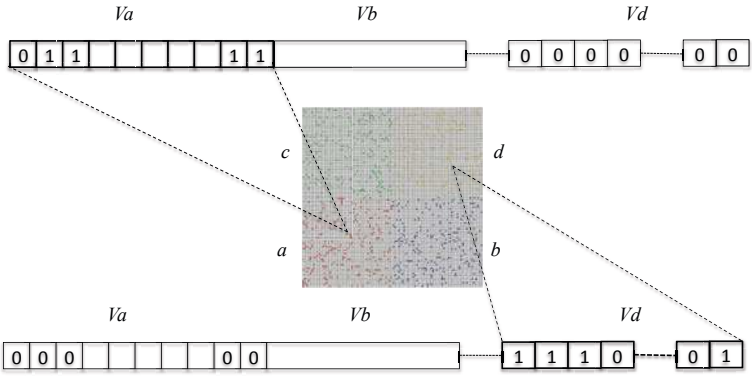


Fig. 8. Encoding of Schemas in Model Ensembles

to 0. Following the initialization phase, the simulation engine starts evolving the ecology of model ensembles in accordance with the *generation*, *evaluation*, and *transfer* mechanisms defined in the previous section. These mechanisms are guided by a set of parameters shown in simulation parameters section of Figure 7. At each time step, list of schemas in each ensemble are evaluated to determine their fitness against objective functions associated with the local ensemble, as well as ensembles in its neighborhood, if ecological fitness is required in the evolution process. The objective functions for each ensemble indicate which configurations of schema vectors are favored in that specific ensemble that targets a particular attribute or aspect of the problem under consideration. For ensembles a, b, c, and d, objective functions are defined as

$$\begin{aligned} O_a &: m(V_a) + m(V_b) + m(V_c), \\ O_b &: m(V_b) + m(V_c) + m(V_d), \\ O_c &: m(V_a) + m(V_c) + m(V_d), \\ O_d &: m(V_a) + m(V_b) + m(V_d). \end{aligned}$$

The mapping $m : V \rightarrow R$ is a function that maps binary vectors (e.g., V) onto real numbers that represent the values of bit vectors. That is, given a bit vector $V_i =< b_j, b_{j-1}, \ldots, b_0 >$, $m(V_i) = b_0 + 2^{b_1} + \ldots + 2^{b_j}$. Note that when coupled with simulations of individual models, actual evaluation criteria will be based on assessment of simulation outputs and their similarity to targeted attribute. This enables exploration and potential discovery of behavioral configurations and schemas that favor models, which can successfully evolve through various transformation operators and feedback to explain multiple phenotypic attributes. Interpenetration of models into multiple ensembles is an indicator for the degree of integration and usefulness of generated schemas to facilitate expressive effectiveness and relevance, while courting for sufficient level differentiation for emergence of novelty.

4.2 Meta-simulation Parameters

Those schemas that score well with respect to a selection threshold are qualified for retainment. They are transferred to the next generation. The parameters of the simulation and their influence on the evolution of model ensembles are defined as follows.

- **Local and Ecological Similarity**: These parameters refer to attribute similarity (i.e., α_a) and extent (i.e., α_e) parameters of the abstract specification, respectively. For instance, in our hypothetical problem, the overall fitness of a schema in ensemble a is defined in terms of attribute similarity and extent parameters: $F = \alpha_a O_a + \frac{1}{3}\alpha_e(O_b + O_c + O_d)$. While O_a depicts the similarity of a schema in the local context in which it was originally defined. The remaining part of the formula facilitates retention of those that are able to migrate to and succeed in other ensembles .
- **Combination Frequency**: Variation and transformation of schemas take place by either elaboration (update of its own traits) or transfers from other schemas. Combination frequency defines the probability at which a combination operator that transfers traits (e.g., components) is selected. Exchanging traits across multiple schemas is a prerequisite for inducing variability and inheritance of generative mechanisms that are successful in multiple analogues.
- **Mutation (Elaboration) Frequency and Rate**: A mutation frequency is the probability that the transformation operation selected during the generation phase involves elaboration or update of its own components. The rate parameter specifies the probability of mutation for a single component during the scan of the components of the section of the binary vector that belongs to an ensemble from which the schema originated.
- **Selection Ratio**: This parameter (i.e., β) controls the degree of receptivity of the evaluator. From a given set of N schemas in an ensemble, βN schemas are selected as being sufficiently phenomimetic to induce a selective pressure toward the evolution of even more phenomimetic solutions.
- **Migration Rate**: This parameter controls the number of schemas that are transferred from a source ensemble to a target ensemble. Each cycle, schemas designated by migration rate are selected and transferred to a neighboring ensemble. The purpose of transfer is to control the rate at which traits and components are exchanged across ensembles. Increased transfer rates are expected to improve analogues and facilitate interpenetration of their mechanistic components.

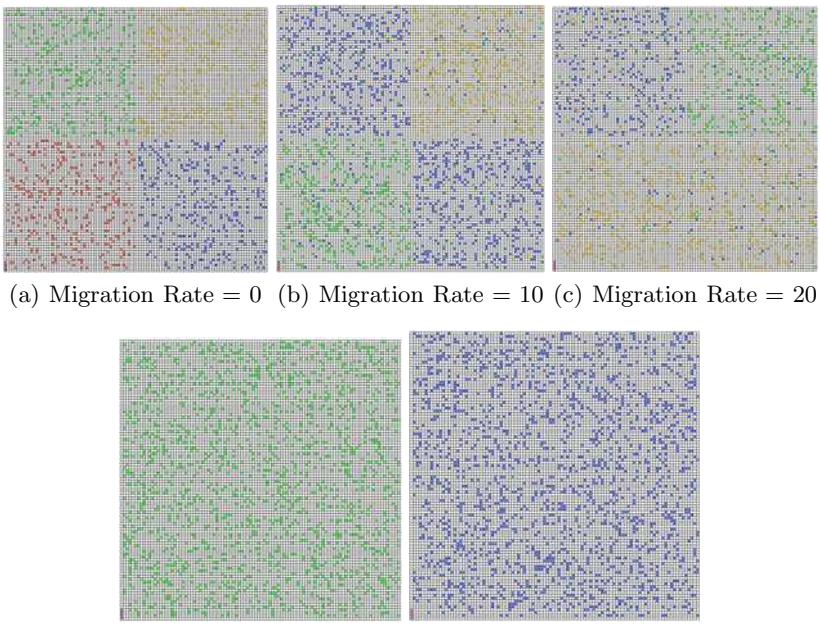
4.3 Qualitative Analysis of Results and Discussion

To study the behavior and sensitivity of GPS to the above parameters, a meta-simulation was performed to observe emergent patterns pertaining to interpenetration of analogue mechanistic components across distinct, yet related, targeted

attributes (i.e., four quadrants). While the capability of an analogue to increase its phenotype overlap and survivability in a new context after transfer is an indicator for its usefulness and extent, dominance of one analogue type across all contexts may be an indicator for lack of differentiation. Failure to achieve balance between differentiation and integration is expected to result in decreased adaptiveness, diversity, and hence an inability of the analogue ensemble to mimic future additions to the set of targeted attributes while retaining similarity for already validated attributes. Just like a controller that is rich and diverse in terms of its possible actions while being resilient in the presence of unforeseen perturbations in its environment, sufficiently differentiated analogue ensembles are more likely to adapt when the set of targeted attributes is expanded.

To better understand and discuss the behavior of generative ensembles of analogues as a creative evolutionary system, we examine emergent patterns under various environmental conditions along with the evolutionary dynamics defined by model migration (transfer) rate, selection ratio, degree and importance of ecological similarity, and frequency of schema combination use.

Migration Rate. The migration rate between ensembles is defined in terms of the number of schemas that are selected randomly from the new population for transfer to a related ensemble. Figure 9 demonstrates emergent patterns as the number of transferred schemas increase.



(a) Migration Rate = 0 (b) Migration Rate = 10 (c) Migration Rate = 20

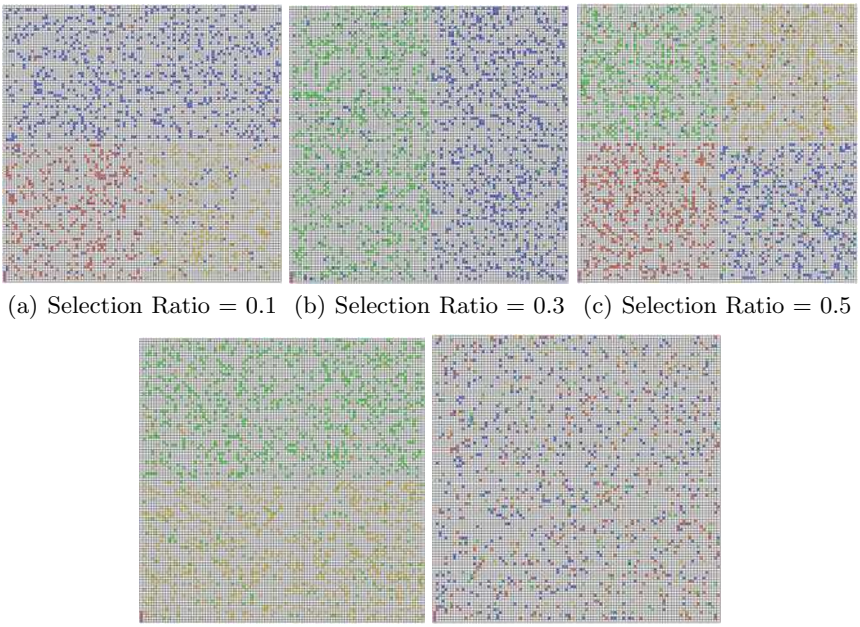
(d) Migration Rate = 30 (e) Migration Rate = 40

Fig. 9. Impact of Migration Rate on Schema Diffusion

In Figure 9-(a), migration rate is 0; hence, ensembles are completely differentiated and original distribution is kept intact. Increasing the rate to 10 schema transfers per iteration, the number of schema types reduces and stabilizes at 3. Further increase in migration rate to 20 schemas per iteration changes the configuration and allocation of schemas; yet, there still exist 3 types of schemas with one schema dominating 2 out of 4 ensembles. Once the migration rate reaches 30 schemas per iteration, the equilibrium state contains a single schema type that dominates all ensembles. Transferring useful schema traits from other ensembles improves both local and relational fitness of schemas that inhabit the target ensemble, as they inherit through combination useful traits to improve usefulness to the source ensemble. More importantly, those transferred models that migrate back to their source domain bring back traits that improve local schemas, which then inhibit interpenetration of future non-local ensembles. Therefore, those types of schemas that improve their local, as well as ecological fitness faster compared to other types start dominating and increasing their interpenetration resulting in decreased differentiation. Therefore, although knowledge transfer (e.g., analogy, metaphor) are powerful mechanisms for inducing novelty, moderate levels of transfer is needed to balance differentiation and integration.

Selection Ratio. Selection ratio (β) controls the size of the original population of schemas that are transferred to the next generation in the same ensemble. That is, small levels of selection ratio indicates high degree of critic and selective pressure toward highly fit schemas. On the other hand, large values are indicative of friendly, criticism-free environment that places little constraint. Figure 10 depicts emergent patterns for increasing levels of selection ratios in the range from 0.1, which imposes high degree of constraint, to 0.9 that is close to a critic-free environment.

None of the configurations, except when selection ratio is 0.9, result in co-existence of multiple schema types in a single domain. While the highest level of selection ratio enables interpenetration and co-habitance of multiple schema types to induce diversity, the emergent organizational pattern points out a disordered state, as opposed to meaningful overlap across ensembles. Since the threshold for survival of a transferred model schema is too low, its co-existence with other schema types in multiple ensembles is not necessarily an indication of balanced integrated differentiation. On the other hand, medium-high and medium-low levels for β enables sustainable penetration of schemas to other ensembles. When β is medium-low (e.g., 0.3), through earlier migrations and combinations, those schema that exhibit slightly higher levels of performance compared to local schema can dominate in a new domain (e.g., ensemble). When β is increased to medium-high (e.g., 0.7), despite significant level of random fluctuations, certain schemas are slightly favored and improve their position for the next iteration. As β increases, the propensity of selective pressure drops significantly, leading to a disordered state as shown in Figure 10-(e). At medium levels of β, the original configuration and degree of differentiation is kept intact since local schemas that have high local fitness find opportunity to reproduce sufficient number of offspring schemas inhibit sustainment of non-local schemas.



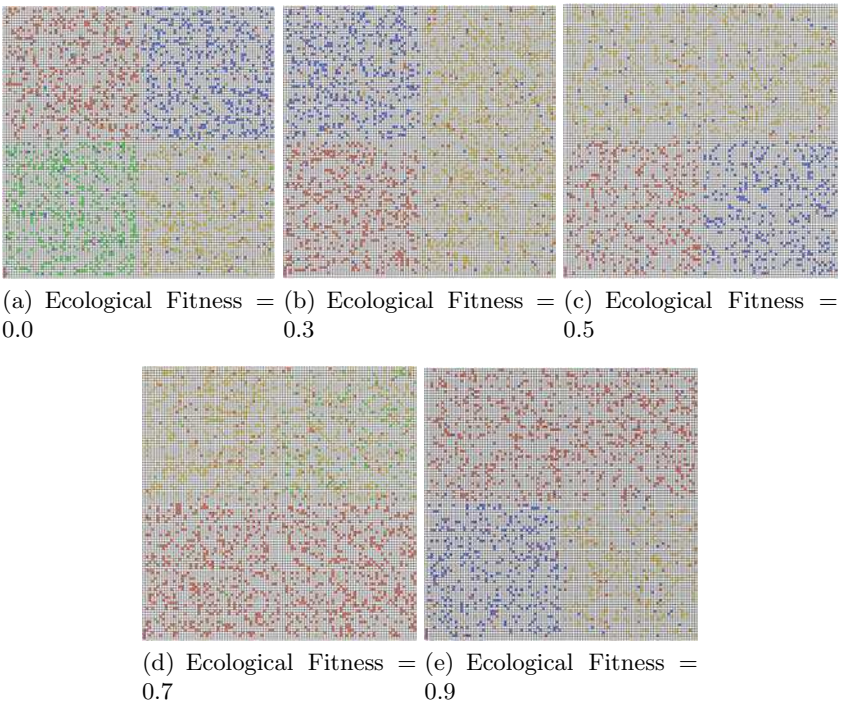
(a) Selection Ratio = 0.1 (b) Selection Ratio = 0.3 (c) Selection Ratio = 0.5

(d) Selection Ratio = 0.7 (e) Selection Ratio = 0.9

Fig. 10. Impact of Selection Ratio on Schema Diffusion

As selection ratio drops to its minimal levels, we observe that selective pressure penalizes experimentation with alternative schema types. The uniformity observed within each domain signifies lack of diversity, which may be explained by high-level of (re)combination frequency in the current configuration. Preliminary results suggest the need for exhaustive exploration of the range between medium-high and high levels of β to determine if there is a critical threshold between the disordered state and desired ordered, yet differentiated phase. Also, interaction between levels of combination frequency and selection ratio needs to be examined to explain uniformity and hence lack of interpenetration within each domain for all levels of β, except when it is high.

Ecological Similarity. To improve interpenetration of schemas with the goal of inducing diversity and resilience, evolution of existing schemas requires consideration of not only fitness against the constraints of the local ensemble, but also its neighboring domains. The expectation is that schemas that exhibit greater similarity in their local ensemble are more likely to survive and sustain when transferred to neighboring ensembles, if relational similarity is factored in. Figure 11 presents a distribution of schemas under different levels of ecological fitness. In part (a) the diffusion and interpenetration of schemas are significantly restrained because schemas that were favored in their local context failed to survive when they were transferred to a new context.



(a) Ecological Fitness = 0.0 (b) Ecological Fitness = 0.3 (c) Ecological Fitness = 0.5

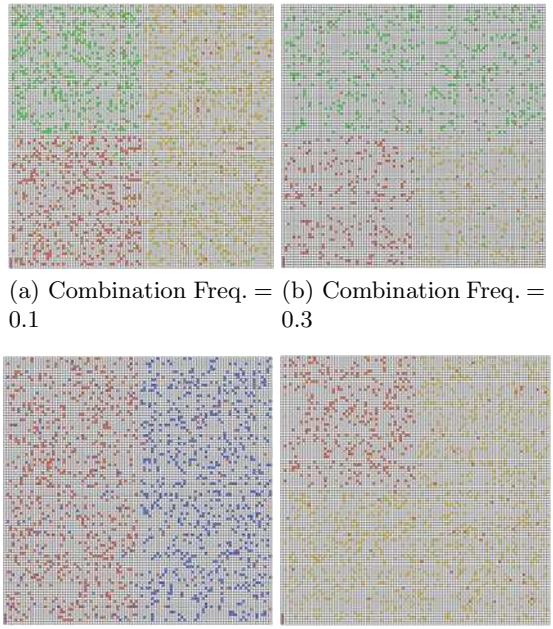
(d) Ecological Fitness = 0.7 (e) Ecological Fitness = 0.9

Fig. 11. Impact of Ecological Fitness on Schema Diffusion

As shown in Figures 11 (b) and (c), small to medium levels ecological fitness avoid global uniformity and keeps ensembles distinct, while the level of differentiation and diversity within each domain is weak. On the other hand, intra-domain diversity and hence interpenetration of model schemas are observable in the domain located in the top right corner when ecological fitness is set to 0.7. Further exploration in this region of operation and interaction with other factors (e.g., combination frequency) are needed to identify when interpenetration peaks. Note that as indicated in Figure 11 (b) each one of the domains become visibly unified and schema diversity disappear. Primary reason for lack of intra-domain differentiation is that schemas that are not necessarily fit with respect to local constraints survive when ecological fitness is high; and these schemas are likely to become extinct and dominated by random fluctuations by even slightly fit schemas that migrate from other ensembles.

Combination Frequency. Schema combination is a powerful transformation operator that we expect will facilitate achieving creative leaps, especially when remote, meaningful and useful associations are made. On the other hand, increased frequency of schema combination is expected to restrain diversity.

As shown in Figures 12-(a), (b), and (c), increase in combination frequency leads to decrease in the types of schemas, leading to global uniformity. Also, for



(a) Combination Freq. = 0.1

(b) Combination Freq. = 0.3

(c) Combination Freq. = 0.5

(d) Combination Freq. = 0.9

Fig. 12. Impact of Combination Frequency on Schema Diffusion

even moderate amount of combination intra-domain diversity (e.g., interpenetration of schemas) drops significantly. In comparison to (c), the pattern in (d), when frequency is significantly increased to 0.9, demonstrates reduction of interpenetration of schemas to other ensembles. In (c), however, although uniformity starts emerging, a small subset schemas originated elsewhere are still able to sustain in neighboring domains. Likewise, in contrast to (a), the pattern in (b), with slight increase in combination frequency, maintains same degree of global uniformity, while degree of interpenetration and diversity within individual domains starts declining.

Summary and Evaluation of Qualitative Analysis. Interaction between multiple parameters require further study. However, preliminary results on sensitivity to selected parameters confirm expectations about the role of combination frequency, knowledge transfer, significance of ecological perspective, and intensity of selective pressure on the degree of integrated differentiation that is conducive to creative discovery. The demand for interpenetration is based on the anticipation that models with distinct mechanisms can co-exist in the same ensemble to suggest alternative explanations for the same phenomena, and hence as a corollary improve discovery process. Yet, complete random and disordered distribution of models makes it difficult to establish coherent explanation and analysis of the phenomena and its phenotypic properties.

Observations with this simplified abstract model suggest the use of moderate transfer rates across ensembles to avoid global uniformity. Presumably, having a single model that can explain all targeted phenotypic attributes seems to be powerful. However, lack of differentiation is less likely to cope with new empirical regularities in new experimental conditions. Experimentation related to sensitivity to selective pressure reveals that non-critical and constraint free environment is likely to lead pseudo-random distribution of models leading to significant entropy and disorder. Optimal levels of differentiation and interpenetration are observed at medium levels of selective pressure. We also observed that ecological fitness levels that are slightly higher than medium levels improve interpenetration, while improving overall global integration. Finally, high degrees of combination frequency as compared to elaboration inhibits interpenetration, while significantly increasing global uniformity.

5 Discussion and Future Work

The proposed approach and its abstract model is capable of evolving and transfer of schemas to improve ability of the ensembles to generate and explore the search space and discover schemas that exhibit better analogue-referent similarity. However, various limitations pertaining to emulation of creative cognition still exist and are discussed in the sequel along with suggestions for future work.

5.1 Improving Autonomy in Schema Evolution and Diffusion

A significant limitation of the proposed strategy in its current form is its reliance on an extrinsic objective function and the simplifying assumption that targeted attributes are fixed and that each such phenotypic attribute is associated with an objective function. The objective function represents the empirical regularity against which the accuracy of models in the ensemble is tested. Yet, behavioral similarity can be based on qualitative assessment of patterns. Although transformation operators of GPS facilitate rich interaction, selection of schemas are exclusively predicated on predefined extrinsic functions, often the degree of embeddedness of schemas within the competition environment in which selective pressure is applied is limited. However, through attraction and repulsion processes based on local or global information, schemas can locally interpret their context, chose from their options of available transformation operators, and evaluate their fitness and effectiveness against requisite phenotypic properties. That is, consideration of implicit encoding of selective pressure in the environment may facilitate self-organization of schemas in a decentralized manner so that qualification of models is decided in a bottom-up manner, instead of explicitly encoding reproduction and generation using a central component. Clustering of model schemas around components that depict phenotypic attributes will be the consequence of self-organization in a decentralized and open-ended generative process.

5.2 Toward Adaptive Growth of Analogue Ensembles for Creative Discovery Systems

The strategy presented in this study is based on a fixed graph structure. Although relations between nodes of ensembles are open to adjustment by reinforcing schema transfer rates to strengthen or weaken the connections, it is assumed that all targeted attributes and phenotypic properties and expected regularities are known in advance. To support the incremental and iterative strategy presented in the motivating scenario, it is desirable to refine the solution to facilitate growth of the graph structure as new opportunities and challenges are provided by the environment. Instead of using the fixed graph structure, an open creative discovery system needs ability to build the structure in a bottom-up manner. Yet, schemas that transfer to new domains (environments) do not have capabilities to develop new modalities and functional relations suitable for their new context. However, our preliminary experiments suggest the potential for having generative and reproducing structures to possess not just one, but multiple phenotypic properties, which involve specific functional relations used in generation of phenotypic properties from information used to encode schemas. Similar to the general feature of evolution by which new functions are performed by organs arise from pre-existing organs, individual schemas may evolve phenotypic structures from existing ones to utilize new types of behavior. Possession of multiple behavioral mechanisms might have an adaptive advantage over those that possess single mechanisms when used in a new context.

5.3 Strategic and Context Sensitive Exploration

Although mechanisms of selection in the creative process are akin to those in evolutionary selection, it is critical to recognize that novelty is not generated randomly. Unlike traditional evolutionary computational mechanisms, which often involve breadth-first style of exploration based on large number of genotypes that are mutated or combined to generate variation and/or facilitate optimization, creative discoveries are created through depth-first exploration. Specifically, novelty is generated *strategically* by taking into consideration the internal model of the relations between the elements of the problem domain. While establishing remote associations and flexibility in combining concepts improve creative result, the process is not random, as combinations should yield meaningful and coherent meanings in the problem domain. The challenge is to define an internal model and constraints for selective elaboration, combination, and transfer of concepts to grow and extend the model base. Furthermore, awareness of the context and its constraints is critical for the evaluator to assess the value and utility of the generated concepts in a meaningful, coherent, and consistent manner. *Context-sensitive* evaluation of produced model schemas may be predicated on the empirical regularities associated with the perspective (e.g., experimental conditions, phenotypic attributes, aspect, resolution, scale) of the problem, for which the ensemble is designed a priori or generated during exploration.

6 Conclusions

Parallax models will allow us to approach a biologically inspired form of modeling derived from synergistic integration of evolutionary dynamics, creative cognition, and ecological perspective. The ability to instantiate, generate, transform, execute, and if necessary, evolve multiple models, in parallel, all of which take similar but slightly different perspectives on the same referent (biological) system, opens the door to the automatic generation and selection (by falsification) of many somewhat different hypothetical, including non-intuitive mechanisms for that referent. In other words, the above ability would allow us to construct, execute, and falsify, many more hypotheses for the way a biological system works than can achieved feasibly through our current sequential, iterative, modeling methods. Modeling throughput would increase exponentially. Such an exponential increase in model and hypothesis throughput would promote creative discovery and increase opportunities for creative leaps; that increase is necessary for generative simulation to begin significantly supplanting some of the current trial and error methods in domains like development of new therapeutics.

Development and use of GPS has potential to advance computational science and achieve targeted objectives on at least two fronts: life science experimental methods and information science and theory and methodology of M&S. The outcome envisioned could in time change how biological research is done and how life scientists are trained, while opening new territories for systems engineering. Demonstrating faster-paced methods for achieving a deeper understanding of morphogenesis will catalyze additional scientific advances on other fronts.

References

Amabile, M.T.: Creativity in Context: Update to the Social Psychology of Creativity, Westview, Boulder, CO (1996)

Campbell, D.T.: Blind variation and selective retention in creative thought as in other knowledge processes. Psychological Review 67, 380–400 (1960)

Csikszenthmihalyi, M.: Implications of a systems perspective for the study of creativity. In: Handbook of Creativity, pp. 313–338 (1999)

Davis, P.K., Bigelow, J.H.: Exploratory analysis enabled by multiresolution, multiperspective modeling. In: Proceedings of the 2000 Winter Simulation Conference, pp. 127–134 (2000)

Engelberg, J.A., Ropella, G.E., Hunt, C.A.: Essential operating principles for tumor spheroid growth. BMC Syst. Biol. (2008), `http://www.ncbi.nlm.nih.gov/pubmed/19105850`

Fishwick, P., Zeigler, B.P.: A multimodel methodology for qualitative model engineering. ACM Transactions on Modeling and Simulation 2(1), 52–81 (1992)

Gero, J.S.: Computers and creative design. In: The Global Design Studio, National University of Singapore, pp. 11–19 (1996)

Gero, J.S., Kazakov, V.: An exploration-based evolutionary model of generative design process. Microcomputers in Civil Engineering 11, 209–216 (1996)

Goldberg, D.: The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. In: Evolutionary Design by Computers. Morgan Kaufmann, San Francisco (1999)

Grant, M.R., Mostov, K.E., Tisty, T.D., Hunt, C.A.: Simulating properties of in vitro epithelial cell morphogenesis. PLoS Comput. Biol. 2(e129) (2007)

Holland, J.H.: Emergence: Chaos to Order. Oxford University Press, Oxford (1998)

Kim, H.S.J., Park, S., Yu, W., Mostov, K.E., Matthay, M.A., Hunt, C.A.: Systems modeling of alveolar morphogenesis in vitro. In: Proc. ISCA 20th International Conference on Comp. Appl. Ind. Engr., pp. 141–144 (2008)

Kim, S.H.J., Park, S., Yu, W., Mostov, K.E., Matthay, M.A., Hunt, C.A.: A computational approach to unravel cellular principles of alveolar morphogenesis. Technical Report: UCSF/UC Berkeley Joint Graduate Group in Bioengineering, University of California, San Francisco (09-PONE-RA-08124) (2009)

Lam, T.N., Hunt, C.A.: Discovering plausible mechanistic details of hepatic drug interactions. Drug Metab. Dispos. (published October 20, 2008), doi:10.1124/dmd.108.023820

Mitchell, B., Yilmaz, L.: Symbiotic adaptive multisimulation: An autonomic simulation framework for real-time decision support under uncertainty. ACM Transactions on Modeling and Computer Simulation 19(1), 1–31 (2008)

Poincare, H.: Mathematical creation. In: The Creative Process: A Symposium, Mentor (1908)

Rosenman, M.: The generation of form using and evolutionary approach. In: Evolutionary Algorithms in Engineering Applications. Springer, Heidelberg (1997)

Sawyer, K.: Group Genius: The Creative Power of Collaboration. Basic Books, New York (2008)

Schneiderman, B.: Creativity support tools: accelerating discovery and innovation. Communications of the ACM 50(12), 20–32 (2007)

Simonton, D.K.: Origins of genius: Darwinian perspectives on creativity. Oxford University Press, Oxford (1999)

Smith, S.M., Blakenship, S.E.: Incubation and the persistence of fixation in problem solving. American Journal of Psychology 104, 61–87 (1991)

Ward, T.M., Smith, S.M., Vaid, J.: Conceptual structures and processes in creative thought. In: Creative Thought: An Investigation of Conceptual Structures and Processes, pp. 1–27. American Psychological Association (1997)

Zeigler, B.P., Oren, T.: Multifaceted. multiparadigm modeling perspectives: Tools for the 90s. In: Proceedings of the 1986 Winter Simulation Conference, pp. 708–712 (1986)

Chapter 10
Establishing a Theoretical Baseline: Using Agent-Based Modeling to Create Knowledge

Jose J. Padilla[1], Saikou Y. Diallo[1], and Andres A. Sousa-Poza[2]

[1] Virginia Modeling, Analysis, and Simulation Center (VMASC),
1030 University Blvd., Suffolk, VA
[2] Engineering Management and Systems Engineering Department,
Old Dominion University, Norfolk, VA
{jpadilla,sdiallo,asousapo}@odu.edu

Abstract. This chapter focuses on the application of a methodology and method on how to create theory from existing theory using Modeling and Simulation (M&S). Unlike traditional approaches of building theory based on direct observation or modeling observable phenomena, the methodology and method presented focus on creating a model out of derived premises from literature on the phenomenon of interest. This model is later simulated to gain insight into the phenomenon through the study of patterns and analysis of data. The M&S process provides traceability, structure, and precision while checking for preconceived ideas about the phenomenon by comparing them with theories found in the body of knowledge. The development of a theory of understanding is presented as a test case. The paradigm of choice was agent-based modeling (ABM). ABM was found to be the most appropriate for the test case as derived premises can be coded as rules of interaction.

Keywords: Theory Building Methodology, Modeling and Simulation (M&S), Agent-Based Modeling (ABM), Systems Engineering (SE).

1 Introduction

In a recent workshop at the Virginia Modeling, Analysis & Simulation Center (VMASC), researchers and practitioners debated whether Modeling and Simulation (M&S) was a set of techniques, which could amount to a discipline, and that could be used to make science. Needless to say there was no consensus in which part, of an apparent spectrum, M&S was located. One group argued that M&S are techniques that some disciplines use to enhance their problem solving and/or decision making capabilities. The use of M&S' techniques takes place when phenomena of interest are expensive - resources and/or time - or dangerous to study. In these cases, models and simulations are used to studying, for instance flow dynamics in aerospace engineering, logistics in industrial engineering, or effects of feedback in decision making in business schools among others. A second group argued that M&S is the grouping of these techniques under one discipline's umbrella. Such discipline focuses on studying and improving the use of such techniques. Further, this group suggested that there are topics that are of unique interest of M&S as a discipline, namely interoperability of systems and

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 259–284.
springerlink.com

composability of models. A third and last group presented that M&S is a science. The science perspective argued that M&S serves the greater purpose of creating new knowledge through the rigorous use of the aforementioned techniques in combination with mathematical and statistical means to gain insight into problems. The departure point in this perspective is the need of answering a modeling or research question. The science perspective seeks to create new theories or hypotheses using M&S that help explain observable/measurable or observable/non-measurable phenomena. Dealing with observable and measurable phenomena belongs to the empirical realm whereas M&S focuses on replicating a real life phenomenon in an artificial world. In this case, M&S serves as a constructive environment that provides descriptive and/or predictive capabilities to the researcher. This environment becomes the reality for the researcher and conclusions in this environment are expected to be comparable to those that reality would have given had the researcher had access to it. This approach is widely used in social sciences and in engineering.

A less explored idea in engineering is that of using M&S to develop new theory, especially about ill-defined problems or phenomena. When dealing with problems for which there are not measurements and even no consensus on what the problem is, they are quickly catalogued as problems outside of engineering. However, these problems have, in some cases, a negative effect on engineering activities. Such is the case of problems found by systems engineers.

In systems engineering, as in most engineering disciplines, academicians and practitioners are focused on solving well-defined problems leaving ill-defined problems outside of scope. To do so, methodologies and techniques need to be created that assist individuals in problem identification and problem solution through knowledge application. However, in order to apply knowledge, knowledge needs to be created. Lessons learned and patents, for instance are some of the pragmatic results of knowledge creation. Another instance of knowledge creation is that of theories. Theories are explanation of a phenomenon; they say what the phenomenon is, what it does and how it does it.

This chapter focuses on the premise that insight can be gained through M&S for engineering activities. This insight, it is argued, is conducive to the creation of knowledge especially in areas such as systems engineering.

This chapter is organized as such: section 2 provides a background on systems engineering and the need of theory that considers problems not addressed by systems engineering techniques. Section 3 provides a background on theory and theory development. Section 4 focuses on the importance of M&S and its relevance as a rationalist approach to theory development and some of the methodologies and methods found in the body of knowledge. Section 5 presents a test case of how a methodology and method that use M&S can be applied to assess how understanding contributes to human-driven complexity. Human-driven complexity is a problem of importance to systems engineering. Finally, section 6 presents final remarks and conclusion of this work.

2 Systems Engineering and Its Challenges

Systems engineering is a compelling case for using M&S as it focuses on problems considered complex. These problems are complex not only because of their size in

terms of elements and interactions, but also because of the uncertainty human beings bring to the table. Systems engineering, as a multidisciplinary discipline, must deal with these issues in a multidisciplinary manner which implies the consideration of the effect of individuals in conjunction with technological challenges. However, systems engineers have mostly focused on technical solutions to problems. These solutions depart from the premise that the corresponding problems were objectively defined and the solutions can be ergodically defined. Further, this focus on solutions leave little space for new theory development especially when there are problems that affect SE that cannot be objectively defined and solved with corresponding technical solutions.

This focus on objectivity has been challenged by methodological approaches such as Soft Systems Methodology – SSM – [1] and from theoretical efforts such as system of systems engineering - SoSE – [2]. Both efforts depart from the premise that problems as the result of objective consensus is seldom if at all achieved. When problems are not agreed upon, but still are perceived as problems by some, they are called problem situations. Vennix [3] in p. 13 posits the nature of these problems as:

> *One of the most pervasive characteristics of messy problems is that people hold entirely different views on (a) whether there is a problem, and if they agree there is, (b) what the problem is. In that sense messy problems are quite intangible and as a result various authors have suggested that there are no objective problems, only situations defined as problems by people.*

The implication of this description is that in problems where more than one individual is involved, we should not be referring to them as problems. Flood and Carson [4] also make a point about the nature of these problems. They present that the hard school accepts that problems exist and it can be known what the problem is. The soft school, according to [4] p. 98, "accepts plurality in human understanding and interests rejects the hard view, preferring to assume situations are problematic rather than to accept that problem exists."

SSM and SoSE depart from the premise that they are working in problems outside of SE domain. [5] presents a list of characteristics that these problem have that leave them outside of the realm of SE.

- Holism: they suggest that problems within complex systems should not be reduced to the component level. This is echoed throughout the literature, however, there are not suggested methods how to holistically deal with this type of problems beyond the consideration of looking at everything.
- Complementarity: this concept based on systems theory says that there are multiple perspectives where each and every one is both correct and incorrect. This principle makes more obvious the difficulty of holism in this kind of problems given that not individual can have a perspective of everything.
- Pluralism: Pluralism is in line with the complementarity principle. However, it takes into account that each individual has a different objective or purpose within a group setting. They suggest that these differences may be due to different worldviews, theories in use, or predispositions towards a problem.

- Emergence: they refer to emergence as the property of the system problem domain to evolve over time that affects the effective establishment of requirements. Emergence is considered within the broader literature as the behavior of a system that is due to the behavior of individual parts. This is where the idea of holism makes sense in studying this type of problems and why reducible approaches do not capture the effect of the interactions among parts.
- Boundaries: they contend that these problems are considered open system problems; with ambiguous, fluid, and negotiable boundaries. These characteristics also make these problems difficult to map action and outcome which is highly dependent (outcome) on initial conditions.
- Metasystem: it refers to the integration of multiple systems and the challenges that integration has on the metasystem.
- Context: context is referred as the "circumstances, factors, conditions, and patterns that both enable and constrain a complex system solution, deployment of that solution, and interpretation of the results of deployment." (Keating in [5] p. 27). The importance of the concept is highlighted in [6].

This list is a sample of characteristics attributed to this kind of problems in the literature. As it can be observed from this list, not only is there much ambiguity in the use of terms but also there are no conditions under which these characteristics arise in a problem or if they are presented simultaneously. Although these problems are suggested to be outside of the interest of SE, they are still encountered by system engineers. Therefore, *these problems are still SE problems for which SE approaches are insufficient to address them*. This is where new theories that increase the coverage of SE are highly needed. Further, new theories need to be rigorous and properly bounded for them to be used as baseline for future research.

Some areas where new theory from a SE perspective could be useful are:

- Complexity: almost used as a buzzword for something considered "extremely" difficult is not properly defined in SE beyond its basic definition of something with a great number parts and relations among parts. Unlike work in areas such as mathematics and computer science that have accepted definitions of complexity, complexity has an ambiguous meaning in SE. This ambiguity stems in part that there is not agreed definition on complexity or measure for that matter, especially when a part of it is human-driven.
- Multiple stakeholders: complementarity and pluralism are conditions present in SE problems. Interdisciplinary work that considers group decision making and problem solving should be integrated to the core of work in SE. Problems under mechanic-unitary contexts are the exception and not the rule.
- Holism and emergence: How can an individual be holistic where complementarity and pluralism are present? Holistic may take a different connotation with the idea of emergence given that the focus is not on the whole, but on emergent patterns [7] According to [7], if a reductionist perspective bases its analysis on parts, a holistic perspective bases its perspective in the identification of emergent patterns or lack thereof and not on the whole problem. [8] p. 11 mentions that "only if we use a holistic

approach, by considering both the both the bottom-up and the top-down pattern formation process, can we understand the emerging patterns and dynamics." In other words, holism is required to deal with or describe problems that present emergence unlike problems that can be described through its parts.

Although not an exhaustive list, these areas provide a glimpse on the problems faced by systems engineers. Problems that were so far thought to be outside of the SE realm are affecting SE efforts. In particular, problems for which their formulation and the formulation of solutions is extremely difficult. Given that there are many perspectives of a problem, it is highly likely that there are many problem formulations. Likewise, as there are many problem formulations, there are many possible solutions. The resulting formulation, even under a consensus, cannot be tested and its possible effects cannot be foreseen with certainty. This is where efforts such as SSM and SoSE play a role into considering perspectives about the problem and its formulation. Further, it also shows the lack of appropriate theories that provide insight into areas such as human-driven complexity, how to conduct holism, or how to adjust for emergence. These new theories should provide much needed definitions and ways of assessing problem situations in a manner that assist systems engineers' efforts.

3 Theory and Theory Creation

Theories are an important output of the knowledge creation process and one of the most important goals of conducting research.

Theory development is contingent on how we create and justify knowledge claims, and from a research point of view, using empirical or rationalist approaches. [9] suggests two independent strategies for theory construction. The first one he calls a *generalizing theoretical strategy* whose objective are to explain and to generalize about the lawful phenomena of open systems. It has a systematic structure (hypothetico/deductive), contains ordinary language, uses inductive abstract methods, and its possible result is the consolidation of theories or data. The second he calls *pure theoretical strategy* whose objective is to predict the behavior of lawful phenomena in close systems. It has a formal structure (mathematics, logic) and it contains no ordinary language, uses idealization as a method and is the accumulation of theory. [9]'s position is consistent with [10] position of the dual meaning of the word theory. For [10], the word theory can mean practical theory or pure theory; the former refers to a generalization developed as a basis for judgment and analysis of facts to anticipate the future and guide action whereas the latter refers to speculation without reference to practical application, and not need to describe reality. According to [9] p. 192, "generalizing and pure theoretical strategies are not mutually exclusive and in practice more sociological theories reflect their mixture, however, some of the features may be incompatible or at least independent". Before deepening in the different theory building approaches, let's take a look at a more general approach.

At a general level, [11] suggests that a complete theory must contain four essential elements:

> *What: which factors (variables, constructs, concepts) logically should be considered as part of the explanation of the social or individual phenomena of interest? Two criteria exist for judging the extent to which we have included the "right" factors: comprehensiveness (i.e. are all relevant factors included?) and parsimony (i.e. should some factors be deleted because they add little additional value to our understanding?).*
>
> *How: Having identified a set of factors, the researchers next question is: how are they related? Operationally this involves using arrows to connect boxes. Such a step adds order to the conceptualization by explicitly delineating patterns. In addition, it typically introduces causality. Although the researcher may be unable to adequately test these links, restrictions in methods do not invalidate the inherent causal nature of theory. Together the What and How elements constitute the domain or subject of the theory.*
>
> *Why: What are the underlying psychological, economic, or social dynamics that justify the selection of factors and the proposed causal relationships? The rationale constitutes the theory's assumptions- the theoretical glue that welds the model together. During the theory development process, logic replaces data as the basis for evaluation.*
>
> *Who, Where, When: These conditions place limitations on the propositions generated from a theoretical model. These temporal and contextual factors set the boundaries of generalizability, and as such constitute the range of the theory.*

An empirical approach to theory development is proposed by [12]. [12] sees it, ideally and initially, as a logical, linear sequence of phases that later becomes circular and self-repeating when the researcher becomes more familiar with it. The steps as he presents them are (p. 12-18):

- *Preliminary Phase*, which implies defining the theory domain and that familiarity exists with the phenomenon from the researcher's side in the domain in question.
- *Identifying Inadequacies in Present Theory* or what it is more commonly known as identifying the "gap". He suggests that answers to questions such as: is there new evidence or experience which appears to challenge existing theory? Or are there conflicting theories about certain areas? Shed the existing inadequacies in existing theories.
- *Making Explicit One's Value-Orientation*, which reflects the worldviews, predispositions and biases of the researcher. In this, the empirical researcher

tries to be as objective as possible in collecting data to test his/her hypotheses.

- *Selecting Appropriate Constructs and Models* in which the researcher selects "a central integrating construct capable of encompassing the entire focal domain." [12] Not only does the researcher needs to define constructs, but also develop theoretical models in order to define relationships among the constructs.
- *Developing a Constructual Framework*, in which the researcher uses the previously selected constructs and models to build a more complete framework.
- *Formulation of Hypotheses*. Hypotheses need to be formulated in observable and measurable terms that allow for experimental verification.
- *Testing Hypotheses* according to rigorous application of the scientific method.
- *Formulating theory* as the result of the testing of the hypotheses being aware of not generalize beyond the boundaries of the data.
- *Concretizing* or finalizing theory formulation and transferring the theory to practical operating terms.
- *Empirical Testing and Utilization* which implies the broader use of the theory if viable.
- *Theory Building as a Circular Process* which starts all over again with identifying inadequacies and ways of improving the generated theory.

A rational approach for theory development is presented by Weick in [13]. In this, Weick poses the case of theoretical work not limited by validation (in the correspondence sense), but by usefulness by using mapping, conceptual development, and speculative thought. He argues in p. 519 that "when theorists build theory, they design, conduct, and interpret imaginary experiments". Imaginary experiments may take the form of conjecture simulated possible scenarios out of representations built from interviews, reports, or observation. These possible scenarios allow for the inclusion of a greater number of heterogeneous variations, more selection criteria, and greater diversity. Overall, he suggests three steps: *Formation of problem statement* that can vary in detail, accuracy and incorporated assumptions. *Thought Trials* or conjectures about how to solve a problem are usually in the form of if-then-else statements. Weick contends that a "greater number of diverse conjectures produce better theory than a process characterized by a smaller number of homogeneous conjectures". Finally, *Self-selection criteria* are the alternatives to select thought trials. In this case also applies the premise that the greater the number of diverse criteria, the higher the probability that those conjectures which are selected will result in a good theory.

Either empirical or rational approaches have a basic requirement: rigor. Rigor, in both cases, translates into a well structured process for which conclusions can be traced back to originating hypotheses or premises. In addition, precision plays a key role on the process of building theory by eliminating any ambiguity that detracts from the value of the created knowledge. On the empirical case, validation takes the form of justification based on data and their correspondence to observable event that leads to the eventual confirmation of a theory based on testing. Empiricism is based on the

correspondence theory of truth that says that truth consists in a certain agreement or correspondence between a statement and the so-called "fact" or "reality" [14]. According to Popper in [15]: "Truth is correspondence with the facts (or with reality); or more precisely, that a theory is true if and only if it corresponds to the facts." (p. 420)

On the rational case, validation takes the form of coherence of premises within a system of premises that are considered to be true. Rationalism is based on the coherence theory of truth. Firth in [16] posits that:

> *The coherence theory of concepts is a doctrine that all our concepts are related to one another in such a way that we cannot be said fully to have grasped any one of them unless we have grasped all the others: they form an organic conceptual scheme, it is said, a system of meanings which cohere in such a way that introducing a new concept at any one point in the system has repercussions which are felt through the system.*

Coherence theory of knowledge holds that knowledge claims require justification, but also that no belief can be justified except by reference to other beliefs [17].

It is noted that in most cases under the empirical paradigm, theory building and theory testing are bundled under the theory creation process despite being two separate processes. Approaches such as grounded theory, which are empirical by nature, are focused on developing new theory without considering how the new theory is to be tested. These approaches are considered by some researchers as soft approaches due to their exploratory nature which entails not having defined variables or constructs for measurement. These variables or constructs are supposed to emerge during the process of observation of the phenomenon of interest. More traditional empirical methods depart from hypotheses which provide the bases for testing a theory. However, the theory in this case has already been formulated. Under the rationalist paradigm, testing can be seen as a test for consistency of the assembled system of premises as it explains a phenomenon and the degree of how well the resulting theory explains the originating premises without contradiction.

SE has benefited from both approaches. For instance, from empiricism when testing products developed under the SE process and from rationalism when speculating about new products through the use of modeling and simulation (M&S). M&S, although a purely rational approach, is considered empirical by many when results from simulations are comparable to data obtained from reality. However, in most cases, M&S is used when access to reality is limited due to resources or personal harm. The role of M&S in theory building is presented in the following section.

4 Building Theory through M&S

To define modeling and simulation and its methodological impact the concepts of systems, model, and simulation need to be defined. These concepts play an important role on creating a world that replicate reality where questions can be asked about the phenomenon of interest.

According to [18]:

> *A system is a set of two or more elements that satisfies the following three conditions:*
>
> - *The behavior of each element has an effect on the behavior of the whole.*
> - *The behavior of the elements and their effects on the whole are interdependent*
> - *However subgroups of the elements are formed, each has an effect on the behavior of the whole and none has an independent effect on it.*
>
> *A system, therefore, is a whole that cannot be divided into independent parts. From this, two of its most important properties derive: every part of a system has properties that it loses when separated from the system, and every system has some properties – its essential ones – that none of its parts do…The essential properties of a system takes as a whole derive from the interactions of its parts, not their actions taken separately. Therefore, when a system is taken apart it loses its essential properties. Because of this – and this is the critical point – a system is a whole that cannot be understood by analysis.*

This account brings to the foreground the holistic premise that no reductionist approach should be able to capture a system. However, the positivistic idea of breaking systems into parts and relations among parts is one of the most powerful and used ideas of systems perspective so far. This idea, for instance, has worked well for systems engineers when dealing with technologically oriented systems.

According to [19] an observer must be careful as referring to "the system". This designation "may refer to the whole system quite apart from any observer to study it – the thing as it is in itself; or it may refer to the set of variables (or states) with which some given observer is concerned" (p. 106). He posits that the observer must give up any ambition to know the whole system and focus on a partial knowledge of the system that although incomplete, it is sufficient for practical purposes. [20] also highlights the importance of the observer of a system. Beer posits that the purpose of the system is given by the observer, making it not only subjective, but also conditioned to the acceptance or rejection of boundaries.

From and M&S perspective it can be said that the system is a way of capturing the phenomenon of interests from reality, the one an individual is interested in modeling. However, a way an individual has to represent that system is through a model.

According to [21], a model is a representation of a system, entity, phenomenon, or process. According to Zeigler et al. in [22], a model is a system specification, such as a set of instructions, rules, equations, or constraints for generating input/output behavior. [22] go on to say that a model must be understood with respect to a source system – real or virtual environment that we are interested in modeling - and an experimental frame – which is a specification of the conditions with which the system is observed or experimented –. This is consistent with [23] premise that "a model is intended to represent or simulate some real, existing phenomenon, and this is called

the target of the model" (p. 4). In other words, a system from reality cannot be fully captured given that each observer has his/her own representation of it and each one of these representation has the potential of becoming a model.

A simulation is the execution of a model to replicate its behavior [22]. [21] define simulation as the act of using a simulation engine to execute a dynamic model in order to study its representation of the model's behavior over time. [24] define it as a method that involves creating a computational representation of the underlying theoretical logic that links constructs together within a world. These representations are then coded into software that is run repeatedly under varying experimental conditions in order to obtain results. This position is consistent with [25] that present simulation as used as a method of theory development given that we can express theories as procedures in the form of a computer program, which is more precise than the textual form of the procedure, which is helpful in refining the theory.

The triad of system, model, and simulation and its implications are highlighted by the semiotic triangle of M&S (Figure 1). The semiotic triangle comes from the work of Ogden and Richards in 1926 reflecting that people discuss a referent at the concept level not at the referent level. This implies that we debate *concepts* from reality and not reality itself. Likewise for M&S; given that system cannot be capture (basic premise of M&S), we can capture it through a model that later we implement on a simulation. The variation in how we perceive and understand reality leads to different models of that reality; all models correct, but all incomplete as problem situations premise is.

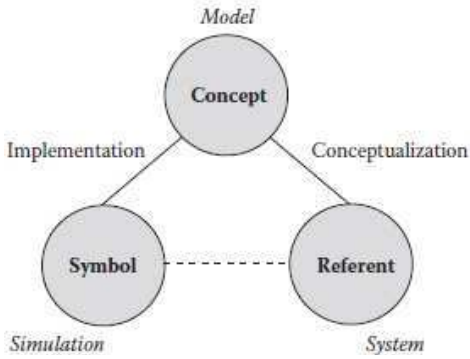


Fig. 1. The Semiotic Triangle of M&S [26]

The impact of modeling and simulation in this work is better expressed by [24]:

> *Simulation is particularly suited to the development of simple theory because of its strengths in enhancing theoretical precision and related internal validity and in enabling theoretical elaboration and exploration through computational experimentation. In particular, simulation relies on some theoretical understanding of the focal phenomena in order to construct a computational representation. Yet simulation also depends on an incomplete*

> *theoretical understanding such that fresh theoretical insights are possible from the precision that simulation enforces and the experimentation that simulation enables.*

The point brought up by this excerpt is the capability generated by M&S to not only provide structure and precision in a rationalist effort, but also an experimental setting where an individual can generate data and generate insight from the analysis of data. This is especially true when dealing with complex phenomena where it is difficult to establish causality allowing the researcher to explore possibilities and test the boundaries of new theories.

According to [27] the interest of doing simulation is to work on a "target" or real world phenomenon in which the researcher is interested and create a "model" of this target which is simpler to study than the target itself hoping that the conclusions drawn about the model will apply to the target. Subsequently, this creates the ability to handle models with numerically indeterminate parameters such as unknown initial conditions or coefficient of equations [28]. [28] mentions that most simulation models allow substituting these formal parameters by hypothetical numerical values making them useful for making inferences for two reasons: 1) it allows to study the consequences of a model without empirically based parameters estimates which generally make conclusions time- and context dependent; and 2) it enables us to study models with many parameters, thus reflecting the full complexity of the verbally formulated theories behind them.

According to [24] simulation has become highly significant as a methodology because not only can it provide superior insight into complex theoretical relationships among constructs especially when empirical limitations exist, but also because it can provide an analytically precise means of specifying assumptions. This insight into complex theoretical constructs is even more important given that, because of the nature of complexity, we may not even be able to establish causal relationships between action and response, between input and output. This implies that any multiple of perspectives can be equally valid in describing the phenomenon due to multiplicity of outcomes. Each one of these perspectives is now necessary and all need to be considered. However, empirically this cannot be done. This is where simulation comes into place; as placing reality as a subset of the perspective, perspectives that now become possible alternatives. This is consistent with the assessment in [29] that computer simulation can be used for exploring substantive theories and theory development, not by the use of data or empirical testing, but by the derivation of consequences of formal assumptions about a phenomenon, (their context was dyadic processes).

[29] are concerned with answering two questions: does the model closely represent the interpersonal processes about which they are theorizing and does it derive consequences correctly? To achieve this they suggest some rules:

1. Every tie from the simulation to the model to the substantive theory needs to be made explicit.
2. The way each algorithm in the simulation works needs to be laid out so others can judge its appropriateness (make aspects of methodology explicit).

3. Every constraint on variables, parameters, numbers of runs and so forth need to be justified.
4. All justification must be in light of the substantive interpretations to be made of the model being simulated.

Hanneman in [30] states that simulation models are formalizations of theories applied to particular scenarios making them concrete and explicitly dynamic, extremely useful in understanding and revising theory because they provide explicit and systematic way of deducing implications about outcomes overtime. Hanneman goes to say:

> *Models produced by theorists often suggest how a theory may play out in particular circumstances, but they tend to be quite vague about the explicit causal dynamics. Models produced by empirical analysts tend to be either restricted by the necessities of the mechanics of parameter estimation (if quantitative) or highly particularistic and informal about causal dynamics (if qualitative). Simulation analysts, because their focus is on understanding the model itself and because their technique demands it, must be exact and specific in attempting to specify causal dynamics that accomplish satisfactory translation between the two.*

In all it can be said that modeling and simulation is thus used as a formalizing tool to elaborate on theories in a manner that can enhance its theoretical precision and enabling theoretical elaboration and exploration through a computational representation.

Although there are many researchers describing the virtues of using M&S for building theory, there are not many formally described methodologies that present this process. Some of those approaches are presented in the following section.

4.1 Existing M&S Methodologies/Methods for Theory Building

According to the Webster Online dictionary the definitions of methodology and method are:

- *Methodology is a body of methods, rules, and postulates employed by a discipline.*
- *Method is a systematic procedure, technique, or mode of inquiry employed by or proper to a particular discipline or art.*

Both definitions highlight a distinction between these terms; the latter is contained within the former. This distinction is important given that in most cases they are used interchangeably. A methodology focuses on the high level philosophical perspective a researcher has to create knowledge. This perspective assists into how knowledge claims are justified under particular epistemological and ontological leanings. For instance, an empiricist departs from the premise of existence of objectively observable and measurable phenomena within reality to create knowledge. A rationalist, on the other hand, departs from the premises of a created, artificial, and incomplete reality dependent on the individual perspective. In other words, reality is a

construct an individual creates to understand reality and that depends on his/her perspective. For an empiricist reality exists and it is not dependent on the individual's perspective. Methodologically, the empiricist relies on methods that allow him/her to capture that reality in a manner that it is measurable. The rationalist, on the other hand, relies on methods that allow him/her to create an artificial world. How close the created world is to reality is "not the necessarily" the priority as long as insight is drawn. It is said "not necessarily" because researchers always look for closeness to reality. However, in many cases this is not achievable.

It is important to note that a researcher usually abides under a philosophical underpinning; the one he/she considers it the "best" way to create knowledge or do research. This "best" way varies from discipline to discipline with methodologies/methods more widely accepted than others.

M&S has both empirical and rational roots depending on the departing point. However, M&S is by nature a rational approach given that it creates an artificial world to understand reality.

There are several methodological candidates for building theory using simulation with pros and cons depending on the starting point. Methodologically, all these approaches are variations of the semiotic triangle as presented in figure 1 that attempt to capture a referent through a model and then simulate the model to gain insight into the referent. However, they have different emphases.

[24] propose a method for developing theory using simulation. Simulation's primary value is in experimentation to produce new theory. They suggest the following method:

- Research Question
- Identify simple theory (conceptual modeling)
- Chose simulation approach
- Create computational representation
- Verify computational representation
- Experiment to build novel theory
- Validate with empirical data (if available)

The *research question* should reflect the deep understanding of the literature and relates to a substantial theoretical issue. A *simple theory* is the combination of one or several theoretical ideas. The simple theory is then elaborated upon by adding constructs throughout the simulation process. Assuming that simulation is the best choice, the next step is the selection of the *simulation approach* and become a crucial step given that the chosen paradigm can hinder or facilitate the theoretical progress. Creating the *computational representation* of the theory involves the operationalization of constructs, building the algorithms that follow the theoretical logic, and the specification of assumptions. Now, to ensure that the computational representation is in accordance with the theoretical logic, the *verification* of the computational representation is needed. This is basically the internal validity of the model and the checking that the model is doing, computationally, what is supposed to be doing. *Experimentation* is suggested as the heart of the value of the simulation method for developing theory. According to [24], experimentation builds new theory by revealing fresh theoretical relationships and novel

theoretical logic and enabling experimentation across a wide range of conditions by changing the software code.

This method is consistent with the M&S premise departs from the premise that insight is gained through experimentation through simulation. However, this method departs from the assumption that there exists a simple theory without elaborating on how to attain it, how to assess its level of simplicity, or where to obtain the constructs used in the simulation to elaborate on the theory. Further, this method may be more appropriate for theory testing than for theory building given that a theory already has been formulated. It is noted that a simple theory for a researcher may be a complex theory for someone else. Finally, this approach seems to be focused more on the simulation aspect than on the modeling aspect. Although simulation is important to establish a computer experimental environment in order to generate data, modeling provides the structure and traceability to capture the phenomenon of interest. The underlying assumption is that if the simulation runs then the model was properly built and that is not necessarily the case.

[31] present a method on how to conduct "good axiomatic quantitative research" in Operations Management (OM). They propose two methods, one that is based on mathematical modeling and the other on simulation. Given that the latter is the one of interest, its suggested steps are:

- Conceptual Modeling
- Justification of proposed solution
- Experimental design
- Analysis of results
- Interpretation of results

According to [31], *conceptual modeling* departs from scientific OM literature on the topic of interest with accepted anchor articles that contain descriptions of the general characteristics in addition to recent articles containing particular information to the process or problem being studied. *Justification of the proposed solution* present evidence that a propose heuristic may perform well under either of two settings: an old problem being studied with new techniques or solutions and a new problem being studied with existing techniques or solutions. The *experimental design* contains all factors that can have an impact on the results. These factors need to be varied in the simulation over ranges of values and kept sufficiently low so simulation and further data analysis can be performed efficiently and effectively. After the simulation is performed, *analysis on the results* is needed. This can be performed using different statistical techniques which can be used depending on the research question. Lastly, interpretation of results is conducted within the context of the problem seeking to derive conclusions about the original questions and to formulate new questions.

Unlike [24], [31] considers the modeling aspect. However, it does not elaborate how conceptual modeling must be conducted considering that conceptual modeling is a broad research area [26]. In this case, it is perhaps assumed that conceptual modeling is an objective process given that in OM problems are well-defined unlike problem situations. Further, [34] suggest that only OM articles on the topic of interest are accepted. This condition discards the possibility of using articles outside of OM that can shed light on the phenomenon of interest. It is noted that in OM is widely

used the practice of building models upon existing models. For instance, a just in time model that considers earliness (a) can be used to build a just in time model that considers earliness and tardiness (b). In this case, model (a) becomes the solution that is modified to address problem (b). In this sense, these problems are of interest to the OM community only.

[32] present a methodology and a method called Rationalist Inductive. It is rationalist given that it builds on a system of premises and not on data. It is inductive given that its focus is on theory building and not theory testing. The methodology (fig. 2) has three components: Exploration and selection, rationalist structuration, and conclusion. The *exploration and selection* stage seeks relevant theories that help answer the research question and provide the axiomatic foundation of the research. This includes the initial establishment of premises, axioms, and context. Rationalist *structuration* takes the theories from the previous stage and produces a coherent structure needed by bounding and fine tuning the system of beliefs and premises. Theories are put together by the means of a mathematical, logical, or computational model. Finally, *conclusion* is the materialization of the rational inductive process in a form of a pattern or patterns resulting from the model that explain the theory. In terms of a method, [32] provide a way of mapping the induction-based methodology and the research process using modeling and simulation techniques. The method (fig. 2) starts with the identification of a problem, premises, and context out of the body of knowledge; continues with the definition of premise and context; then follows with the simulation aspects of selection, execution, and testing; and ends with interpretation of results and conclusions.

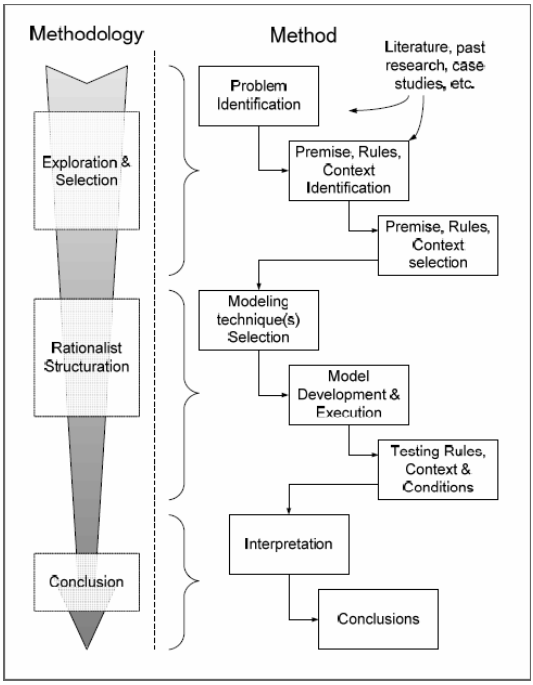


Fig. 2. Rationalist/Inductive Methodology and Method [32]

The method consists in identifying premises from the body of knowledge and put them together in a coherent system where assumptions are made explicit and no contradictions are created. Coherence is then established via modeling and simulation and from the results of the simulation an interpretation is conducted. This approach is based on the traceability of the resulting theory to the set of premises and the set of premises to the body of knowledge as a form of validation of the theory. However, the method, as [24], focuses more on the simulation as a way to establish coherence and generate data for drawing insight. Further, it does not establish criteria of selection of premises and which to include in the simulation.

It is noted that in all three cases modeling takes a supporting role to the simulation effort. In other words, modeling is important as long as the simulation is the one providing the insight. However, the modeling process can also create theory that can be enriched with results from the simulation process. In addition, these approaches do not explicit provide mechanism for studying ill-defined problems. The proposed approach covers the development of theories for this kind of problems.

4.2 A Methodology for Theory Building Using M&S

The purpose of the chapter is not to provide a new method, but to exemplify how a general methodology to building theory using M&S can be used. Further, it shows how a method can be derived from such methodology that can be used when the problem is ill-defined. The methodology used, similar to [24], [31], and [32], is rational in the sense that it creates a world to study a phenomenon in reality. In addition, like [24], [31], and [32], it considers simulation an integral part, but also focuses on the modeling process and on the correspondence of the model and its simulation. Unlike [24], [31] and [32], the methodology provides criteria for developing the conceptual model, especially on how premises are identified and put together in a system of premises. It is noted that, unlike [31], it is not bound to the use of articles from one discipline only. It draws from any discipline that provides insight into the phenomenon at hand. Finally, unlike [24] it does not depart from a simple theory or from existing models as it may be the case in [31]; it departs from many theories towards building a new theory.

The methodology consists of three parts: A *research question*, *capturing reality* in a model through the identification of common threads and assumption in related literature, and implementing the model in a *simulation* that better answer the research question. It can be observed that this methodology is rational given that its focus is on creating a reality to deal with a phenomenon and not on observing reality to deal with the phenomenon in question.

The *research question* provides the seed of what the researcher wants to study and it relates to a problem or phenomenon of interest. In engineering, for instance, it is required that the problem is justified answering questions like why is it a problem? Or what do we gain with solving such a problem? In this sense, the research question must consider the problem at hand and why its solution is important to the body of knowledge.

Capturing reality in a model is needed to provide an axiomatic structure upon which rigor can be established. To "capture" reality and create an artificial world, theories from different areas that address the problem at hand are gathered, analyzed, and most importantly, explain why some are included and others are discarded. Possible reasons

for this filtering are: administrative, to reduce the number of working theories to a manageable size; rigor, pruning inconsistent, ambiguous, and contradictory arguments within a theory; commonalities, identify what are the common thematic threads and assumptions across all participating theories. The premise behind creating a model that eventually can be implemented in a simulation is that the model be formal and for that it must be precise so the resulting theory can be formal and precise. The identified common thematic threads are used to create constructs and relations among constructs. Assumptions are used as sub-problems that need to be addressed or solved. In addition, they serve as a pragmatic way to verify the resulting theory as the challenged assumption was properly addressed with the resulting theory. In the process of putting constructs together, as system of premises as highlighted by [32] is being built. It is important to note that this system of premises is built with "filtered' theories and not with theories as they are. This is because theories-as-they-are present contradictions and inconsistencies that must be identified before putting the system together so coherence can be established. Conflicting elements usually reside in unchallenged assumptions or poor definitions. It is noted that the modeling process itself may provide theoretical insight by building a structure and considering combinations that were not possible until constructs were defined.

As in [24], [31], and [32], the *simulation* is a computer implementation of the model with the purpose of further establishing formality (computer program is a function that may or may not be written in mathematical terms), provide an experimentation setting, and generating data. Data can be qualitatively and quantitatively assessed to gain insight about the phenomenon of interest.

One important aspect to consider is the correspondence between modeling and simulation. In addition to the possibility of many models explaining a system or phenomenon of interest, there may be many possible simulations of one model. Although a researcher may not control the different perspectives captured in different models, he/she may control the simulation of his/her model by constraining it to the simulation that best answers the research question. For instance, a model that shows how a capability can be improved does not explain what the capability is and vice versa. If the research question is what interoperability is, it is important to focus on a model and a corresponding simulation that focus on establishing a baseline of what interoperability is. On the other hand, if the question is how to improve interoperability, the focus is not on defining interoperability, but on how to improve it under certain conditions.

4.3 Selecting the Modeling Paradigm

The selection of the most appropriate paradigm of M&S to answer the research question is an important aspect of the theory building process. A simple heuristic that can assist the researcher in this effort is based on the premises generated in the modeling process. A basic guide is provided by answering the following questions:

1. Can causality be established that describe the relation among constructs of an identified dominant structure?
2. Is there an identifiable sequence of events?
3. Are there premises that can be used as rules of behavior?

If the answer to question 1 is yes, then System Dynamics is the most appropriate paradigm given that equations can be established through causally among constructs and within the identified dominant structure. If yes to question 2, then Discrete Event (DE) Simulation is the most appropriate paradigm given that sequence of events describing entity flow are the bases for DE. If yes to question 3, then ABM is the most appropriate paradigm given that constructs can be seen as agents and premises as underlying rules of behavior. For a more elaborated explanation on paradigm selection in systems engineering, the reader can refer to [33] for further information. For the test case, SD was not selected because causality among constructs could not be established within a dominant structure nor the dominant structure be identified. DE was not selected because a sequence of events was not identified among the defined constructs. Finally, ABM was selected because constructs and relations among constructs could be defined as agents and rules of behavior of agents.

5 Test Case: Building a Theory of Understanding Using Agents

5.1 Brief on ABM and Its Relevance on Theory Building

According to [25], p. 172 "although there is no generally agreed definition of what an 'agent' is, the term is usually used to describe self-contained programs that can control their own actions based on their perceptions of their operating environment." [34] define an agent as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators." Perception in this context "refer to the agent's perceptual inputs in a given instant" and the term percept sequence as "the complete history of everything the agent has perceived." [35] propose that an agent should perceive its environment, and act in its environment. Further, an agent should communicate with other agents to establish a social ability. Moreover, an agent should be autonomous, outside of central control, and flexible, being able to react to, pursue goals, or adapt to changes in its environment.

[36] present three main contributions of ABM to the advancement of theory:

- Explicitizing: The ABM environment demands an exacting level of clarity and specificity in expressing a theoretical model and provides the tools, structures and standard practices to achieve this high level.
- Emergence: The computational power of ABM enables the researcher to mobilize an otherwise static list of conjectured behaviors and witness any group-level patterns that may enfold through multiple interactions between the agents who implement these conjectured behaviors.
- Intra/interdisciplinary collaboration: The lingua franca of ABM enables researchers who otherwise use different frameworks terminology and methodology to understand and critique each others' theory and even challenge or improve the theory by modifying/and or extending the computational procedures that underlie the model.

Explicitazing is important given that demands to state assumptions and presuppositions about the model and about the system or theory being modeled. In addition, it provides a level of formalization and precision that would not be achieved if the theory is expressed in natural language [23]. Emergence occurs when interaction among parts of a system or objects in a model at a lower level give rise to different types of parts or objects at a higher level [25]. This interaction among objects translates to interaction among agents making emergence a characteristic widely associated with this modeling paradigm given the behavior arising from the interaction of individual agents that was not a behavior programmed directly into the system. Finally Intra/interdisciplinary collaboration allows for researchers across disciplines to work with one another towards solving common problems which is of particular importance for SE.

5.2 Importance of Understanding to Problem Situations

As previously mentioned, a problem situation is where there is no agreement on what the problem is or even that there is a problem. Problem situations are ever more pervasive in SE efforts challenging the objectivity assumption that is centerpiece in the discipline. Given that objectivity cannot or should not be assumed, a measure of complexity is no solely on the number of parts and relations among parts, but also on how a person understands a problem differently than another. Understanding is a common thread found from topics ranging from complexity to decision making and learning, especially when referring to human-driven complexity. [4], p. 24 state that "in general, we associate complexity with anything we find difficult to understand." [37] p. 131 concurs with this assessment and states that "in addition to the common sense characterization of the degree of complexity as the number of interrelated parts, it also has a somewhat subjective connotation since it is related to the ability to understand or cope with the thing under consideration." This dependence on the individual to seeing problems as complex extends to systems engineering where formulations and decisions are made by a group of stakeholders.

In terms of learning and decision making, [38] remarks that we use learning to revise our understanding of the world and in so doing we affect the decisions we make. In terms of problem solving, [39], p. 308 note that "the definition of the initial state would reflect the individuals' understanding of the nature of the problem at the beginning, and the desired end-state would be described as the goal expected to be achieved by solving the problem."

Although it can be seen that understanding is an important concept, first it is not properly defined and second it has not been studied to assess how it can affect concepts of interest to systems engineers such as complexity.

5.3 Implementing the Methodology for Theory Building Using M&S

As previously mentioned, different methods may abide by a particular methodology. The methodology presented, unlike [24], [31], and [32], provides detail in the modeling process, in the simulation process, and their correspondence depending on the research question. Figure 3 shows one method that abides by the presented methodology and shows how it is implemented in the topic of understanding [7].

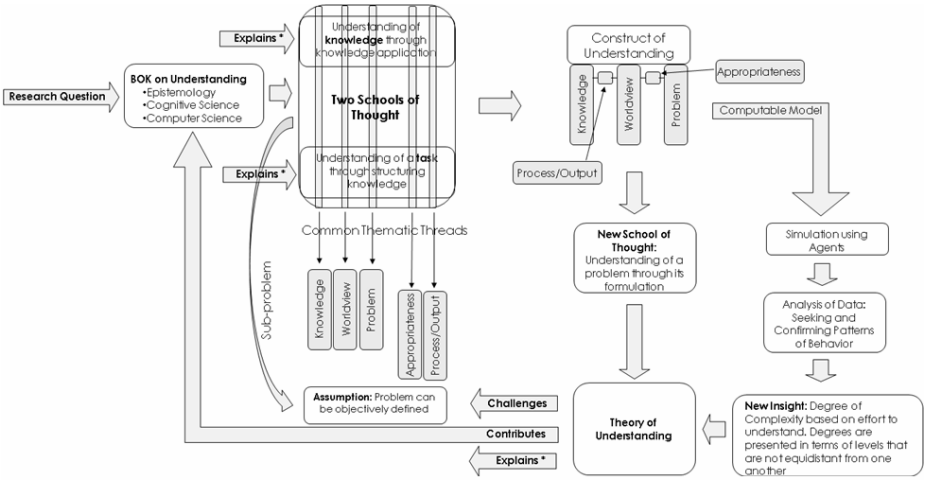


Fig. 3. Implementing the Methodology to build a Theory of Understanding (Adapted from [7])

The formulation of the research question is first. In this case the question is: *what is understanding?* As it was presented in section 2 and section 5.2, the definition of this concept is of great importance towards better understanding human-driven complexity. Following the formulation of the research question and problem formulation, a literature review on the Body of Knowledge (BOK) of the concept of understanding is conducted. Three areas in the body of knowledge have studied understanding, namely epistemology, cognitive science, and computer science. From these three areas, *two schools of thought* were identified. The schools of thought have *common thematic threads* and *assumptions*. Common threads correspond to the concepts of knowledge (*K*), worldview (*W*), and problem (*P*). In addition, conditions such as appropriateness and that understanding can be seen as a process and output is also found. The main assumption found was the problem can be objectively defined (*Objectivity Assumption*). This assumption subsequently allows understanding to be evaluated through the matching of a solution to the problem. This assumption does not stand, as mentioned, when dealing with problem situations. This assumption becomes a sub-problem that needs to be addressed. The common threads become constructs that are used to create the axiomatic structure in the form of a meta-construct (*Construct of Understanding*). Unlike knowledge, worldview, and problem, appropriateness and process/output do not become constructs but conditions that regulate the interaction of constructs and a way to understanding respectively. Finally, *K*, *W*, and *P* were divided up in two constructs each (six constructs total) based on the descriptions on respective descriptions found in the literature. To further study the six constructs an expanded literature review was required. The six constructs in conjunction with the appropriateness condition provide the basic derived premises from the BOK that later can be implemented as a rule base in a computer program.

It is noted that the two identified schools of thought can be explained with the built meta-construct therefore establishing that it does not contradict the existing theory. In addition, a new school of thought was identified. This school of thought addresses the sub-problem posited when challenging the objectivity assumption by considering different formulations of the problem. This new school of thought becomes a new theoretical insight derived from the modeling process.

The built model is then simulated. This simulation serves as the ultimate way of evaluating the rigor of the model by making it run in a computer and verify that results are consistent with the encoded premises. In other words, verifying that the match between model and simulation is established by seeking answers to the research question. In this moment, the model becomes the reality upon which the researcher performs experiments to collect data to draw insight about the phenomenon. It is important to note that this simulation sought to establish a baseline of what the process of understanding was like and not to seek how to 'better understand'. The work in the BOK focuses repeatedly on better understanding despite understanding not being properly defined. In addition, the use of 'better understanding' is convoluted with processes such as learning and problem solving.

To conduct experiments, a design of experiments that would consider all possible combinations among constructs was established. The six constructs in addition with a time constraint (also based on the literature) became seven (7) variables or factors for consideration. Considering at least two levels for each factor (reflecting high or low levels of a particular knowledge, worldview, or problem type) was calculated that 128 experiments were needed (2^7). A pilot run of ten (10) experiments was performed to calculate the number of runs necessary to attain a 95% confidence interval with a 10% margin of error. It was determined that 250 runs were needed. Given that there were three schools of thoughts and each needed to be studied separately (no theory was found on how they would intertwine), three (3) different run scenarios were established. Considering scenarios, runs, and initial conditions, it was determined that 96000 experiments were needed.

To implement the derived premises and to conduct the experiments ABM was used.

Each one of the six factors became an individual agent that interacts with the others based on the appropriateness derived condition. *K* agents would interact with *W* and *P* agents, *W* with *K* and *P*, and *P* with *K* and *W*. No interaction among *K*, among *W*, and among *P* was defined given that no theory, corresponding to understanding, would establish how those interactions would take place. It is noted that high or level of *K*, *W*, or *P* are represented by a high or low number of agents and these numbers ultimately affect their interaction. In addition, to be able to establish the baseline, some restrictions were imposed:

- As presented in the proposed approach, no predisposed idea of the phenomenon by the researcher was considered. Every premised needed to be based on the BOK.
- All forms of movement and interaction were defined as random.
- No memory and sequencing of events were introduced.

These restrictions were necessary to eliminate any imposition on the way the simulation should behave in the process of looking for emergence based on simple rules of interaction.

Finally, two output variables were considered: one called effort which reflected the interaction between *K*, *W*, and *P* agents and the second called time. Effort, or the failed interaction between *K*, *W*, and *P*, was based on the proposed definition of understanding and was stored in a counter. Time was a counter that stored the length a problem would take to be understood. The 96000 experiments would account for both variables and identify them for each initial condition of the 128 and for each scenario.

To capture the model of the phenomenon of understanding, unified modeling language (UML) was used. Although UML diagrams captured most of the part components and behavior of the model, shortcomings were found due to the modeling paradigm. For instance, the sequence diagram presents sequences between entities. On an agent representation, agents run in parallel instead on a predefined sequence. Nonetheless, the diagram was useful to capture the basic interaction among the three types of agents (*K*, *W*, and *P*) as it can be seen in figure 4.

Figure 4 shows how *K*, *W*, and *P* agents relate to one another depending on the school of thought in consideration. For instance, for the school of thought named KP then *W*, the match starts with *K* and *P* and locking onto one another for a period of time. In this period of time (definable by the user), *K* and *P* wait for *W* to finalize the match and update the counter. The diagram does not capture when *K* and *P* separates after the user defined time frame runs out. This separation is needed to avoid the effect of memory in the matching of *K*, *W*, and *P*. The separation can be captured by an activity diagram.

A complementary form of capturing the conceptual model towards an agent implementation is with propositional logic (Russel and Norvig, 2003). In this case, propositional logic captures the behavior of agents in the form of logical connections between clauses. For instance, $A_i \wedge B_i \wedge C_i$ where A = *K* in patch, B = *W* in patch, and C = *P* in patch, *i* = type of *K*, *W*, or *P* (Boolean). When *i* has the same value, understanding occurs, otherwise, the person does not understand.

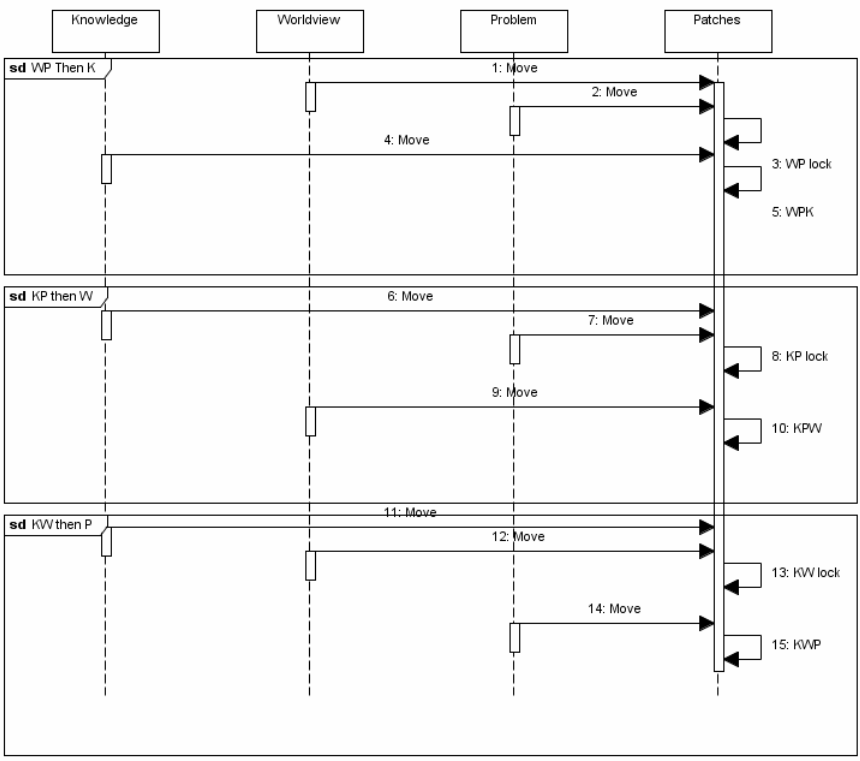


Fig. 4. Sequence Diagram for the Model of Understanding [7]

Finally, the model was implemented in an application called Netlogo. The interface allows for establishing the desired initial conditions that reflect the different experiments, the school of thought, and the time frame. The output, although can be observed on the interface per run, it is capture (for all runs) in a file that can be exported to a spreadsheet and analyzed with a package that facilitates statistical analysis. In this particular case, SPSS was used to conduct statistical analysis ranging from normality tests to parametric and non-parametric comparison of means.

The analysis of data was initiated by a qualitative analysis; observing whether there was a discernable pattern in the data. Time did not provide a discernable pattern, however, effort did. This pattern was used to conduct the statistical analysis of data that was mostly based on the comparison of means for each initial condition within and across schools of thought. The main finding was that initial conditions clustered on seven levels. In level 7 the individual had extreme difficulty in understanding the problem compared to difficulty in level 6. Level 1, 2, 3, and 4 were 'close' in terms of effort compared to level 5, 6, and 7. Levels 5 and 6 were relatively close compared to level 7. This separation of effort in terms of levels provided the insight on degree of complexity based on understanding, or lack of it, especially when considering the different distances between levels. This implies that for a person is better to be in level 6 than in level 7 while being at level 4 and 3 makes not a great difference.

The insights drawn from the three schools of thought and from the analysis of data allows the formulation of a theory of understanding that: defines understanding, what understanding is, what it does, and how it does it and provides insight into concepts such as complexity which are of importance to system engineers.

In the methodological aspect, it is highlighted that the resulting theory contributes to the BOK and explains existing school of thoughts while challenging existing assumptions.

6 Final Remarks and Conclusion

Theory creation is, in many cases, an overlooked aspect of the research endeavor. The over emphasis and imposition, in some cases, of empiricism limit the production of new theories and trade them for their testing. Paul Feyerabend (1924-1994), a well-known contemporary philosopher, posited that many of the major human scientific achievements of mankind do not come through empirical means. He suggests that any method is the method and researchers, in their investigative processes, should not constraint to one method in particular. As such, new methods that advance the creation of new knowledge should be welcomed as long as they facilitate the researcher's goal while providing rigor and consistent outputs. In other words, methodologies and methods should be selected on the basis of their appropriateness to the problem at hand and research question and not by the religious imposition of one particular research mentality.

This chapter presents a methodology and method that guide the researcher in building theory out of existing theory with the assistance of Modeling and Simulation. It is shown that this approach is of particular importance when studying ill-defined problems. To support this argument, it is presented that theory can be built through empirical or rational means. In the latter case, the researcher is likely to have data, means of

measurement, and access to an objectively defined problem. In the former, limited to no access to data, no means of measurement, and lack of consensus about the problem are predominant. To overcome the limitations in the former case, researchers may rely on speculation, argumentation, or thought processes to develop theories. It is presented that M&S can be used effectively and efficiently as a conduit to establishing rigorous research to theoretical endeavors. This rigor is established in the form of traceability, structure, and precision when modeling and through a computable implementation of a model. It is suggested that this approach can be used to develop new theories in disciplines such as systems engineering. Systems engineering is a compelling case of a discipline where problems are becoming more human-oriented. These new types of problems are not currently addressed by the technology-focus of the discipline. New theories have surfaced to address these human-oriented issues under the umbrella of new disciplines, even though these problems still affect systems engineers. The majority of these theories are based on anecdotal experience or speculation on how these issues can be resolved. Although widely used, these theories do not or insufficiently cover many different issues such as the human-driven complexity of a problem. As a way to highlight the methodology, a test case is presented. This case showcases how through the proposed methodology a theory of understanding was built. This theory not only provides insight into the phenomenon of understanding but also insight into the human-driven complexity of a problem. Agent-based Modeling was used to implement the model captured from the literature in a computable form and generate data that further generated insight.

7 List of Acronyms

ABM: Agent-based Modeling
BOK: Body of Knowledge
DE: Discrete Event
K: Knowledge
M&S: Modeling and Simulation
P: Problem
SD: System Dynamics
SE: Systems Engineering
SoSE: System of Systems Engineering
SSM: Soft Systems Methodology
UML: Unified Modeling Language
VMASC: Virginia Modeling, Analysis and Simulation Center
W: Worldview

References

1. Checkland, P., Scholes, J.: Soft Systems Methodology. Wiley, New York (1990)
2. Keating, C.B., Unal, R., Sousa-Poza, A., Dryer, D., Rogers, R., Safford, R., Peterson, W., Rabadi, G.: System of Systems Engineering. Engineering Management Journal 15(3), 35–44 (2003)
3. Vennix, J.: Group Modeling Building. Wiley, New York (1996)

4. Flood, R., Carson, E.: Dealing with Complexity: An Introduction to the Theory and Application of Systems Science. Plenum Press, New York
5. Keating, C., Padilla, J.J., Adams, K.: System of Systems Engineering Requirements: Challenges and Guidelines. Engineering Management Journal 20(4), 24–31 (2008)
6. Keating, C.: Paradoxes in the Engineering of Complex System of Systems. In: Proceedings from the 29th American Society for Engineering Management National Conference, West Point, NY, November 12-15 (2008)
7. Padilla, J.J.: Towards a Theory of Understanding within Problem Situations. Doctoral Thesis at Old Dominion University, Norfolk, VA (2010)
8. Hubler, A.: Predicting Complex Systems with a Holistic Approach. Complexity 10(3), 11–16 (2005)
9. Freese, L.: Formal Theorizing. Annual Review of Sociology 6, 187–212 (1980)
10. Allin, B.: Theory: Definition and Purpose. Journal of Farm Economics 31(3), 409–417 (1949)
11. Whetten, D.: What Constitutes a Theoretical Contribution? Academy of Management Review 14(4), 490–495 (1989)
12. Hearn, G.: Theory Building in Social Work. University of Toronto Press, Canada (1958)
13. Weick, K.: Theory Construction as Discipline Imagination. Academy Management Review 14(4), 516–531 (1989)
14. Hempbel, C.: On the Logical Positivists' Theory of Truth. Analysis 2(4), 49–59 (1935)
15. Keuth, H.: Tarki's Definition of Truth and the Correspondence Theory of Truth. Philosophy of Science 45, 420–430 (1978)
16. Firth, R.: Coherence, Certainty, and Epistemic Priority. The Journal of Philosophy 61(19), 545–557 (1964)
17. Walker, R.C.S.: Spinoza and the Coherence Theory of Truth. Mind 94(373), 1–18 (1985)
18. Ackoff, R.: Ackoff's best: His Classic Writings on Management. John Wiley and Sons, New York (1999)
19. Ashby, W.R.: An Introduction to Cybernetics. Methuen and Co., New York (1984)
20. Beer, S.: The Heart of Enterprise. John Wiley and Sons, New York (1979)
21. Davis, P., Anderson, R.: Improving the Composability of Department of Defense Models and Simulations. RAND Corporation, Santa Monica (2003)
22. Diallo, S., Tolk, A., Weisel, E.: Simulation Formalisms: Review and Comparison of Existing Definitions of Key Terms. In: Fall Simulation Interoperability Workshop. 07F-SIW-061 (2007)
23. Gilbert, N.: Agent-based models. SAGE Publications, Thousand Oaks (2008)
24. Davis, J., Eisenhardt, K., Bingham, C.: Developing Theory through Simulation Methods. Academy of Management Review 32(2), 480–499 (2007)
25. Gilbert, N., Troitzsch, K.: Simulation for the Social Scientist. Open University Press, New York (2005)
26. Tolk, A., Diallo, S., King, R., Turnitsa, C., Padilla, J.J.: Conceptual Modeling for Composition of Model-Based Complex Systems. In: Robinson, S., Brooks, R., Kotiadis, K., Van Der Zee, D. (eds.) Conceptual Modeling for Discrete-Event Simulation, pp. 355–381. CRC Press, Boca Raton (2010)
27. Gilbert, N.: Models, Processes and Algorithms: Towards a Simulation Toolkit. In: Suleiman, R., Troitzsch, K., Gilbert, N. (eds.) Tools and Techniques for Social Science, pp. 3–16. Physica-Verlag, Heidelberg (2000)
28. Muller, G.: Computer-Assisted Interfacing: On the Use of Computer Simulation for Theory Construction. In: Suleiman, R., Troitzsch, K., Gilbert, N. (eds.) Tools and Techniques for Social Science, pp. 26–47. Physica-Verlag, Heidelberg (2000)

29. Leik, R., Meeker, B.: Computer Simulation for Exploring Theories: Models of Interpersonal Cooperation and Competition. Sociological Perspectives 38(4), 463–482 (1995)
30. Hanneman, R.: Simulation Modeling and Theoretical Analysis in Sociology. Sociological Perspectives 38(4), 457–462 (1995)
31. Bertrand, J.W.M., Fransoo, J.C.: Operations Management Research Methodologies Using Quantitative Modeling. International Journal of Operations & Production Management 22(2), 241–264 (2002)
32. Sousa-Poza, A., Padilla, J.J., Bozkurt, I.: Implications of a Rationalist Inductive Approach in System of Systems Engineering Research. In: Proceedings of IEEE International Conference on System of Systems Engineering, Systems, Man, and Cybernetics (2008), doi:10.1109/SYSOSE.2008.4724186
33. Hester, P., Tolk, A.: Applying Methods of the M&S Spectrum for Complex Systems Engineering. In: Proceedings from the Spring Simulation Conference, Orlando, Fl, April 11-15 (2010)
34. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River (2003)
35. Tolk, A., Uhrmacher, A.: Agents: Agenthood, Agent Architectures, and Agent Taxonomies. In: Yilmaz, L., Ören, T. (eds.) Agent-Directed Simulation & Systems Engineering, pp. 87–126. John Wiley & Sons, New York (2009)
36. Abrahamson, D., Wilensky, U.: Piaget? Vygotsky? I'm game!: Agent-based Modeling for Psychology Research. Paper presented at the annual meeting of the Jean Piaget Society, Vancouver, Canada (2005)
37. Klir, G.: Complexity: Some General Observations. Systems Research 2(2), 131–140 (1985)
38. Sterman, J.: Learning in and About Complex Systems. System Dynamics Review 10(2-3), 291–330 (1994)
39. Nair, K.U., Ramnarayan, S.: Individual Differences in Need for Cognition and Complex Problem Solving. Journal of Research in Personality 34, 305–328 (2000)

Chapter 11
"*The User Around the Marketplace*": Automatic Engineering of Interactive E-commerce Applications

Martín López-Nores, Yolanda Blanco-Fernández, and José J. Pazos-Arias*

Department of Telematics Engineering, University of Vigo,
ETSE Telecomunicación, Campus Universitario s/n, 36310 Vigo, Spain
{mlnores,yolanda,jose}@det.uvigo.es
http://idtv.det.uvigo.es

Abstract. When thinking about opportunities for e-commerce, it is necessary to differentiate scenarios in which the user's aims and attention are close to the items that may be offered to him (e.g. when he has just entered an online bookshop) from scenarios in which they are not (e.g. in delivering publicity while he watches TV). The support available for the former is quite advanced nowadays, especially thanks to the development of recommender systems. On the contrary, in cases in which the user gets to know about interesting items casually, there is much place to provide him with better guidance and thereby increase the number of transactions accomplished online. The key aspects have to do with contextualizing the selection of the items that will be offered to the users, delivering the information they may find more valuable and understandable in the most accessible way, and identifying the most convenient providers taking into account not only price and availability but also the users' location. We describe a system that furnishes these features, relying on semantic and rule-based reasoning techniques to automatically compose interactive services that let the users browse details of (and purchase online) various kinds of items they might be interested in, as inferred from knowledge about their individual preferences, interests and needs. A prototype targeted at improving Digital TV advertising is described.

1 Introduction

The evolution of e-commerce technologies in the last few years has been dominated by the consolidation of *recommender systems*, which apply a range of artificial intelligence techniques to proactively discover the items that best match the preferences, interests and needs of each individual at any time. There are recommenders working behind the scenes in many e-commerce sites [1,2], where

* Work supported by the Ministerio de Ciencia e Innovación (Gobierno de España) research project TIN2010-20797, and by the Consellería de Educación e Ordenación Universitaria (Xunta de Galicia) incentives file 2007/000016-0.

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 285–307.
springerlink.com

it makes perfect sense to face the visitors with lists of items they may appreciate, as inferred from records of their previous purchases. This is because the user's aims and attention are sufficiently close to the items that may be recommended to him: e.g. if he has just entered an online bookshop, it is most likely that he will be looking for a new read. In such contexts, which we classify under the metaphor "*the user inside the shop*", the actions triggered by the user selecting one item from the list are straightforward, so the recommender system has completed its task. In contrast, whenever the user is not focused on the kind of items that may be recommended to him (e.g. when watching a movie on TV), the opportunities for e-commerce should arise from non-invasive, casual discovery of potentially interesting stuff, as much related as possible to whatever his interests are at the moment (e.g. the topic of the aforementioned movie). In fact, one would expect the recommender to go further that assembling a list of items, for example, to select the most convenient offers from among various providers, to look for the pieces of information that describe each item in the most complete or accessible way for the user in question, to arrange the most suitable interfaces to place an order through whichever device, etc. In other words, with "*the user around the marketplace*", we believe a recommender can be put in charge of *building the shop* in which the user will feel most comfortable to browse potentially interesting items.

The discussion above puts forward a concept of online shop that differs from the traditional one. Nowadays, it is almost straightforward for any provider to build a web site to browse the items of a catalogue, to manage shopping carts, to process payments, and so on. However, those sites deliver information in a *one-size-fits-all* manner, i.e. all the users are always faced with the same information, which may contain uninteresting data or be difficult to understand at all. Obviously, it should not be a task for human developers to create suitable sites or applications for all the different users, considering also the multiplicity of items, access devices and contexts —simply because no workforce would suffice. To address this problem, we describe in this chapter a system that uses semantic and rule-based reasoning techniques to automatically engineer interactive commercial applications, personalized according to information about the individual preferences, interests and needs of the users. As the distinctive features of the proposal, it is possible to contextualize the selection of the items that will be offered, to personalize the information delivered and to link the most convenient providers, taking into account not only price and availability but also the users' location. Since the decision of which providers to link is not made beforehand, we are promoting an open environment that lowers the barriers for providers to trade over the Internet and that ensures visibility, just like the traditional search engines (e.g. Google or Bing) have been doing for decades as a prelude to "*the user inside the shop*" experiences.

The chapter is organized as follows. First, in Section 2, we provide an overview of the state-of-the-art in the areas of recommender systems, web services and web mashups, which are directly related to our proposal. Afterwards, Section 3 describes the elements that come into play during the composition of interactive

commercial applications in our system, whereas Section 4 explains the procedural questions. Section 5 shows how this proposal has been implemented in the context of DTV advertising, also including the results of preliminary experiments aimed at assessing personalization quality and users' satisfaction. Finally, Section 6 provides a brief summary of conclusions.

2 Background

As regards the classical goal of identifying the most suitable items for a given user, recommender systems work by matching the information in a *user profile* against metadata descriptions of the items available. The first possibility explored to do so was *content-based filtering*, which consists in making recommendations by looking at contents that gained the target user's interest in the past [3, 4]. This strategy is easy to adopt, but bears a problem of *overspecialization*: the recommendations tend to be repetitive for considering that one will always appreciate the same kind of stuff. To tackle this problem, the scientific community came up with *user-based collaborative filtering*, which proceeds by evaluating not only the profile of the target user, but also those of users with similar interests (his/her *neighbors*) [5, 6]. This approach solved the lack of diversity in the recommendations, but faces problems like poor performance with people whose preferences are dissimilar to the majority (the *gray sheep*). An alternative was *item-based collaborative filtering*, which recommends items related to others that the target user liked in the past, considering two items related when users who like the one tend to like the other as well [7]. However, this approach faces problem like *sparsity*: when the number of items available to recommend is high, it is difficult to find users with similar valuations for common subsets. Not surprisingly, there exist *hybrid approaches* that attempt to neutralize the weaknesses and combine the strengths of content-based and the different variants of collaborative filtering [8, 9].

Regardless of the filtering strategy, the first recommender systems used *heuristics* or *syntactic* matching techniques, which relate items by looking for common words in their attached metadata. These techniques miss a lot of knowledge during the personalization process, because they are unable to reason about the meaning of the metadata (for example, it is not possible to link items about "*Golden Retriever*" with items about "*Boxer*", as the two words are dissimilar). Besides, they are a source of overspecialization, because the recommendations so computed can only include items very similar to those the users already know. To go one step beyond in personalization quality and diversity, research is now focused on applying techniques from the *Semantic Web*, which enable reasoning processes that gain insight into the meaning of words (so that, for example, "*Golden Retriever*" and "*Boxer*" can be automatically recognized as two breeds of dogs, the latter having nothing to do with a combat sport). The key here lies within the use of *ontologies* [10] to describe and interrelate items and their attributes by means of class hierarchies and labeled properties (see Figure 1).

The most popular language to build ontologies nowadays is OWL (*Web Ontology Language*) [11].

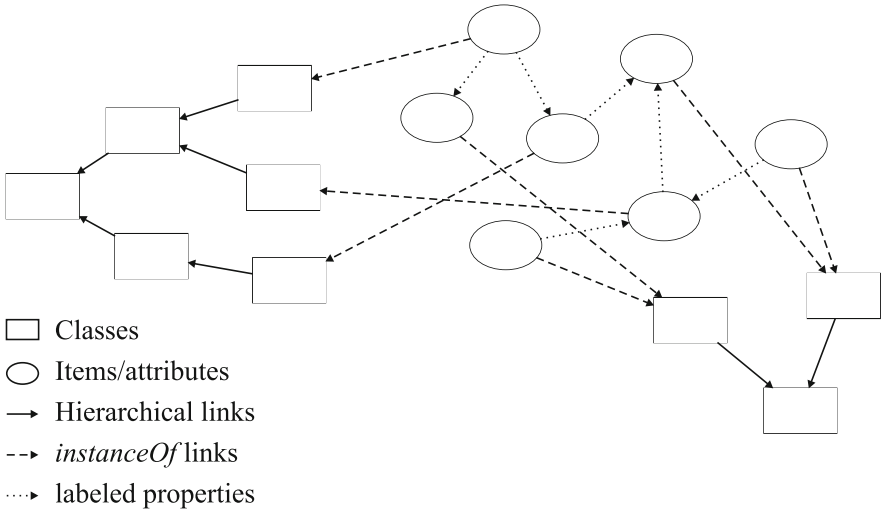


Fig. 1. The structure of an ontology

One promising line of research in personalization is that of *context awareness*, aimed at acquiring information about physical and social situations to maximize the value of the information delivered to the users. The knowledge available about context can be added to that in the user's profile to drive the selection of contents and services. Regarding formats and length, it is possible to match the time the user will have to read or watch material, the size of the screens where it will be presented, the input mechanisms available, etc. As for the semantics, the goal is basically to identify the topics the user may welcome at a given moment: news, entertainment, medical advice, etc. Historically, the first possibility explored in context awareness was to develop location-sensitive mobile applications that would display different pieces of information following the users' moves in indoor environments (e.g. museums) or outdoors (e.g. in guided city tours) [12,13]. Other dimensions were progressively added, like informational context (e.g. inferred from words or pictures on screen), infrastructure (e.g. input devices available or surrounding communication resources) and physical conditions (noise, light, etc). The recent works on *affective computing* [14], coupled with theories on emotion and cognition, make it possible to take the user's feelings (mood, stress, etc) into consideration as well.

In all of the aforementioned works, recommender systems limited themselves to providing the users with lists of the items that best match the information stored in their profiles, so there is practically nothing in literature about generating personalized interactive applications in an automated fashion. A solution to this problem can be inspired by the extensive literature of *web services*

composition, that builds upon standards like WSDL (*Web Services Description Language*) or OWL-S (*OWL-Services*) to characterize functional elements (the web services) in terms of inputs and outputs, some conceptualization of their internal processing or purpose, etc. Specifically, OWL-S involves three interrelated subontologies: *Profile* ("*what the service does*" for purposes of discovery), *Process Model* ("*how the service works*" for invocation and composition) and *Grounding* (a realization of the process model into detailed specifications of message formats and protocols). Leaning on this standard, there exist various composition engines that can automatically provide the back-end logic of many e-commerce services on the Internet [15, 16, 17, 18], in some cases considering user preferences about non-functional aspects like quality of service, payment methods, security or privacy [19]. In contrast, as regards the front-end of the applications, there have been few attempts to automate the generation of user interfaces from web service descriptions [20, 21], and there remains an open issue in the selection of the pieces of information to display within those interfaces, taking into account user preferences about content sources, topic, language, accessibility and so on. These aspects have been recently addressed in the area of web mashups, which are value-added web applications developed by integrating heterogeneous elements from various web sources, including RSS/Atom feeds, web services, content scraped from third-party web sites, or portable software components commonly called *widgets* [22]. The automatic generation of personalized mashups was addressed in [23], relying a number of ontology-based models to describe users of web portals and their potential information needs, and then applying rule-based reasoning to bring multiple resources together.

Notwithstanding the aforementioned advances, the support available nowadays for the paradigm of "*the user around the marketplace*" is still insufficient in two main aspects. The first one relates to the quality of the automatic generation processes, because the resulting applications may fail to make a cohesive whole out of pieces of information retrieved from diverse sources [24]. The second aspect has to do with the format of the applications, since there are plenty of consumer devices out there which offer specific development platforms instead of general engines to properly run web apps. This is important because of the current global trends of the Internet: as noted in [25], after many years in which more than 90% of the traffic corresponded to HTML data delivered via the HTTP or HTTPS protocols (i.e. to web browsing), the percentage has now shrunk to less than 25%; the rest has been taken up by non-web applications like peer-to-peer file transfers, e-mail, virtual private networks, the machine-to-machine communications of APIs, VoIP calls, online games, movie streaming, etc. Therefore, it is increasingly necessary to develop means to automatically generate platform-specific applications. The aim of our work is to do so in the realm of Digital TV technologies. This is a domain where the deficient support for "*the user around the marketplace*" is especially apparent, because most of the stakeholders' expectations by the mid 1990's are yet to be realized. Initially, everyone envisaged lucrative opportunities for entertainment and e-commerce, due to the possibility of delivering interactive applications jointly with the audiovisual

contents [26]. Standards like MHP (*Multimedia Home Platform*) and various supporting tools have made things easier and easier for development and deployment, but interactive DTV applications are still the exception rather than the rule. Yet, there exist a few systems that aim to personalize electronic programming guides and TV advertising, again by means of simple heuristics [27], syntactic matching techniques [28,29] or semantic reasoning [30]. Unfortunately, none of those systems has generalized the possibility of linking the programs or the advertisements with interactive applications —no matter how simple— that would let the users browse details of the corresponding items, purchase online, subscribe to the notification of novelties, etc. At the most, they could link programs and advertisements to one from among a set of manually-developed applications (which is certainly insufficient) or to some URL that would open the web site of a specific provider. Thus, it is clear that much of the potential of Digital TV technologies for e-commerce remains underexploited.

3 Elements to Engineer Personalized Interactive Applications

The system we describe in this chapter extends the proposal of [22] to generalize the provision of interactive commercial applications in "*the user around the marketplace*" scenarios, introducing the ability to personalize (i.e. to craft according to the knowledge available about the target user) not only the type of service provided, but also the information displayed (which may come from different sources, just like in web mashups) and the commercial functionalities enabled. The elements that come into play in the composition of such applications are determined by the semantic and rule-based nature of the reasoning behind the personalization features. Next, we enumerate those elements briefly to get an overall idea of our approach:

- First, we use a domain ontology (expressed in OWL) to formally represent the concepts and relationships of the e-commerce domain, as well as the properties of the items available to recommend and the properties that may be used to characterize the user's context. This ontology brings together metadata from several specifications, including MPEG-7 and eCl@ss.[1] At the core, there is a hierarchy that classifies items and their attributes as shown in the micro-excerpt of Fig. 2, which includes a documentary about the *Bergamo Alps*, *vinegar* and *wine* brands from the Italian city of *Modena*, and a *sports car* also manufactured in Modena.
- To capture the information available about the user's interests, preferences and needs, we manage profiles that store demographical information (e.g. gender, age, marital status, income, etc) plus consumption histories to keep track of items purchased/consumed in the past. Those items are linked to a number between -1 and 1 called the *degree of interest* (DOI) of the user in

[1] We chose eCl@ss instead of other products-and-services categorization standards for the reasons of completeness, balance and maintenance discussed in [31].

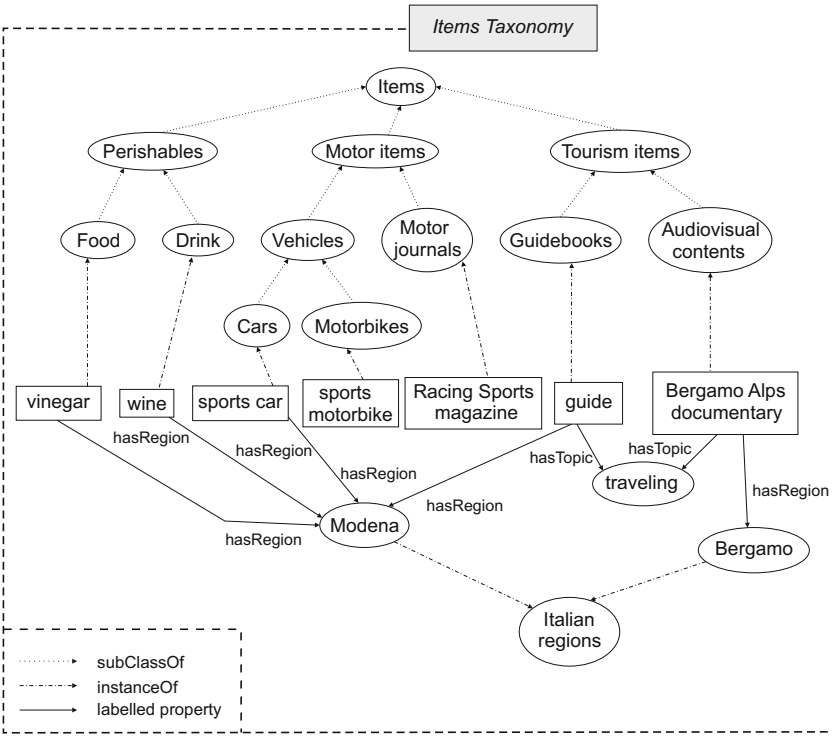


Fig. 2. A micro-excerpt from the items taxonomy of the domain ontology

them: 1 represents the greatest liking; −1 the greatest disliking. This number may be explicitly provided by the user or inferred from ratings given to other related concepts in the domain ontology —the procedure in this regard is identical to the one presented in [28].

– In order to decide whether a given item or a related piece of information might be appealing to a user, our system runs a hybrid filtering strategy against his preferences and the semantic annotations of the item in the domain ontology (further details will be given in Sect. 4.1). For example, the reasoning can relate the *Raci* magazine and a *sp* of Fig. 2 through the ancestor *Motor tems*; likewise, a given *guidebook* and *Bergamo Alps documentary* would be recognized as similar items because they share the attribute *traveling*, and also because they are bound to two *Italian regions* (*Modena* and *Bergamo* are sibling attributes).
– The services provided by the interactive commercial applications are characterized semantically as per the OWL-S standard, which provides an extensive ontology of functions where each class corresponds to a class of homogeneous functionalities. On top of that ontology, we have built a *services taxonomy* that represents the capabilities of the elements that may be included in an application. A micro-excerpt from the services taxonomy is depicted in

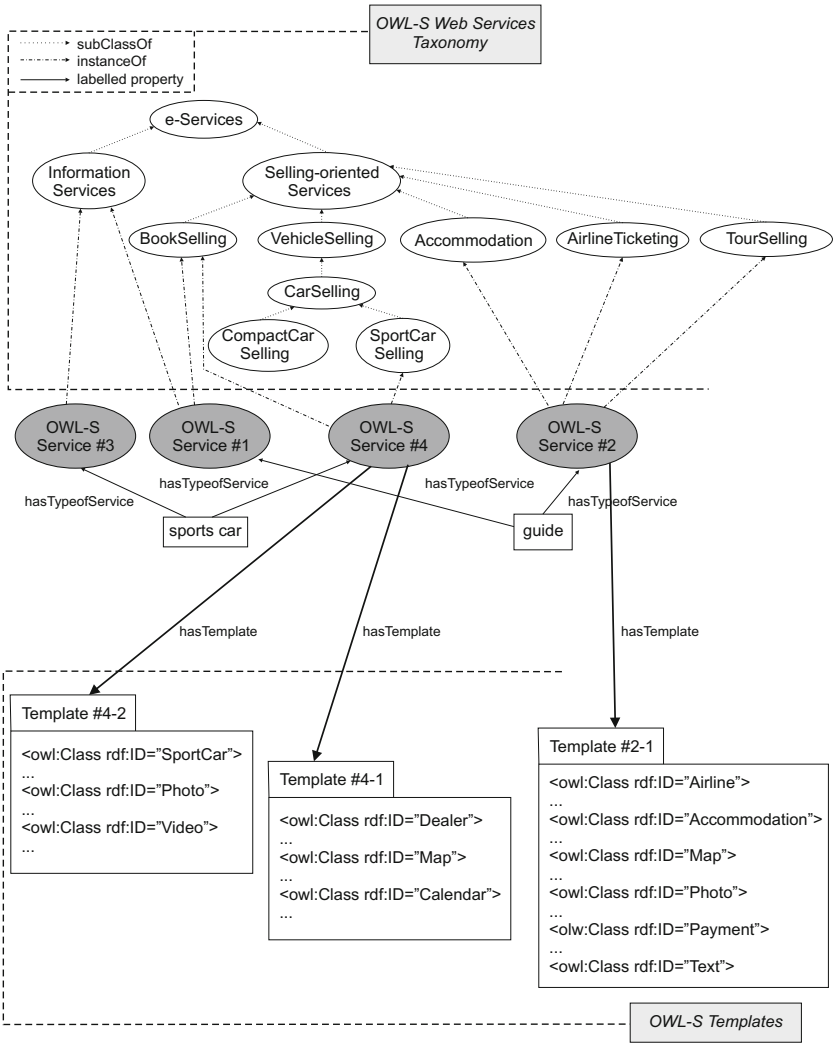


Fig. 3. A micro-excerpt from the taxonomy of semantic web services and templates

Fig. 3, with each item linked to different types of services by *hasTypeofService* properties.

Actually, the domain ontology merges the items and services taxonomy, so that the semantic reasoning features can relate the functionalities available to the items recorded in the user profiles. For example, the guidebook about Modena is associated to a type of service (denoted by *OWL-S Service* #1 in Fig. 3) that may be used to purchase one copy while browsing information about the main tourist attractions of Modena; it is also linked to a service (*OWL-S Service* #2) that may be used to purchase a travel to this Italian region with possibilities to book accommodation and flight. Similarly, the

sports car is linked to a service (*OWL S Service* #3) that may be used to simply describe its features, and to a second one (*OWL-S Service* #4) that may offer the chance to arrange an appointment with a dealer.

– Whereas the OWL-S service descriptions define functionalities from an abstract point of view, their implementation is accomplished by means of *templates*, which are reusable and configurable pieces that provide the code that glues together different functional elements and pieces of content. This includes the programming of the user interfaces and the logic needed to interact with web services as per the SOAP (*Simple Object Access Protocol*) standard [30]. Actually, a same type service as described above can be offered delivered multiple templates, which differ from each other in the interactive elements or the pieces of information shown to the user. This fact is represented in Fig. 3, where the elements of each template are identified by semantic concepts (e.g. *Map*, *Dealer* or *Calendar*), conveniently formalized in the ontology.

4 The Personalization Procedures

The support we propose in this chapter for "*the user around the marketplace*", as explained in Section 1, brings in innovations regarding the reasoning behind the recommendation of items and the composition of personalized, interactive commercial applications. The procedures followed in these two tasks are explained in Sections 4.1 and 4.2, respectively. Some final considerations about the *feedback* needed to keep the users' profiles updated —a key aspect to ensure the quality and accuracy of the personalization— are given in Section 4.3.

4.1 Reasoning-Driven Recommendation of Items

In our approach, the selection of items is done by matching the information captured in the user's profile with the classes and properties from the ontology that characterize his context and those of the items available to recommend. To this aim, we incorporate content-based filtering and collaborative filtering strategies. Next, we briefly explain how our two filtering strategies decide whether a given item $\mathcal{I}$ is appealing to a user $\mathcal{U}$, whose profile is denoted by $\mathcal{P}$.

Content-Based Strategy. Our content-based strategy consists of computing a *matching level* between the item $\mathcal{I}$ and $\mathcal{U}$'s preferences, by averaging the values of similarity between $\mathcal{I}$ and the items stored in the profile $\mathcal{P}$, weighed by their respective DOIs. Intuitively, the matching level grows with the similarity between $\mathcal{I}$ and the favorite items for $\mathcal{U}$ (i.e. those with the highest ratings in his profile). To measure such similarity, we have adapted the semantic similarity metric presented in [31], that quantifies the strength of the relationships that can be inferred —from the knowledge captured in the domain ontology— between $\mathcal{I}$ and $\mathcal{U}$'s preferences. Specifically, our metric considers not only the explicit relations defined by the hierarchy of classes, but also others which are hidden behind the

attributes of the items. Therefore, we talk about two similarity criteria, which are finally weighed and combined by means of a factor $\alpha \in [0, 1]$.

- The notion of *hierarchical similarity* has appeared in many previous approaches (see for example [32, 33, 34]), and consists of valuing the relationship between two nodes of the ontologies by the existence and specificity of a common ancestor in a hierarchy of classes. In our approach, the value of hierarchical similarity between $\mathcal{I}$ and each item i in U's profile grows with the depth of their *lowest common ancestor* (LCA) and also with its proximity to $\mathcal{I}$ and i in the items taxonomy. The depth of a node is given by the number of hierarchical links traversed to reach the node from the root of the hierarchy; thereby, the hierarchical similarity between two nodes is 0 if they do not have other common ancestor than the root class. For example, in Fig. 2, the *sports car* is more similar to the *sports motorbike* than to the *Racing Sports magazine*, because the LCA between the two sports vehicles is more specific (deeper) than the LCA between the *sports car* and the *motor magazine* (i.e. depth(*Vehicles*)=2 > depth(*Motor items*)=1).
- The notion of *inferential similarity* consists of measuring similarity by looking at relationships between the semantic attributes of the items compared. In this regard, two items are considered similar if they have common attributes (e.g. *traveling* for the *guidebook* and the *Bergamo Alps* documentary in Fig. 2) or sibling ones (e.g. *Modena* and *Bergamo*). To calculate the value of inferential similarity between $\mathcal{I}$ and i, in addition to the existence of common or sibling attributes, we consider the level of interest of $\mathcal{U}$ in such attributes (i.e. his/her DOI indexes in those attributes). Consequently, inferential similarity between both items is high when they share many attributes and when these attributes are highly rated in $\mathcal{U}$'s profile.

After computing the *matching level* between $\mathcal{U}$ and the item $\mathcal{I}$ (by combining the semantic similarity values and DOI indexes), our content-based strategy recommends the item to the user if the resulting value is greater than a given threshold. Otherwise, $\mathcal{I}$ is considered again by the collaborative strategy.

Collaborative Strategy. Our collaborative strategy starts out by delimiting the neighborhood of the user $\mathcal{U}$, matching his profile $\mathcal{P}$ against those of other users. To this aim, we first create a *rating vector* containing the DOIs of the item classes most appealing or most unappealing to the user (identified by DOIs close to 1 and -1, respectively). Next, we look for other profiles that contain DOIs for at least 70% of those classes, and create their respective rating vectors. Finally, we compute the Pearson-r correlation [2] between the rating vector of each partial stereotype and the vector corresponding to the user $\mathcal{U}$.

The user's neighborhood is formed by the profiles that yield correlation values greater than a given threshold γ. Once the neighbors have been identified, we predict the level of interest of the user in the item $\mathcal{I}$ by considering both his preferences and the interest of his neighbors in it. Specifically, if a neighbor has rated this item, we use his DOI index; otherwise, we predict his level of interest

by computing the matching level between $\mathcal{I}$ and the neighbor's preferences as explained above. As a result, the interest value predicted for $\mathcal{I}$ is greater when this item is very appealing to the selected neighbors and these are strongly correlated with $\mathcal{U}$'s preferences. Again, $\mathcal{I}$ is finally recommended to the user when the predicted interest is greater than a given threshold.

4.2 Composition of Interactive Commercial Applications

The generation of an interactive commercial application is triggered when the user shows interest in one of the items recommended to him. Then, it is necessary to decide which type of service best matches his preferences, and which is the most convenient template to create one application that provides the service. The selected template must be finally populated by retrieving contents for each one of its interactive elements.

In order to select the type of service to provide, we adopt an approach similar to the one presented in [35], which consists in using SWRL (*Semantic Web Rules Language*) rules to relate user preferences and context information in a personalization system. We have defined a set of rules to associate personal information of the users (such as hobbies and income) with the kind of services classified into the services taxonomy. The user preferences are antecedents in the rules, while the service types appear in their consequents. For example, by the *SWRL rule* #1 shown below, we infer that a user with sufficient income who is fond of traveling may appreciate services that permit to book a tour (instead of services that, for instance, only describe the tourist attractions of a given destination). Likewise, *SWRL rule* #2 makes it possible to conclude that a user with high income will likely be interested in services for the selling of luxury items (to the detriment of services that, for instance, only provide information about them). Such inferences are made by a *Description Logic* (DL) reasoning module that computes a relevance factor for each type of service associated to the item chosen by the user. Obviously, the factor is greater when the type of service appears in the consequent of a rule whose antecedent is highly rated among the user's preferences.

SWRL rule #1

user (?u) ∧ swrlb:notEqual (income (?u), "LOW") ∧
swrlb:equal (hobby (?u), "TRAVEL") ⇒
appealingTypeOfService (?u,?ws) ∧ TourSellingService (?ws)

SWRL rule #2

user (?u) ∧ swrlb:equal (income(?u), "HIGH") ⇒
appealingTypeOfService (?u,?ws) ∧ LuxuryItemsSellingService (?ws)

Having selected a type of service, the next step is to decide which one of the templates associated to it may be most interesting for the user. In other words, the goal is to identify the most suitable interactive elements to assemble an interactive application about the item selected by the user. This decision is driven by both the user preferences (e.g. the kind of elements the user has accessed in previous applications) and parameters such as the computational power, input/output capabilities or availability of a return channel for bidirectional communication. Therefore, for example, we refrain from using templates with maps in the case of users who have never fiddled with such artifacts in previous applications, while we limit the amount of text to be shown in the case of small screen devices or with users who have never fully read lengthy descriptions of other items.

Following the selection of a template to shape the application, we use a semantics-enhanced *registry* based on the UDDI (*Universal Description, Discovery and Integration*) standard [30] to look for services categorized under the selected service type, and also for contents to place in all the elements of the selected template. For example, assembling an application via *Template #2-1* shown in Fig. 3 requires to discover selling services that offer information about airlines and hotels providers, as well as maps and photos of rooms. Our registry works much like the one presented in [36], which takes advantage of the expresiveness of OWL-S to match the capabilities offered against the requested ones. In this line, we have developed an OWL-S/UDDI matchmaker to map the OWL-S service profile into the corresponding WSDL representations of the services offered in the semantic registry. To conclude, it is necessary to invoke the OWL-S services discovered in the preceding step, by exploiting the WSDL descriptions of message formats and in/out arguments provided by the service groundings. This way, our application composition logic retrieves contents that are pre-filtered considering the user preferences, which are finally put into the template.

It is worth noting that, for certain types of services, we can populate templates with information about providers who are not listed in the registry, either because they are online but do not offer a web services interface (as it happens with many providers listed in sites like *PriceGrabber*, *PriceRunner* or *bizrate*), or because they have never even thought about e-commerce (as many small- to medium-sized enterprises known to the *Yellow Pages* or *Google Maps*). The interest of showing these providers to a user may have to do with the price/availability of their offerings or their physical proximity to his location.

4.3 Feedback

The interaction between a user and the interactive applications generated for him makes it possible to infer useful information to update his profiles and to refine future recommendations of items and tailor-made applications. To begin with, the user can rate the composed item. If he does so, his rating is also assigned to the item being advertised in the applications, so that both the item and its rating are stored in his profile. Otherwise, we can infer ratings from parameters such

as the time spent interacting with the application and the number of interactive elements reviewed in it. This information is exploited by our content-based and collaborative strategies during the selection of the most relevant items for each user, as detailed in Sect. 4.1.

Feedback is also harnessed to infer useful information for the selection of the most suitable templates for each user. In this regard, we track, for example, the kind of interactive elements mostly accessed by each user (e.g. videos against photos), the length of the textual descriptions the user queried in previous applications (which can be easily inferred from the scrolling), and his preferences for some specific information provider (e.g. the user tends to go through photos extracted from *Flickr* against those from *TwitPick*), just to name a few possibilities. All these parameters help improve the quality of the future personalized i-spots shown to the user.

5 Our Proposal in DTV Advertising

We have implemented the approach described in this chapter as an enhancement to the MiSPOT recommender system [28], which could select the most suitable advertisements to display while the user is watching TV, matching the knowledge available about the user with the semantic characterization of the items available to recommend and the low-level (color, texture, motion, ...) or high-level (type of show, topic, genre, ...) features of the scenes. The new version of the system offers the possibility of linking the advertisements with interactive applications to run on MHP set-top boxes. This environment poses a number of specific design and implementation challenges, mainly related to the fact that Digital TV is still much based on broadcasting, with no means for bidirectional communication available in many cases.

Since we want the personalization features to be permanently available, even in scenarios with only downstream communication from servers to receivers, we have to run the filtering procedures locally in the latter. To this aim, we have adapted the logic presented in Section 4 to the architecture of Figure 4, that works as follows:

- First, as it is not possible to broadcast at the same time all the available items, functional elements, templates and pieces of information, we make a *pre-selection* to deliver only the material that is potentially most interesting for the audiences expected at any given moment in the region covered by the broadcasting facilities. This is a filtering process in itself, though not driven by the profile of an individual user, but rather by a set of *stereotypes* that average the preferences and needs of different groups of users.
- Second, we have devised a *pruning* procedure to reduce the amount of information to be handled by the receivers. This procedure consists of cutting off metadata from the domain ontology to leave only the most relevant concepts about the pre-selected items. As a result, we get *partial ontologies* of a manageable size for the receivers to work with, plus *partial stereotypes* to support the final filtering.

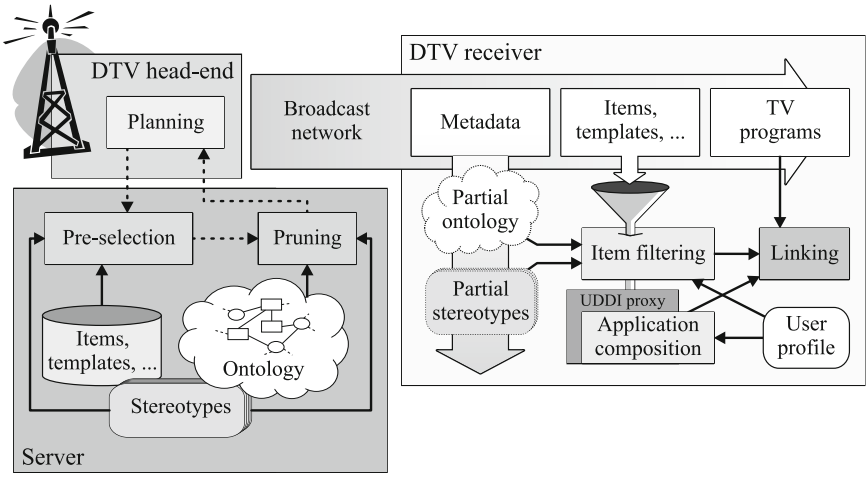


Fig. 4. The architecture of the enhanced MiSPOT system

- Following the pre-selection and pruning processes, a *planning* module in the Digital TV head-end takes care to arrange items, functional elements, templates, pieces of information, partial ontologies and partial stereotypes in the broadcast emissions. When those data have been loaded into the receivers, it is finally possible to run the filtering algorithms to decide what items will be offered to each individual user, and to build the corresponding personalized interactive applications.
- Clearly, with no way to communicate with remote servers, the collaborative filtering strategy cannot do with the profiles of real users, but rather with the partial stereotypes delivered through broadcast. We keep this *pseudo-collaborative* approach anyway because it helps overcome the problem of overspecialization that is typical of content-based strategies. Likewise, we have introduced means to differentiate three different modes of interactivity that make sense in Digital TV: *real interactivity*, requiring permanent access to the Internet for information retrieval; *local interactivity*, dealing exclusively with contents delivered in the broadcast streams; and *deferred interactivity*, halfway between the others, storing and forwarding information when a return channel is available from time to time.
- The semantic registry is accessed via a proxy module running in the receivers, that provides a unique interface to retrieve elements regardless of whether they are received through broadcast or through the return channel.

5.1 A Simple Example

To illustrate the reasoning and the functionalities enabled by the new version of the MiSPOT system, we shall consider the case of a childless man from London in his early 20s, with significant income and subscribed to the *Racing Sports*

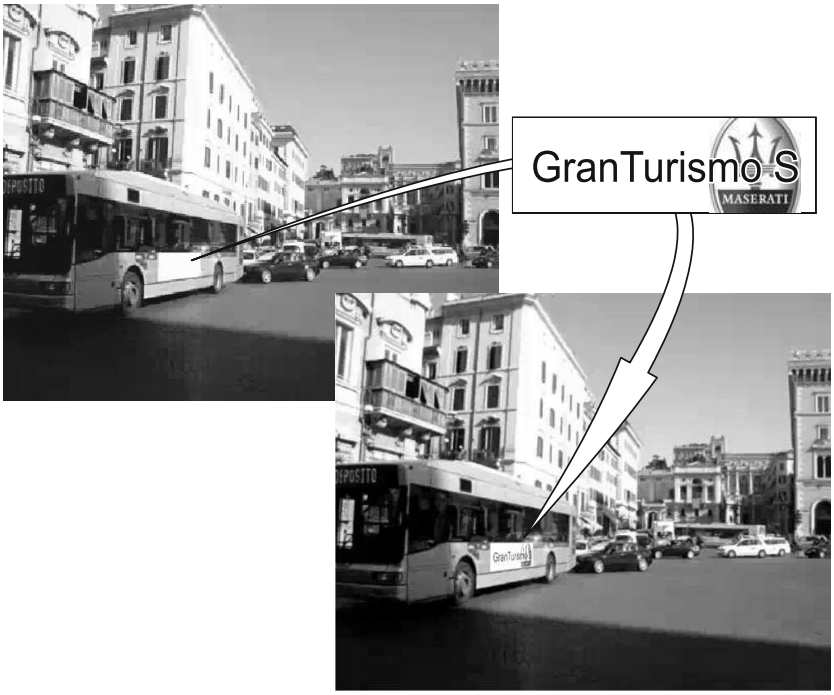


Fig. 5. Warping the sports car logo over a bus

magazine. This viewer is currently watching a documentary about Italy, using a high-end Digital TV receiver permanently connected to a cable network. The provider's repository of advertising material contains logos and videos for a range of cars and tourist resorts worldwide.

Assume that, as shown in the top-left corner of Fig. 5, the current scene of the movie is set in a noisy city street, and that the producers have identified the banner on the bus as a suitable place to render static images with an aspect ratio nearing 4:1. As the first step in the MiSPOT operation, the low-level scene matching discards audio and video advertisements, and the same happens with non-fitting static images. Then, the semantic similarity metric identifies the cars as the most suitable items to advertise within this scene, because cars are commonly found in city streets and the semantic characterization of the tourist resorts makes them more suitable for relaxing scenes.

When it comes to reasoning about the viewer's preferences, the data in his profile lead to finding a sports car as a potentially interesting item, for several reasons: (i) the explicit interest in motor sports reinforces the relevance of cars over tourist resorts, as we do not know anything about the viewer's fondness for traveling; (ii) the viewer's high economic power does not disregard him as a potential client for expensive items; and (iii) the viewer does not need space for children, which could promote other types of cars instead of sports ones. The

specific brand selected in this example was found more relevant than others in this context because it has its headquarters in the Italian city of Modena. Thus, the car's logo ends up rendered within the hook as shown in the bottom-right corner of Fig. 5.

We assume that the viewer feels curious about the sports car and activates the advertisement using his remote control, which tells the system to compose an i-spot about it. In doing so, the high level of income first leads the SWRL engine to select a car selling service instead of one that would simply provide information about the car. Next, the computing and communication capabilities of the viewer's high-end receiver makes it possible to use a template with demanding interactive elements like maps and video players. Finally, the viewer's for motor-related issues suggests getting information from specialized sites like *w guide.com*. Figures 6 to 8 show a few snapshots of the resulting i-spot: the first one displays information about the car's specifications retrieved from the aforementioned web site; another tab provides a collection of pictures from Flickr and videos from *Youtube*; the last one provides an interactive map to locate dealers of the car brand around the region where the viewer lives, plus a calendar to arrange an appointment for a test drive.

Fig. 6. Snapshots of the interactive application assembled for the sports car (I)

5.2 Experimental Evaluation

Within the aforementioned settings, we conducted experiments to assess the personalization quality achieved by the MiSPOT system in terms of *precision* (%of advertised items that the viewers rate positively) and overall perception of

Fig. 7. Snapshots of the interactive application assembled for the sports car (II)

Fig. 8. Snapshots of the interactive application assembled for the sports car (III)

our proposal. The experiments involved 60 viewers recruited among our graduate/undergraduate students and their relatives or friends. They made up a diverse audience, with disparate demographic data and educational backgrounds; there were nearly as many men as women (54% vs 46%), with ages ranging from 12 to 55 years old.

Prior to making any recommendations, we defined a set of 15 stereotypes by clustering the viewer profiles that had built up during previous experiments with the AVATAR recommender system [40]. Specifically, 14 clusters contained the profiles that had comparatively high (close to 1) or comparatively low (close to −1) DOIs for items classified under *Sports*, *Nature*, *Technology*, *Science*, *Health*, *Culture* or *Traveling*. One final cluster gathered the profiles that did not meet any of those conditions. From each cluster, one stereotype was computed by averaging the DOIs of the profiles they contained. Having done this, we asked each viewer to rate his/her interest in topics related to *Sports*, *Nature*, *Technology*, *Science*, *Health*, *Culture* and *Traveling* with a number between 0 and 9, and their individual profiles were then initialized by weighing the DOIs of the corresponding stereotypes.

The viewers interacted with our prototype system during at least 6 hours over a period of 3 months. After each session, the viewers were faced with a list of the items that had been advertised to them, which they had to rate between 0 and 9. At the end, we collected the log files and analyzed the data, measuring 62% precision, which is lower than the performance achieved by current server-based recommender systems, but much greater than the precision of syntactic approaches to receiver-side personalization in Digital TV (e.g. in [40] we had measured the approach of [41] to reach barely above 20%).

As regards the viewers' perceptions, we ran a poll offline asking them to rate the personalization service, the new advertising model and the interest of enhancing the traditional TV publicity with interactive commercial functionalities. The results are shown in Table 1.

To begin with, it is noticeable that the viewers' satisfaction with the personalized offerings was quite high, with 68% of the viewers rating the experience

Table 1. Experimental results: Viewers' opinions

	Very positive	Positive	Neutral	Negative
Opinion about the personalized offerings	29%	39%	23%	9%
Opinion about the advertising model	36%	31%	18%	15%
Interest in the interactive applications	29%	39%	21%	11%
Opinion about the functionalities delivered	19%	46%	23%	12%
Quality and coherence of the contents displayed	12%	28%	33%	27%

positively or very positively. Many of the test subjects noticed that the quality of the recommendations increased as they interacted with the system (obviously, thanks to the relevance feedback), but they agreed that the targeting of the publicity was not any worse than usual even during the first sessions. In what concerns the advertising model, the viewers' appreciation was just as good, with 67% of positive or very positive ratings. Here, almost 15% of the viewers considered the product placements a nuisance, but this was often due to cases in which the integration of the advertisements within the TV programs was not always as smooth as desired —a question of technical refinements. Finally, regarding the interactive applications, a significant number of viewers (more than 35%) admitted that they do not yet think of using the TV receivers for anything else than TV watching. Anyway, nearly 65% of them gave positive or very positive ratings to the functionalities delivered, which confirms the interest of the local and deferred interactivity modes. The bad news have to do with the quality and coherence of the contents displayed. In line with the comments given in [24], this fact reveals that it is necessary to develop more fine-grained reasoning about the contents, or to restrict the possible sources by requiring greater amounts of metadata.

6 Conclusion

In this chapter, we have motivated the need to improve the support offered nowadays to casual e-commerce experiences through different media, by means of suitable personalization tools and engines to generalize the provision of interactive applications. The key aspects have to do with contextualizing the selection of the items that will be offered to the users, delivering the information they may find more valuable and understandable in the most accessible way, and identifying the most convenient providers. We have described a system that furnishes these features through a combination of semantic and rule-based reasoning techniques.

References

1. Hung, L.: A personalized recommendation system based on product taxonomy for one-to-one marketing online. Expert Systems with Applications 29, 383–392 (2005)
2. Cho, Y., Kim, J.: Application of Web usage mining and product taxonomy to collaborative recommendations in e-commerce. Expert Systems with Applications 26, 233–246 (2004)
3. Ardissono, L., Gena, C., Torasso, P., Bellifemine, F., Chiarotto, A., Difino, A., Negro, B.: User modeling and recommendation techniques for personalized Electronic Program Guides. In: Personalized Digital Television, pp. 474–486. Targeting programs to individual users. Kluwer Academic Publishers, Dordrecht (2004)
4. Foltz, P.W., Dumais, S.T.: Personalized information delivery: An analysis of information filtering methods. Commun. ACM 35(12), 51–60 (1992)
5. Leung, C.W., Chan, S.C., Chung, F.L.: A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. Knowledge and Information Systems 10(3), 357–381 (2006)

6. Mobasher, B., Jin, X., Zhou, Y.: Semantically-enhanced collaborative filtering on the Web. In: Web Mining: Applications and techniques, pp. 57–76. Idea Group, Hershey (2004)
7. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on the World Wide Web, Hong Kong, China, pp. 285–295 (2001)
8. Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction 12(4), 331–370 (2002)
9. Smyth, B., Cotter, P.: Surfing the digital wave: Generating personalized TV listings using collaborative, case-based recommendation. In: Proceedings of the 3rd International Conference on Case-Based Reasoning, Munich, Germany, pp. 561–571 (July 1999)
10. Staab, S., Studer, R. (eds.): Handbook on ontologies. Springer, Heidelberg (2003)
11. McGuinness, D., van Harmelen, F.: Web Ontology Language overview. W3C Recommendation (2004)
12. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263–277 (2007)
13. di Flora, C., Ficco, M., Russo, S., Vecchio, V.: Indoor and outdoor location-based services for portable wireless devices. In: 25th International Conference on Distributed Computing Systems, Columbus, USA, pp. 244–250 (June 2005)
14. Stanojevic, M., Vranes, S.: Semantic classifier for affective computing. In: International Conference on Computational Intelligence for Modelling Control & Automation, Vienna, Austria, pp. 849–854 (July 2009)
15. Hu, S., Muthusamy, V., Li, G., Jacobsen, H.A.: Distributed automatic service composition in large-scale systems. In: Proceedings of the 2nd International Conference on Distributed Event-Based Systems, Rome, Italy, pp. 233–244 (July 2008)
16. Kvaloy, T.A., Rongen, E., Tirado-Ramos, A., Sloot, P.: Automatic composition and selection of semantic web services. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 184–192. Springer, Heidelberg (2005)
17. Lin, N., Kuter, U., Hendler, J.: Web Service composition via problem decomposition across multiple ontologies. In: Proceedings of the IEEE Congress on Services, Salt Lake City, USA, pp. 65–72 (July 2007)
18. Sirin, E., Parsia, B., Wu, D., Hendler, J.A., Nau, D.S.: HTN planning for Web Service composition using SHOP2. Journal of Web Semantics 1(4), 377–396 (2004)
19. Agarwal, S., Lamparter, S.: User preference based automated selection of web service compositions. In: Proceedings of the Workshop on Dynamic Web Processes, in Conjunction with ICSOC, Amsterdam, The Netherlands, pp. 1–12 (December 2005)
20. He, J., Yen, I.L.: Adaptive user interface generation for web services. In: Proceedings of the IEEE International Conference on e-Business Engineering, Hong Kong, China, pp. 536–539 (October 2007)
21. Pei, Z., Zhenxiang, Z.: A framework for personalized service website based on TAM. In: Proceedings of the International Conference on Service Systems and Service Management, Troyes, France, pp. 1598–1603 (October 2006)
22. Benslimane, D., Dustdar, S., Sheth, A.: Services mashups: The new generation of Web applications. IEEE Internet Computing 12(5), 13–15 (2008)
23. Bakalov, F., König-Ries, B., Nauerz, A., Welsch, M.: Ontology-based multidimensional personalization modeling for the automatic generation of mashups in next-generation portals. In: Proceedings of the 1st International Workshop on Ontologies in Interactive Systems, Liverpool, UK, pp. 75–82 (September 2008)

24. Cappiello, C., Daniel, F., Matera, M., Pautasso, C.: Information quality in mashups. IEEE Internet Computing 14(4), 14–22 (2010)
25. Anderson, C., Wolff, M.: The web is dead. long live the Internet. Wired Magazine (September 2010)
26. Morris, S., Smith-Chaigneau, A.: Interactive TV standards. Focal Press (2005)
27. Thawani, A., Gopalan, S., Sridhar, V.: Context-aware personalized ad insertion in an Interactive TV environment. In: Proceedings of the 4th Workshop on Personalization in Future TV, Eindhoven, The Netherlands (August 2004)
28. Kastidou, G., Cohen, R.: An approach for delivering personalized ads in interactive TV customized to both users and advertisers. In: Proceedings of the 4th European Conference on Interactive Television, Athens, Greece (May 2006)
29. Lekakos, G., Giaglis, G.: A lifestyle-based approach for delivering personalised advertisements in digital interactive television. Journal of Computer-Mediated Communications 9(2) (2004)
30. López-Nores, M., Pazos-Arias, J.J., García-Duque, J., Blanco-Fernández, Y., Martín-Vicente, M.I., Fernández-Vilas, A., Ramos-Cabrer, M., Gil-Solla, A.: MiSPOT: Dynamic product placement for digital TV through MPEG-4 processing and semantic reasoning. Knowledge and Information Systems 22(1), 101–128 (2010)
31. Paternò, F., Santoro, C., Spano, L.: Model-based design of multi-device interactive applications based on web services. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R.O., Winckler, M. (eds.) INTERACT 2009. LNCS, vol. 5726, pp. 892–905. Springer, Heidelberg (2009)
32. Hepp, M., Leukel, J., Schmitz, V.: A quantitative analysis of product categorization standards: Content, coverage and maintenance of eCl@ss, UNSPSC, eOTD, and the RosettaNet Technical Dictionary. Knowledge and Information Systems 13(1), 77–114 (2007)
33. Walsh, A.E.: UDDI, SOAP, and WSDL: The Web Services Specification reference book. Pearson Education, London (2002)
34. Blanco-Fernández, Y., Pazos-Arias, J., Gil-Solla, A., Ramos-Cabrer, M., López-Nores, M., García-Duque, J., Fernández-Vilas, A., Díaz-Redondo, R., Bermejo-Muñoz, J.: AVATAR: Enhancing the personalized television by semantic inference. International Journal of Pattern Recognition and Artificial Intelligence 21(2), 397–422 (2007)
35. Ganesan, P., Garcia-Molina, H., Widom, J.: Exploiting hierarchical domain structure to compute similarity. ACM Trans. Inf. Syst. 21(1), 64–93 (2003)
36. Rada, R., Mili, H., Bicknell, E., Blettnet, M.: Development and application of a metric on semantic nets. IEEE Trans. Syst., Man, and Cybern. 19(1), 17–30 (1989)
37. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. J. Artif. Intell. Res. 11(4), 95–130 (1999)
38. Plas, D., Verheijen, M., Zwaal, H., Hutschemaekers, M.: Manipulating context information with SWRL. Report of A-MUSE project (2006)
39. Kawamura, T., de Blasio, J.A., Hasegawa, T., Paolucci, M., Sycara, K.P.: Public deployment of semantic service matchmaker with UDDI business registry. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 752–766. Springer, Heidelberg (2004)
40. Pazos-Arias, J., López-Nores, M., García-Duque, J., Díaz-Redondo, R., Blanco-Fernández, Y., Ramos-Cabrer, M., Gil-Solla, A., Fernández-Vilas, A.: Provision of distance learning services over Interactive Digital TV with MHP. Computers & Education 50(3), 927–949 (2008)

41. Ghaneh, M.: System model for t-learning application based on home servers (PDR). Broadcast Technol. 19 (2004), http://www.nhk.or.jp/strl/publica/bt/en/rep0019.pdf

List of Acronyms

DTV : Digital Television.
OWL : Ontology Web Language.
OWL-S : Ontology Web Language – Services.
WSDL : Web Services Description Language.
RSS : Really Simple Syndication.
HTML : HyperText Markup Language.
HTTP : HyperText Transfer Protocol.
HTTPS : HyperText Transfer Protocol – Secure.
API : Applications Programming Interface.
VoIP : Voice over Internet Protocol.
MHP : Multimedia Home Platform.
MPEG : Moving Pictures Experts Group.
DOI : Degree of Interest.
SOAP : Simple Object Access Protocol.
LCA : Lowest Common Ancestor.
SWRL : Semantic Web Rules Language.
DL : Description Logic.
UDDI : Universal Description, Discovery and Integration.

About the Authors

Dr. Martín López-Nores was born in Pontevedra, Spain in 1980. He received the Telecommunications Engineering Degree from the University of Vigo in 2003, and the Ph.D. degree in Computer Science from the same University in 2006. Nowadays, he is an associate professor in the Department of Telematics Engineering, teaching in courses related to computer networks, operating systems and information services. Starting from works on applied formal specification techniques, his research interest have evolved to embrace (i) the design and development of interactive services for a range of consumer electronics devices, (ii) the design and evaluation of communication protocols and innovative applications for mobile ad-hoc networks, and (iii) the management of health-related data in semantics-based recommender systems and pervasive computing environments.

Dr. Yolanda Blanco-Fernández was born in Orense, Spain in 1980. She received the Telecommunications Engineering Degree from the University of Vigo in 2003, and the Ph.D. degree in Computer Science from the same University in 2007. Nowadays, she is an assistant professor in the Department of Telematics Engineering, teaching in courses related to network management systems, multimedia services and operating systems. Her main research activity involves development of personalization services for Interactive Digital TV and e-commerce, by exploiting technological foundations borrowed from the Semantic Web, Web 2.0 and cloud computing.

Prof. José J. Pazos-Arias was born in Baiona, Spain in 1964. He is Full Professor at Department of Telematics Engineering at the University of Vigo. He received his degree in Telecommunications Engineering from the Technical University of Madrid (UPM) in 1987, and his Ph.D. degree in Computer Science from the Department of Telematic Systems Engineering of the same University in 1995. He is the director of the Interactive Digital TV Laboratory (http://idtv.det.uvigo.es), which is currently involved with national and international projects, receiving funds from both public institutions and industry. With the aim of combining the power of semantic reasoning technologies and the participation phenomena arising in the knowledge society, his research is now focused on the use of social-semantic technologies to assist the users when it comes to facing complex decision takings in the cloud. In this regard, he is highly interested in gaining deeper knowledge in social network analysis and emergent semantics.

Chapter 12
Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies

Raja Jurdak, X. Rosalind Wang, Oliver Obst, and Philip Valencia

CSIRO ICT Centre, Australia
{raja.jurdak,rosalind.wang,oliver.obst,philip.valencia}@csiro.au

Abstract. Wireless Sensor Networks (WSNs) can experience problems (anomalies) during deployment, due to dynamic environmental factors or node hardware and software failures. These anomalies demand reliable detection strategies for supporting long term and/or large scale WSN deployments. Several strategies have been proposed for detecting specific subsets of WSN anomalies, yet there is still a need for more comprehensive anomaly detection strategies that jointly address network, node, and data level anomalies. This chapter examines WSN anomalies from an intelligent-based system perspective, covering anomalies that arise at the network, node and data levels. It generalizes a simple process for diagnosing anomalies in WSNs for detection, localization, and root cause determination. A survey of existing anomaly detection strategies also reveals their major design choices, including architecture and user support, and yields guidelines for tailoring new anomaly detection strategies to specific WSN application requirements.

1 Introduction

Wireless Sensor Networks (WSNs) represent an emerging system paradigm that tightly couples the network with its deployment environment [1]. Relying on resource-constrained embedded devices for communication, processing, and sensing, WSNs can experience unexpected problems during deployment, due to hardware, software, or environmental anomalies. The volatility of WSNs is always in tension with ambitious application goals, including long term deployments of several years, large scale networks of thousands of nodes, and highly reliable data delivery. As the WSN field matures, strategies for detecting (and possibly correcting) the anomalies that are inherent to their physically coupled low-end system design will only grow in importance. In fact, providing appropriate tools that can effectively detect and respond to anomalies can greatly increase uptake of the technology by stakeholders.

While significant work on conventional network management tools exists [2], WSN counterparts have been slow to gain traction within the community. One of the main challenges for WSN anomaly detection is determining where to embed the intelligence for detecting and localizing anomalies. While centralized approaches rely on more comprehensive network state information available at the back-end and are thus simpler to implement, distributed approaches

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 309–325.
springerlink.com

provide more scalable and responsive anomaly detection, as nodes can detect network problems in their vicinity immediately. A challenge for distributed anomaly detection is its implementation complexity and the limited state information available at resource-constrained sensor nodes.

Another key requirement for any anomaly detection strategy is catering to the needs and to the feedback of the human operator. A user-friendly detection strategy should provide several modes of notification, such as email and SMS alerts, and adapt its frequency of alerts to user feedback, in order to avoid "crying wolf" too many times and risking user apathy to more significant alerts. An effective anomaly detection strategy should also provide the versatility to cater to diverse user requirements, supporting both network managers who require detailed diagnostic information, and end users who are only interested in data quality.

A recent review article on anomaly detection in WSNs [3] focuses on data anomalies, mainly due to security attacks, and the statistical approaches for detecting them. Because of their tight coupling to often harsh physical environments, WSNs and other networks used in extreme conditions (like, e.g. in space [4]) are more likely than conventional networks to experience anomalies related to connectivity or hardware failures. Recent work also focuses on devising detection strategies that target network level [5,6], data level [7,8], or node and data level [9,10] anomalies.

One shortfall of existing strategies is that none of them comprehensively addresses network, node and data level anomalies in WSNs. A common reason for this are application-specific design choices in sensor networks that tend to tailor anomalies detection strategies to a family of applications with a given set of constraints and assumptions. The lack of comprehensive anomaly detection strategies for WSNs contributes to slower adoption and more frustration in deploying and maintaining these networks.

From a WSN user or operator perspective, it is crucial that a network management tool embeds the required intelligence to detect all possible anomaly types, as the network is perceived holistically as an intelligent data delivery system. To design such system-level tools demands a comprehensive understanding of all types of WSN anomalies, their likely causes, and their potential solutions. This chapter examines WSN anomalies from a systems perspective, covering anomalies that arise at the network, node and data levels. It introduces a simple process for diagnosing anomalies in WSNs for detection, localization, and root cause determination. A survey of existing anomaly detection strategies also reveals their major design choices, including architecture and user support, and yields guidelines for tailoring new anomaly detection strategies to specific WSN application requirements.

2 Types of WSN Anomalies

We begin by defining the scope of the term anomalies in this chapter. Anomalies can range from faults, such as complete hardware failures, to unexpected system

performance, such as gradual degradation. Note that certain outliers in spatial or temporal sensor data can signify events of interest in the monitored area, and should not be reported as anomalies to the network operator unless explicitly specified. Otherwise, separate data analysis tools can handle these outliers.

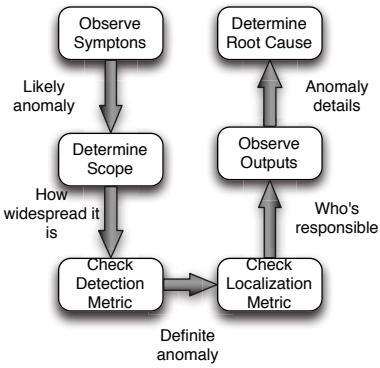


Fig. 1. Anomaly diagnosis process

The conditions that signal an anomaly relate to user policy for a particular application. For instance, an operator sets the frequency and timeout period for data delivery by the sensor nodes. These determine thresholds for detecting and reporting anomalies to operators. Whenever data from a sensor node is not received according to the expected schedule, the operator can be notified. The frequency, urgency and level of detail within the the notification can also be user-defined.

Anomaly diagnosis is established in conventional network management tools [2]; however, we revisit it here from a WSN perspective to expose how it can apply to the different types of WSN anomalies. The main goal of WSN anomaly diagnosis is mapping the symptoms to possible root causes, in order to possibly suggest remedial actions [9]. The process for characterizing a sensor network fault or anomaly is very similar to diagnosing an illness, as Figure 1 illustrates. The *symptoms* must first be examined, followed by a specification of the *scope* of the affected region. The process then involves some testing on the affected region (detection), where the operator must *localize* the anomaly causing node and expect some *diagnostic* information on the nature of the problem. This feedback yields a hypothesis on the most likely *root cause* of the problem.

Based on the above process, we now examine how to detect common WSN anomalies, which fall into three broad categories: (1) network anomalies; (2) node anomalies; (3) data anomalies. Figure 2 illustrates the scope of each of the three anomaly types, and Table 1 summarizes the most common WSN anomalies that fall into these three categories. The remainder of this section visits each anomaly type individually.

Table 1. WSN anomalies

Type	Symptom	Scope	Detection Metric	Localization Metric	Diagnostic Outputs
Network Anomalies	Loss of connectivity	many/all	packet delivery rate=0	packet origin address	packet delivery rate
	Intermittent connectivity	many/all	high delivery rate variability tx/rx link quality	packet origin address link endpoints	packet delivery rate link quality changes
	Routing loop	many	origin address =forwarder address	path node addresses	routing loop count, node IDs
	Broadcast storm	many/all	high broadcast	network-wide packet frequency	node IDs, storm duration, duplicate packet count
Node Anomalies	Solar charging degradation	one	sustained reduction in solar current	packet origin address	solar current degradation rate
	Battery degradation	one	battery voltage decrease rate	packet origin address	battery current degradation rate
	Node failures	one	lack of interaction with neighbors	node id with stale entries in neighbor tables	time last heard
	Node resets	one	packet counter = 0	packet origin address	timestamp last sequence number
Data Anomalies	Temporal anomalies	one	sensor value time series	packet origin address	duration/rate of change in local sensor values
	Spatial anomalies	many	local/neighborhood sensor reading variation	packet origin address	degree of skew from neighboring nodes
	Spatiotemporal anomalies	many	local time series and local/neighborhood comparison	packet origin addresses sensor IDs	degree of local and neighborhood variations

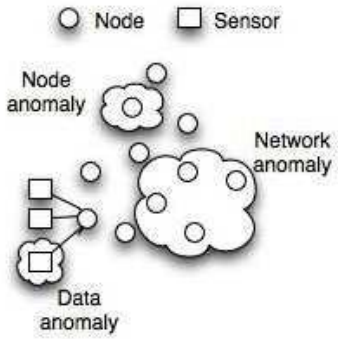


Fig. 2. Scope of each anomaly type

2.1 Network Anomalies

Network anomalies are communication-related problems that arise in WSNs. Their typical symptoms are an unexpected increase or decrease in amount of packets traversing the network.

Loss of Connectivity. The simplest type of network anomaly to detect is loss of connectivity. The interruption of incoming packets from two or more nodes indicates loss of connectivity with these nodes. Loss of connectivity can occur at a group of nodes, or indeed at all network nodes. Detection of connectivity losses follows a simple process: any network entity can keep track of incoming packets from other nodes. The lack of reception of packets from a group of nodes for a certain duration, defined by the timeout period set in the operator policy, signals partial or full loss of connectivity with these nodes. Alternatively, operators can set a threshold for the number of missed packets to signal loss of connectivity. The process for localizing this anomaly relies simply on extracting the node IDs of missing packets. The extraction of node IDs with missing packets can be done either at a central database or locally at a sensor node, depending on the detection architecture (see section 3.1 for more details). The anomaly's scope helps narrow down the possible root causes behind the loss of connectivity. If a group of nodes with a common routing parent has disappeared, the likely cause of the problem is a hardware or software failure at the parent node. Alternatively, if nodes lose connectivity in different parts of the network topology, the likely cause is less evident, and could be related to link degradation issues at these nodes.

Intermittent Connectivity. A related network anomaly is intermittent connectivity. Intermittent connectivity occurs when the frequency of data reception from certain nodes is highly variable relative to an operator-set link stability threshold. This anomaly also varies in scope and can cover few or all nodes. Detection of this anomaly involves setting a certain threshold on packet delivery

rate or link quality variability. Most collection protocols in WSN's, such as CTP [11], include unique sequence numbers for each packet, so a simple check of sequence number gaps can identify missing packets for each node in the network. The diagnostic parameter for further characterizing intermittent connectivity is the variability of connectivity, indicated by the packet delivery rate variance for affected nodes. The root cause of intermittent connectivity in most cases is highly time-variant link qualities and Received Signal Strength Indicator (RSSI) values, and it is heavily dependent on the anomaly's scope. Intermittent connectivity at a spatially clustered group of nodes in a single region of the network indicates high noise, multi-path, or interference affecting that region. To localize intermittent link qualities at disjoint links simply requires examination of the link's end points. The operator should be provided with a measure of how drastic a change has occurred for the link's quality to further diagnose this anomaly. Likely causes of sudden and transient changes in link throughput include environmental changes, such as storms or fires, or movement of animals, people, or objects in the deployment area, all of which can cause sharp variations in link qualities of low power radios. If the throughput of links in different regions of the network exhibit prolonged variations, this could indicate low receiver sensitivity or high inter-node distances relative to the radio transmission range of the nodes.

Routing Loops. Routing loops are yet another network anomaly whose detection is difficult at the back-end. Routing loops occur when a packet is relayed across several nodes and arrives back at the originating node. Because such packets may never make it to the base node, detecting this anomaly requires either: (1) injection of diagnostic packets by the base into the network to determine that the routing loops exist and the software problem that is causing them; (2) maintenance of the full network topology at the base node; or (3) use of source routing protocols to allow nodes to detect their own ID in the path to the destination, and to generate problem reports. Since routing loops inherently involve several nodes, all of the above methods involve large communication overhead to maintain node IDs in packets or to generate global topology information at the base, so they do not scale well with network size. Detection and resolution of routing loops within the energy constraints of WSN's remains an open issue, even in the latest IETF Routing Over Low Power Lossy Links (ROLL) draft standard [12].

Broadcast Storms. A final type of network anomaly for WSN's is broadcast storms, which typically affect many or all nodes in the network. Nodes that lose connectivity with their routing parents may decide to continuously broadcast packets to discover alternate paths to the base station. This behavior is due to Trickle [13] timers that drive the beacon period in collection protocols. Trickle timers are used for maintaining consistent state within WSN's. As long as there are no changes to the local state, a node doubles its Trickle timer, effectively doubling the beaconing period. However, when a node detects a change in its local state, it resets its Trickle timer to the lowest possible value (commonly

in the range of milliseconds). When multiple nodes simultaneously follow this behavior, the local channel becomes heavily congested. To make things worse, neighboring nodes may forward broadcast packets to nodes further away, which causes heavy congestion and eventually saturation on the wireless channel in the wider network. Fortunately, detecting broadcast storms is relatively simple: if a certain number of duplicate broadcast packets arrive from more than one node, then a broadcast storm has started. Network operators can set thresholds on the number of packet duplicates that indicate a broadcast storm, the minimum number of nodes involved, and the frequency of running this query. The anomaly detection system should provide operators with a list of nodes involved in the storm, the number of packet duplicates, and the duration of this storm. A common root cause of broadcast storms is loss of routing connectivity, which causes nodes to probe neighbors in hope of regaining connectivity.

2.2 Node Anomalies

Node level anomalies are rooted at hardware or software problems at a single node and are not related to communications with neighboring nodes. Because they arise in single nodes, distributed detection of node anomalies can be highly effective.

A common node-level symptom is when a node stops transmitting data. The most likely cause of this node anomaly is the failure/degradation of solar panels, which leads to on-board power drops.

Solar Panel Issues. There are four main possibilities for solar panel failures: (1) not enough sunlight during the day, (2) blocked solar panel, (3) shifted solar panel, and (4) solar panel degradation. During cloudy or rainy days, the solar charging current of several nodes is affected instead of just one. This exemplifies multiple node anomalies arising simultaneously, and it can be detected by observing the drop in solar current of the multiple nodes. This is in contrast to a network-related problem at the same group of nodes, which would exhibit connectivity losses that are not correlated to solar current changes.

Based on our deployment experience in many long term environmental monitoring sensor networks in forests, lakes, farms, and deserts around Australia [21], we have observed numerous unexpected obstructions to solar panels in the field. Blocked solar panels involve cases such as bird droppings, cob webs, or fallen leaves on the solar panel. Moving animals in the deployment region or strong winds can also shift solar panels away from the favourable direction for maximum solar harvesting. Both shifted or blocked solar panels result in dramatic and sustained drops in solar current. Finally, long term degradation of the panel can also cause node power drops. This anomaly requires long term observation of solar current trends to detect decreases in solar output over time.

Battery Issues. Another cause for power drops is battery failure or depletion. There are two possibilities when a battery does not provide enough power: (1) insufficient battery charging, and (2) hardware failure of the battery. Insufficient

battery charging can be diagnosed, locally or centrally, through examination of the solar current of the previous day, potentially combined with relevant weather data for indicating cloud cover during that period. Battery failure can be diagnosed when a node fails to send data during the night. If a node that has enough harvested solar energy to remain active all night ceases to send data before sunrise, then it is highly likely that the node's battery has reached the limit for its recharge cycles and needs replacement.

Node Failure. The node memory, CPU, or radio may also enter a locked state or fail during a deployment. This situation may arise due to poor or defective hardware components, or poor software integration of the components. Detecting this anomaly is challenging, as the problem with a specific component may not exhibit any detectable symptoms by neighbors or at the base node. In this case, only proactive checking of node integrity can detect the problem's existence.

Node anomalies that cause the node to stop operating are more easily detectable. If a node stops interacting with any other node for a pre-specified period of time, the node is deemed to have failed. In beaconing based routing protocols, this entails the examination of neighboring nodes' routing state tables and identifying any stale entries. Each node must compare the stale entries in its own table with stale entries of neighboring nodes, so that a consensus can be reached on the node's failure. Any remote designation of a node failure remains speculative without a global cross-check of neighbor table entries. Mobility of nodes complicates the detection of node failures, since the lack of packets from a node may indicate that it has moved away rather than failed.

Node Resets. The final node anomaly we examine is node resets. The operator can set a grenade timer to reset the node periodically for operational reasons, which causes the node's packet counter to reset to zero at the same interval. Alternatively, the packet counter may be unexpectedly reset by the node. Since user-specified resets are predictable and periodic, they can be ignored as anomalies. All other resets are classified as node anomalies that indicate a software bug and potential race conditions in the code.

Note that malicious entities may attempt to gain control of sensor nodes and alter their operation. This is an increasingly viable possibility with the advent of the 6LoWPan standard, which implements IPv6 on sensor nodes. Detecting such security attacks remains an open issue as the malicious entity may be able to emulate normal operation of the node. If the attack specifically alters data readings, this may be detectable by statistical methods, which are discussed in the next section.

2.3 Data Anomalies

Data anomaly detection depends on statistical irregularities in the data. These irregularities may be caused by miscalibrated or faulty sensor hardware or environmental variations. Sensor hardware problems are data anomalies rather than

node anomalies because they manifest themselves in erroneous data values rather than the failure or degraded performance of the node. Faulty sensors typically report extreme or unrealistic values that are easily distinguishable. In contrast, environmental variation causes sensor data values to change rapidly, but the sensor readings remain within reasonable ranges. Furthermore, one can distinguish data anomalies by spatial or temporal comparison from several sensors since it is unlikely that many sensors will exhibit a calibration skew or failure at the same time. Security breaches can also lead to anomalous data values from sensor nodes.

There are three broad categories of data anomalies: *temporal* anomaly at a single node location due to changes in data values over time; *spatial* anomaly at a single node location due to comparison with neighboring nodes; and *spatiotemporal* anomaly detected through a number of node location due to changes in data value over time and space.

Temporal Anomalies. Temporal anomalies exhibit one of several symptoms: high variability in subsequent sensor readings; lack of change in sensor readings; gradual reading skews; and out-of-bound readings. High variability in subsequent sensor readings at the same node could signify major changes or events in the sensed environment, or they could arise from sensor voltage fluctuations. In some instances, sensor samples may remain the same over long periods of time, which may indicate a locked state or that the sensor has failed to obtain new samples. Gradual yet sustained skews in sensor readings could indicate a need for sensor recalibration. Finally, out-of-bound readings represent sensor values that are physically not possible, which points towards a major malfunction of the sensor. Detection of temporal data anomalies can take place locally at each node, which requires the node to store historical data, or at a central point. For a detailed review of temporal data anomaly detection methods, see [3].

Spatial Anomalies. Spatial data anomalies can be detected by comparing the values with the surrounding sensors. For example, if the measurement for air temperature on one node differs from measurements of all the surrounding nodes, then it is highly likely that the data is spatially anomalous, and is due to a calibration error in the sensor. This applies to some types of data, such as temperature and humidity, which typically have low spatial variation, but not to data such as audio and video, which differ dramatically at neighboring nodes, depending on the angle and location of signal capture.

Spatiotemporal Anomalies. Spatiotemporal data anomalies combine both spatial and temporal variations, inherently involving more than one node. For example, changes of chemical content in a waterway [14] represent a spatiotemporal anomaly. This will affect nodes along the waterway at different times. Diagnosing this anomaly should account for data throughout the network over a certain period of time. Temporal anomalies can be detected locally at each node, while spatial and spatiotemporal anomalies require more involved inter-node interaction to establish the existence of the anomaly.

2.4 Other Anomalies

Anomalies that do not fit any of the above types are classified as other anomalies. These can be related to back-end software issues, which may cause loss of data or connectivity, without the anomaly residing within the sensor network. Another class of node software-related anomalies may not be explicitly detectable since it does not degrade performance in the short term but causes logical errors in the long-term, for instance due to timer rollovers. Addressing these types of anomalies is currently done offline, by disconnecting the network, fixing the problem, then reactivating the network. Online solutions for these anomalies remain an active research area.

3 Anomaly Detection Strategies

Having examined the main anomaly types in the previous section, we now focus on the strategies for WSN anomaly detection and their design choices. Table 2 shows a set of current representative approaches to anomaly detection. We now briefly survey these approaches to extract the relevant design features and strategies for anomaly detection.

Current approaches vary in their level of development, with some available as fully functional tools and others that are algorithmic. The available tools typically adopt a centralized approach, where a process at the back-end monitors incoming traffic for detecting anomalies. For example, Sympathy [6] is a tool for anomaly detection and debugging in sensor networks. It embeds network-related metrics in packets from all nodes. The sink node observes packets and hypothesizes on the presence and locale of anomalies. Another tool that also uses back-end processing of data appears in [9]. This tool relies on rules at node and data level, such as whether the node is broken or has a bad sensors. It maps these rules to possible root causes of anomalies and provides potential remedial actions to fix the anomaly. Octopus [15] is a sensor network monitoring and visualization tool that provides live topology and link state information on the nodes and enables users to issue simple preset commands to the nodes.

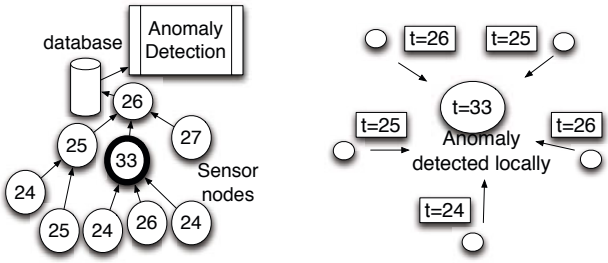
Table 2. Existing Tools

Strategy	Concept	Status	Type	Architecture	Usability
Octopus [15]	topology/state	tool	Network	centralized	visualization
Momento [5]	variance-based	modules	Network	distributed	target FP rate
Sympathy [6]	metrics collection	tool	Network	centralized	specify epoch
Wang et al. [16]	Bayesian	algorithm	Data	centralized	n/a
Rajasegarar et al. [17]	cluster-based	algorithm	Data	hybrid	n/a
Walchli and Braud [7]	prototypes	algorithm	Data	hybrid	n/a
Ramanathan et al. [9]	rules-based	tool	Data/Node	centralized	n/a
Chen et al. [10]	majority voting	algorithm	Data/Node	distributed	unattended
Krishnamachari and Iyengar [8]	Bayesian	algorithm	Data	distributed	n/a
Chang et al. [18]	Recurrent NN	algorithm	Data	centralized	n/a
Obst [19,20]	Recurrent NN	algorithm	Data	distributed	n/a

A separate class of approaches distributes the anomaly detection logic at the nodes. In contrast with tools, these approaches are algorithmic in nature. Momento [5] is a network health monitoring system for sensor networks that relies on a variance-based detector. Its main concept is that the presence of significant changes in node states can indicate the presence of anomalies. It relies on node collaboration on the basis of periodic heartbeats in order to detect deviations from expected behavior. Rajasegarar et al. [17] devise a cluster-based algorithm for detection of data anomalies. Their algorithm distributes the computation of the detection algorithm over all nodes for scalability. However, decisions on whether anomalies have occurred are left for the central sink node, where the results of computations from all nodes converge. A similar computation model is proposed in [7], where sensor nodes classify incoming signals according to learned prototypes, prior to sending the classification to a fusion center for system-wide access patterns.

Distributing the decision process even further, Krishnamachari and Iyengar [8] propose two Bayesian algorithms that run on each node to distinguish between node/data faults and unusual events in the environment. A similar distributed approach that relies on majority voting in a local neighborhood to establish likely ground truths data measurements is proposed in [10]. Decentralized approaches based on recurrent neural networks are presented in [19], which uses only local communication with a few neighbor nodes, and learn a model of sensor values dependent on time series data from their neighbors. It uses online learning by gradually acquiring an improved model of sensors over time.

This brief survey of the existing anomaly detection approaches highlights a set of relevant design choices, including: architecture and target anomaly types. The remainder of section visits these design choices, in addition to examining the usability features of the existing strategies. The degree of distribution has, at least indirectly, ramifications for the usability of the approach. For instance, centralized anomaly detection approaches are easier to handle for a centralized user interface, whereas decentralized approaches can signal faults more easily and more quickly at the local node, using e.g. status LEDs.



(a) Centralized anomaly detection (b) Distributed anomaly detection

Fig. 3. Architectural choices for anomaly detection

3.1 Architecture

Architecture of an anomaly detection strategy refers to the design of the procedure that actually checks for anomalous behavior in the network. There are three options for the architecture of an anomaly detection strategy: (1) centralized; (2) distributed; or (3) hybrid.

Centralized. Figure 3(a) illustrates a simple example of centralized anomaly detection. In this example, each node samples its temperature sensor periodically and sends the temperature data in a packet over its radio towards the central base node, which has a connection to a back-end database. The number on each node represents its current temperature reading. The node in bold reports an anomalous temperature reading which differs significantly relative to all its neighboring nodes. In this centralized approach, detection of this anomalous data reading occurs at a process that monitors the back-end database where all the data converges. This process then localizes the anomaly through the sender ID of the anomalous data packet (in this case node 33), and proceeds to diagnose the anomaly to determine possible root causes, which include miscalibrated sensors or interesting yet highly localized environmental events.

In general, centralized anomaly detection (such as [6,9,16]) relies on analysis of information that converges at the base station or back-end database, as in the example above. Detecting anomalies at the central sensor node itself has the advantage of containing the anomaly detection logic within the network. This avoids any additional interaction between the base station and the back-end system. However, this approach restricts the complexity of queries that can run to detect anomalies, given the limited processing power and storage space at the base station relative to full blown PCs. For instance, most sensor nodes currently have 8- or 16-bit micro-controllers, with typical clock speeds of 16Mhz, 8-10KB of RAM, and up to 1MB of external flash. Storing diagnostic and data packets from tens of nodes over several days can cause the external flash to fill up quickly at the base node. In addition, running multiple queries at this node while forwarding all node packets to the back-end can strain the micro-controller. Recent work towards sensor nodes with 32-bit processors and more RAM may alleviate these issues to some extent.

Alternatively, central anomaly detection can use information at the back-end database, as in Figure 3(a). This design enables far more powerful queries for detecting more involved anomalies, as the database resides on a PC-class machine with relatively high memory and processing resources. Composite queries that examine data packet arrivals over several hours or days become possible. For example, to detect a broadcast storm requires a check of the following:

- A node has attempted to deliver a broadcast data packet, which can result in network flooding
- Multiple retransmissions of this packet have occurred (same sequence number and originator ID), due to lost or delayed acknowledgements
- This behavior is replicated at multiple nodes

The set of queries required to identify these conditions can easily run every few minutes at the database. The same task is not possible at the base station node, mainly because of processing limitations.

Distributed. In distributed anomaly detection [8,10,19], it is up to the sensor nodes themselves to monitor their respective conditions and detect anomalous behavior by themselves or their neighbors. Distributed anomaly detection trades off the resources of centralized detection for quicker and more localized detection. It also exploits spatial correlation of environmental events to distinguish unusual (anomalous) events from faults.

Figure 3(b) illustrates the use of spatial data for distributed anomaly detection. As in the example above, all nodes sample their temperature sensors and send their data periodically. Nodes can snoop on their neighbors' packets and compare them with their own temperature reading. The node in the middle of Figure 3(b) has a local temperature reading of 33°C, while all its neighbors have readings between 24-26°C. As a result, the node in the middle can detect its anomalous data and possibly diagnose this anomaly locally.

Hybrid. Hybrid anomaly detection strategies [5,7,17] try to combine the best of both worlds: the availability of network state information at the back-end of the centralized approach and the responsiveness and low communication overhead of the distributed approach. A hybrid approach uses a centralized strategy for anomalies whose detection metric can be readily examined at the database, such as connectivity anomalies, broadcast storms, and node resets, and complex data anomalies. For any anomaly that can be detected locally at nodes, such as routing loops, link quality changes, variations in solar or battery currents, and threshold-based data anomalies, hybrid approaches place the detection and localization responsibility on the nodes themselves.

3.2 Usability

The usability of a WSN anomaly detection refers to the interaction with human users. Figure 4 provides an overview of user interaction with WSN anomaly detection. Regardless of the detection architecture, an anomaly is reported to a notifier/scheduler process at the base station. This process schedules and sends alerts to users through a service provider network (cellular or Internet) in the form of SMS, email, or web notifications. As explained in the following section, the communication strategy between individual nodes may depend on the way notifications are triggered. A further aspect of usability is the feedback the user provides to the notifier/scheduler process to change how, when, and how often anomaly notifications are sent.

Notifications. We mentioned above the ramifications of the architecture for a user-friendly design of the system. Stand-alone applications are useful for providing an overview of the system status or to report problems on demand. Normally,

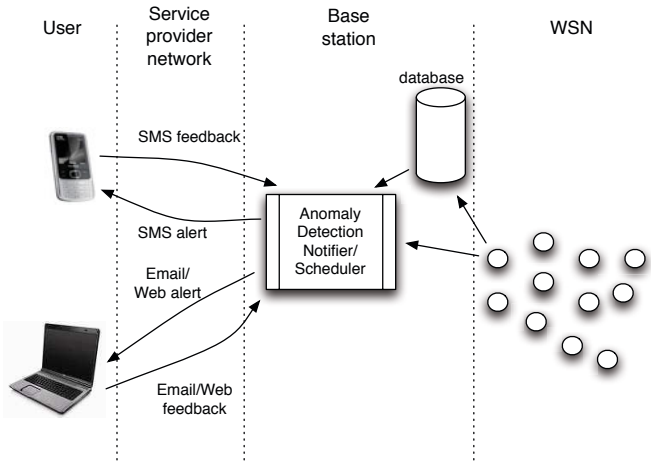


Fig. 4. User interaction with WSN anomaly detection

however, users will want to be automatically notified of exceptional events, and otherwise assume the WSN is working as expected. Thus, the two main methods of remotely notifying users are either by email or by Short Message Service (SMS) on mobile telephones. Specific widgets or applications for smart phones can represent an interesting compromise between the two.

Dependent on the architecture of the anomaly detection approach, sending notifications to users in real time has an impact on the communication strategy of the system. In a centralized architecture, the relevant information is readily available at the base station. In this case, no additional communication between individual nodes is required to notify users. On the other hand, distributed architectures may be able to detect anomalies locally. However, local detection requires nodes to report events to the base station. An alternative design choice is to report events only when queried by the network operator for on-demand operation.

The notification modality is highly related to the length of the message and the speed at which users receive it. SMS notifications are limited to 160 characters, which limits them to short high level descriptions of the anomaly. In contrast, an email notification can be as long as necessary, thus is able to include detailed information about the anomaly, for instance, specific node IDs and diagnosis details. The tradeoff is that people tend to carry their mobile phones all the time, so SMS alerts provide a more timely mode of notification. A final modality for user notification is through a secure website, where interactive maps can provide richer context information for anomalies.

Scheduling. Scheduling involves deciding when and how often to send notifications. This aspect is important because users need to know about faults in the network, however, if a problem persists when it is not fixed straight away, constant notifications of the same problem will eventually cause human operators

to ignore the messages. The simplest solution is to keep a flag or timer in the diagnostic logic to track if and when a message has been sent to a user about a particular anomaly, and to suppress further notifications. Alternatively, the user should be able to configure a set of parameters that indicate how long and how often he/she should receive alerts after an anomaly appears.

Another issue of scheduling is how urgently to send notifications. In other words, if a fault was detected in the network, when should the users be notified? The answer is user-specific, and depends on whether they would want instant notification or after a problem has persisted for a while. Related to this is the scheduling of the diagnostic tool, i.e how often should it check for anomalies in the network? In general, the tool should run at a time interval in the same order as the nodes' data reporting periods. In distributed implementations that run on resource-limited nodes, anomaly check frequency can be reduced accordingly.

User Feedback. The last aspect of usability is user feedback, which involves changing various user-specified parameters, temporary suspension of alerts, and adding new features to notifications. User feedback should be received and processed automatically through emails, SMS or the web interface, allowing for scalability and responsiveness.

The simplest user feedback is the "out of office" reply sent by most email programs when a user is away. During that time, a user might not wish to receive notifications of anomalies in the network. Therefore, the diagnostic tool should stop sending notifications until the user sends another email to resume the notifications. A related feature is for a user to simply send an email or SMS to start/stop receiving notifications. Other feedback mechanisms to the system is changing the various user specified parameters, such as notification scheduling, alert types, and deployments of interest.

In centralized strategies, user feedback simply feeds into the back-end system and adjusts its configuration. In distributed or hybrid architectures, user feedback must propagate to the nodes to alter their frequency of detection or their thresholds for various anomalies. While user feedback can still use the same modalities (email, SMS, web), the back-end can then relay the new settings to individual nodes through Tiny RPC messages. Individual nodes can then respond to such RPC requests by adjusting their local configuration accordingly.

4 Design Guidelines and Conclusions

This section summarizes the guidelines for design of anomaly detection strategies, on the basis of the discussion in previous sections. The first set of guidelines relates to architectural choices for anomaly types.

For *network* anomalies, centralized detection is useful when diagnostic network data can be piggybacked in packets, providing the centralized anomaly detector with a comprehensive view of network state. More complex network anomalies, such as routing loops, are best detected with distributed or hybrid detection, where nodes can snoop on packets that they forward to detect a loop,

or link quality changes, where a node is in the best position to determine the quality of its own links.

For *node* anomalies, centralized, hybrid and distributed approaches are all suitable for detection and localization using the source node id. However, finding the root cause of the node anomaly is more challenging in centralized detection, as the strategy can only hypothesize on the cause on the basis of the data at the back-end. A distributed approach provides more responsive node anomaly detection.

For *data* anomalies, centralized detection is slow relative to distributed detection, which can rely on threshold, spatial, or temporal comparisons run locally at each node. For computationally challenging anomalies, a hybrid approach that locally detects problems then triggers more involved analysis centrally is most suitable.

Usability is another major factor in promoting uptake of WSN anomaly detection tools. Thus, usable tools should provide several modes of notification, including email, SMS, and html, and tailor notification formats to each mode's constraints. Tools should also enable users to flexibly schedule notifications by deciding when, how often, and how urgently notifications are received. Alongside the scheduling feature, several modes of user feedback should be supported to maintain user control of the type and frequency of notifications they receive.

This set of guidelines will hopefully serve as an enabler for further research into the design of more effective and comprehensive anomaly detection strategies.

Acronyms

WSN Wireless Sensor Networks
SMS Short Message Service
tx transmit
rx receive
CTP Collection Tree Protocol
RSSI Received Signal Strength Indicator
ROLL Routing over Low Power Lossy Links
CPU Central Processing Unit
6LoWPAN IPv6 Compression for low power networks
FP False Positive
LED Light Emitting Diode
RPC Remote Procedure Call

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38(4), 393–422 (2002)
2. Subramanian, M.: Network Management: An Introduction to Principles and Practice. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
3. Rajasegarar, S., Leckie, C., Palaniswami, M.: Anomaly detection in wireless sensor networks. Wireless Communications, 34–40 (August 2008)

4. Prokopenko, M., Wang, P., Foreman, M., Valencia, P., Price, D.C., Poulton, G.T.: On connectivity of reconfigurable impact networks in ageless aerospace vehicles. Journal of Robotics and Autonomous Systems 53(1), 36–58 (2005)
5. Rost, S., Balakrishnan, H.: Memento: A Health Monitoring System for Wireless Sensor Networks. In: SECON 2006, Reston, VA, pp. 575–584 (September 2006)
6. Ramanathan, N., Chang, K., Kapur, R., Girod, L., Kohler, E., Estrin, D.: Sympathy for the sensor network debugger. In: SenSys 2005, pp. 255–267. ACM Press, New York (2005)
7. Wälchli, M., Braun, T.: Efficient signal processing and anomaly detection in wireless sensor networks. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 81–86. Springer, Heidelberg (2009)
8. Krishnamachari, B., Iyengar, S.: Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. IEEE T. on Computers 53, 241–250 (2004)
9. Ramanathan, N., Balzano, L., et al.: Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks. UCLA CENS, Tech. Rep. (January 2006)
10. Chen, J., Kher, S., Somani, A.: Distributed fault detection of wireless sensor networks. In: DIWANS 2006, pp. 65–72. ACM, New York (2006)
11. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: The collection tree protocol. In: Sensys, pp. 1–14. ACM, New York (2009)
12. Thubert, P.: Draft ietf roll standard (February 2010), `http://tools.ietf.org/wg/roll/draft-ietf-roll-of0/`
13. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: NSDI 2004: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation, pp. 2–2. USENIX Association, Berkeley (2004)
14. Dunbabin, M., Udy, J., Grinham, A., Bruenig, M.: Continuous monitoring of reservoir water quality: The wivenhoe project. Journal of the Australian Water Association 36, 74–77 (2009)
15. Jurdak, R., Ruzzelli, A., Baribirato, A., Boivineau, S.: Octopus: monitoring, visualization, and control of sensor networks. Wireless Communication and Mobile Computing, 1–21 (2009)
16. Wang, X.R., Lizier, J.T., Obst, O., Prokopenko, M., Wang, P.: Spatiotemporal anomaly detection in gas monitoring sensor networks. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 90–105. Springer, Heidelberg (2008)
17. Rajasegarar, S., Leckie, C., Palaniswami, M., Bezdek, J.C.: Distributed anomaly detection in wireless sensor networks. In: ICCS 2006, pp. 1–5 (October 2006)
18. Chang, M., Terzis, A., Bonnet, P.: Mote-based online anomaly detection using echo state networks. Distributed Computing in Sensor Systems, 72–86 (2009)
19. Obst, O.: Construction and training of a recurrent neural network. Australian Provisional Patent Application 2009902733 (June 2009)
20. Obst, O.: Distributed backpropagation-decorrelation learning. In: NIPS Workshop: Large-Scale Machine Learning: Parallelism and Massive Datasets (2009)
21. Corke, P., Wark, T., Jurdak, R., Hu, W., Valencia, P., Moore, D.: Environmental Wireless Sensor Networks. Proceedings of the IEEE 98(11), 1903–1917 (2010)

Chapter 13
Enterprise Ontologies – Better Models of Business

Ian Bailey

Model Futures Limited
London, United Kingdom

When I first came here, this was all swamp. Everyone said I was daft to build a castle on a swamp, but I built it all the same, just to show them. It sank into the swamp. So I built a second one. That sank into the swamp. So I built a third. That burned down, fell over, then sank into the swamp. But the fourth one stayed up. And that's what you're going to get, Lad, the strongest castle in all of England.

Monty Python and the Holy Grail

Actually, Major Major had been promoted by an I.B.M Machine with a sense of humor almost as keen as his father's.

Catch 22, Joseph Heller

1 Introduction – Intelligence-Led Systems Engineering

The title of this book is "Intelligence-Based Systems Engineering". This presents something of a challenge to systems engineers – it implies that there is a practice of systems engineering that *isn't* intelligence-based. Gulp! What should we call this practice? Stupidity-based systems engineering? Suck-it-and-see systems engineering? Guesswork-based systems engineering? Do-it-like-we-did-it-last-time systems engineering?

Despite the well documented benefits(1) of a systems engineering approach, many projects that use the approach fail, and often fail spectacularly. Symptoms are usually one or more of:

- Cost overruns
- Failure to deliver in time
- Failure to meet requirements
- Failure to deliver something useful, even if it does meet requirements

In fact, if you asked someone to think of ten major engineering projects that have failed from the last 30 years, chances are most of them would have used a systems engineering approach. Does this mean that the discipline of systems engineering is flawed, or is it that each of those projects that failed were not following the approach properly? If you ask the customers for each of those failed projects, they would probably all answer that the engineers didn't properly understand their business requirement. If you ask the engineers, they will nearly all answer that the customer failed to describe their requirements

A. Tolk and L.C. Jain (Eds.): Intelligence-Based Systems Engineering, ISRL 10, pp. 327–342.
springerlink.com

adequately, or that the requirements changed mid-way through the project. One of the main tenets of systems engineering is to avoid this sort of conflict through sensible management of requirements. How did it go so badly wrong for all those major projects? There are any number of causes, and it would require more than one book chapter to go into all of them. There are some causes that this chapter will focus on though:

- It is not uncommon to blame the "information revolution" for the rise in the complexity of systems, and the accompanying escalation in systems development costs. The explosion in computing technology has meant that the interface management between sub-systems has become orders of magnitude more complex than when the discipline of systems engineering first came along. Does this mean that systems engineering isn't fit for purpose in the information age? There is clearly some excellent thinking in systems engineering and in general systems theory – common sense tells us that the approaches espoused by systems engineers are good. It's just that information management is a very different discipline to engineering and requires a different set of strategies to achieve success. Information systems development projects tend to focus on the systems, and make the information itself a secondary concern – engineers retreating into their comfort zone and focussing on functional rather than information requirements.
- User (and often systems) requirements are used as a contract between customer and supplier. The customer issues a contract for some requirements to be satisfied, usually for a fixed price. This works quite well on projects that deliver within a short time-frame (e.g. less than a year). For projects that are going to take many years to deliver, changing circumstances mean that requirements *will* change. If it's an information management system, the requirements can change on a weekly basis. Because systems engineering drives towards a solution of minimum complexity, it can also drive towards a solution of minimum flexibility(2). Customers can't understand how a seemingly minor (to them) change in requirements can cause such wailing and gnashing of teeth amongst the engineers. In a rigidly inflexible system, built to meet the requirements at minimum cost, even the smallest change to those requirements can have a catastrophic impact. It's easy to argue that we should never build systems like this, but when requirements form a contract, and the cheapest solution wins, the cheapest solution will nearly always be inflexible. If you bid with a more flexible (assumed to be more expensive) solution, you won't be selected. Furthermore, the customer is unlikely to build flexibility into their requirement specification out of fear of increased cost and the possibility of not getting what they wanted.
- If used unwisely, a systems engineering approach can, and often does, result in project stove-piping. When a systems engineer budgets out requirements, weight, cost, etc. to various sub-systems, the teams responsible for the delivery of those systems are often only able to innovate within their own sub-system boundary. The allocation of budget also means there is little-or-no financial motivation for pan-system innovation. The result of this is often that no-one is designing and innovating at the system level. The consequence is often that there is rarely a system-wide approach to information.

> Information is managed by each system and exchanged on an ad-hoc basis across systems interfaces. It is very rare to find a project which has an overall information model that all sub-systems work from. A project can have superb interface management procedures, but will still fail if there is no overall design for the information

Information management is a key aspect of each of the above points. It is the contention of this chapter that a systems engineering approach simply does not work well for systems where information management is a key feature. When the systems engineering approach was first proposed, information was much less of a factor in the systems that were being built. The principles of good engineering – designing-out complexity, working to a clear set of fixed requirements, budgeting problems out into sub-systems – simply don't work for information management problems.

1.1 Introduction - Business Ontologies

Having dealt with the title of the book, it's only fair to deal with the title of this chapter. If the term "ontology" is used in an information technology context, it is usually either a reference to data structures defined using semantic web standards, or to data structures used in artificial intelligence projects for purposes of reasoning and inference(3). There is however another use of ontology in IT which is now often overlooked. Several experienced data modellers and business analysts have (often independently) come to the conclusion that the way we currently build information systems is broken(4). To find a better way, they have started to de-construct the current methods, and some have even begun to take a look at what is new in the fields of logic and philosophy(5). They are developing the next generation of information models based on the principles of formal ontology.

1.2 Information System Requirements Gathering

Before looking at business ontologies in detail, it is worth examining how information systems are currently built. The process will vary depending on the scale of the project and the methodology the chief architect was educated in. There are usually some common threads though:

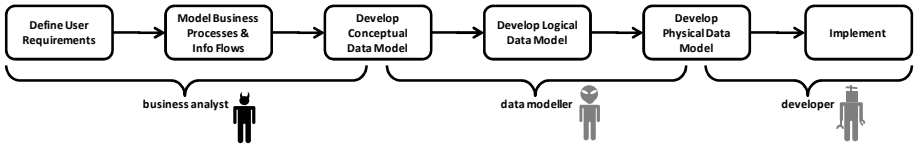


Fig. 1. Typical Development Process for Information Systems

The roles and responsibilities vary from project to project, and there may be more than one person involved at each stage, depending on the size of the project, but this is pretty much the development cycle for the information side of most systems. It seems a fairly sensible approach – separating relatively orthogonal concerns. Someone who understands the business (the business analyst) works with the

stakeholders to formalise their functional and non-functional requirements. The functional requirements are (if you're lucky) formalised as a process model, and the non-functional (in part) as a conceptual model. The conceptual model will be built from user terminology, and the information flows between processes. The conceptual model is then used as the basis for a logical model, which is then used as the basis for an implementation model, which is then implemented.

This looks eminently sensible at first glance, but if it's such a good method why do so many large-scale information systems projects fail, often spectacularly? Let's take a closer look at each of the stages and the people involved.

First of all, there is the business analysis. Does the business analyst really understand the business? Chances are the BA is a consultant and has come from an IT background rather than an operational one. Do process models capture an accurate picture of the business? In most cases, the answer is a resounding no. They are usually developed by people who have scant knowledge of the business they are modelling, and they are either based on observation or user interview. Either approach is flawed. If you observe a business, its behaviour will change, simply because it is being observed. Also, you have no idea if the processes you observe are typical, or unique to the time you observe them. If you interview users, they'll emphasise what is on their mind at the time you interview them (the crocodiles closest to the boat) which may not be important at all next week, or in general for the business as a whole.

OK, so we've got off to a bad start. It seems our business models aren't all they're supposed to be, but surely our data modeller will save the day. Data modellers have all kinds of tricks up their sleeve, surely? The reality is that data modellers are not well understood – to the rest of the development team, they seem to practice the dark arts. Business analysts don't understand what data modellers do, and neither do the implementers. The data modeller is usually just left to get on with their job with no scrutiny from the business or developer community. There is an industry joke that if you give the same requirements spec or process model to ten different data modellers, you'll get eleven different data models. The route from requirement to data model is not governed by a repeatable, scientific method, it is driven by opinion, experience and shallow pattern recognition.

All is not lost at this stage though. Surely those bright young things in development can paper over the cracks with some clever coding? There definitely are some very bright coders out there, but what gets them excited isn't generally anything to do with keeping the customer happy. They like to use cool new technology. They measure success by who can write the most parsimonious code. They treat logical and set-theoretic constructs such as subtyping as implementation hacks to ensure the right properties and methods are inherited conveniently, rather than as any interpretation of business facts and rules.

All this process amounts to is an elaborate and expensive game of Chinese whispers (the British name for a game where a message is whispered along a chain of people to see how distorted the message becomes - other names for the game include "telephone" and "whisper down the lane"). At each stage in the process, the deliverable becomes further and further removed from the business requirement. The sheer insanity of this approach is obvious when discussed in this way, but yet it is still the overwhelming choice of method for developing large scale information systems. Let's not forget, it is always the large scale information systems that seem to fail most

often. The problem is never fixed, because each actor in the process blames the one next to them – and in large scale systems development projects there are always different actors. The data modeller scoffs at the business analyst for his lack of philosophical thought. The business analyst blames implementation when the software fails user acceptance testing. Everyone involved thinks the data modellers are unnecessary, egotistical and not team players. Nobody takes a step back and blames the process. Furthermore, in many projects, the business analysts and data modellers (often consultants or contractors) are long gone by the time the problems are found. The result is that development gets the blame, and a great deal of effort has been put into improving development methods, when in fact the problem lies elsewhere.

1.3 Driving-Out Complexity

In general, a systems engineer will strive for the least complex system that can satisfy the user requirements. There will always be some level of complexity in systems development – the system often needs to be as complex as the problem it is trying to solve – but generally it is considered a good thing to drive-out complexity. The less complex the system is, the easier it is to predict and control. This means it will generally be safer (or at least simpler to produce its safety case and risk assessment) and more reliable.

When it comes to information, the opposite is true. Information systems are required to store and manipulate information that can be immensely complex. Attempts to simplify the information models used in the system will usually result in a reduction in functionality. It is important to make a distinction between complication and complexity. A complex information model is usually characterised by having a large number of interconnected elements that can be put together in different ways for different purposes. A complicated model generally shares the size characteristic, but usually doesn't offer the possibility for re-use of the same elements for different purposes. Complicated models are easier to understand than complex ones, in that they offer one and only one way to structure information, but their size and the number of attributes and relationships required to make them work means that changes to the requirement have expensive consequences. There are plenty of complicated information models around. In fact cynics would argue that most data modellers cook up the most complicated structures possible as some sort of display of intellectual prowess.

The problem gets worse when the data model goes to implementation. Developers will often fail to recognise the importance of certain design characteristics in the information model and implement the model in a simpler fashion in the mistaken belief that their solution is more elegant or efficient. If an information system is to usefully support real world processes, it must have in-built flexibility, and a degree of extensibility and self-reference(6).

1.4 Stovepipes

We've already joked about ten data modellers producing eleven different data models, but what if they're all working on the same project? On a large project, it's not

uncommon for each of the subsystems to have its own data storage capability. Good systems engineering practice should ensure clear interface specifications between the subsystems, but it does not push towards a common information approach across all the sub-systems. In fact, the approach of budgeting out requirements to sub-systems tends to have a negative effect on information coherence.

Often, the same data analysis team will work across the various sub-systems. Even with this approach, it is not uncommon to end up with incoherent data models across the systems. This can be due to a number of issues:

- By far the most common cause is epistemic – when viewing things from different perspectives (e.g. different projects, sub-systems, etc.) information architects fail to recognise when two things are in fact the same. The classic example is customer and supplier. In reality, these are different roles that a person or organisation can play in respect to another person or organisation. However, it is not uncommon (especially in multi-purpose systems such as ERP systems) to see two separate data tables for each of these. When you go and examine the data in them, you will often find the same organisations in both. Elsewhere in the ERP system you may also find a table for organisations, but this will contain another set of organisations – e.g. the departments of the company, regulatory bodies, etc. – as well as some of the organisations in the customer and supplier tables.
- Information modelling is one of the first things to get cut when budgets get tight. Modellers might be asked to do the job as quickly as possible, which prevents any opportunity for reaching consensus across sub-systems. Even worse, the modellers might be cut out of the loop altogether in favour of letting the sub-systems developers do their own data modelling.
- The tendency to develop data models from process models, as outlined in our section on information system requirements gathering, means that the scope for the information model is defined by the scope of the processes relevant to the given sub-system – i.e. there is no requirement to look beyond the sub-system boundary.

The resulting information incoherence just causes problems for the interface managers. Data conversion is required at each interface, and uncertainty about whether each system treats a particular term the same as another causes delays and frustration. Many of the issues around semantic mismatch are not actually found until the systems have been running for some time.

2 What Is Needed for Better Information Systems?

The current approach is clearly broken. Recent history is littered with embarrassing failures of large-scale information systems, and for each one of those failures there are ten that have been declared successful even though they weren't, or simply brushed under the carpet. Very few information systems of an enterprise scale ever deliver on expectations. Read that again. Everyone in the IT industry knows this is true. CIOs know it, and spend most of their working days defending the results rather than working on information strategy. Most CEOs even know it, and regard it as a

necessary evil. The problem is recognised, but not the cause. All attempts to fix the problem have centred around software development methods and architectures (e.g. Agile, SOA, etc.) rather than attempting to fix what is really wrong – the inability of the systems to meet the information requirements. If you buy the arguments in the previous sections of this chapter, then it seems what's needed most is:

1. *An accurate, repeatable and defensible way to ascertain the business information requirements*
2. *A way to build flexibility into our information models without making them over-complicated*
3. *A way to ensure that the same concept is treated the same way by different analysts so that when systems are integrated there is less need for triage at the interfaces.*

2.1 Better Analysis – Getting Your Hands Dirty

The key to achieving point 1) is a reduction in the number of "Chinese whispers" that take place in the analysis stage. Also, the dependence on process analysis needs to be minimised as it is far too open to interpretation. Contrary to popular belief, there are other ways to find out what the business does, but they involve getting your hands dirty. Dirty data is the scourge of IT, and data quality problems are blamed for everything from cost overruns through system crashes to failures in day to day business. This dirty data is an absolute goldmine for the canny information analyst. If analysed intelligently, the data can reveal much more about what the business *actually* does than any process model. The majority of data quality issues are down to simple data-entry errors. However, from experience it is well known – although not often documented – that a lot of what are called data quality problems are actually workarounds instituted by users who needed somewhere to store crucial business information. These are not failings in data, they are symptoms of a failure to develop an information system that meets the user requirements. Examining this sort of data, with the appropriate analysis techniques, can result in a much better information model than one arrived at through process analysis.

If we can also cut the layers of information / data model from three (conceptual, logical, physical) to one, we stand a much better chance of ensuring that what gets implemented genuinely meets the business requirement. To facilitate this, we need to look at why there ever were three (or sometimes two) layers of data model. In a nutshell, the reason is that database technologies don't store information in the same way that humans tend to think about things in the real world. Hence conceptual models are used to capture concepts and their relations as a first step. The relations can then be rationalised, and attributes added in order to ensure that there is a logically consistent model. Finally, the logical model is bashed and bent to fit whatever underlying storage technology is going to be used. This was a hot topic of research in the 1960s and 70s. Bill Kent (4) highlights the problem: "For some time now my work has concerned the representation of information in computers. The work has involved such things as file organizations, indexes, hierarchical structures, network structures, relational models, and so on. After a while it dawned on me that these are all just maps, being poor artificial approximations of some real underlying

terrain." Griethuysen's approach (17) is one of the early 'ostriches', suggesting that there is a simple direct link between the data structures and reality.

What if the underlying storage technology worked the same way as the logical and conceptual models? Then we could have just one model. That, combined with a more rational and defensible analysis technique (based on forensics, not witchcraft), should help ensure we implement something that aligns much more closely with the business.

That last paragraph has left a big question open – data storage technology. Let's return to that point once we've had a look at 2) and 3).

2.2 Flexibility – Using the Full Range of Logic

The key to achieving a more flexible information management system is in understanding which relationships in the information model constrain its use, and which give it more axes of movement. This is the essential difference between complexity and complication. Characteristics of a flexible model are:

- Subtypes – i.e. the hierarchy of specialisation of concepts. Flexible models often have the characteristic of extensive subtyping, usually from one top-most class. This allows the frequently used model "infrastructure" to be moved up the specialisation tree and simply inherited by more specialised concepts below. This leads to re-use of implementation patterns resulting in significant savings in code production and maintenance. The problem with this is that if the subtyping isn't based on defensible criteria, the inheritance doesn't work. A new analysis method is required to ensure inheritance works well and accurately mirrors the real world concepts being represented. The classic example discussed by datamodellers is the Circle – Ellipse problem(7). Which is a sub-type of which depends on the definition of inheritance.
- Higher Order – flexible models tend to be able to handle their own classifications. As an example, instead of committing a set of common equipment types as classes (e.g. "pump", "valve", "engine", etc.), a flexible model will allow the users to introduce a class called equipment, and a related class called "equipment type" – though the model may give the users a 'starter pack'. This allows the user to manage the types of equipment at run time as opposed to being fixed in implementation. It sounds obvious for things like equipment, but the requirement for this is less obvious for other data elements such as transactions, and the opportunity to build in flexibility is often missed.
- Names – most systems commit the users to work with only one set of names for things. More often than not, developers will commit this to code by using these names as primary keys in databases. What happens then is that when other communities are offered the chance to use the system, they don't, because the system doesn't use their terminology. The system has effectively stovepiped itself within one business vertical. One could imagine a more flexible system that could cope with multiple communities and multiple names, and occasionally (usually in very specialised situations), these systems do turn up.

These tenets of flexibility run counter to traditional systems engineering and software development. Software developers use subtyping as a convenient way to inherit properties and methods, and pay little or no regard as to the real-world relevance of their assumptions. Higher order systems are rare, and tend only to be used in very specialised circumstances. Systems that can cope with unbounded multiple names are rarer still. Systems that can achieve all three aspects of flexibility are all but impossible to find.

Again, this leaves a dangling question about data storage technology and performance – most developers would shy away from these approaches in the belief that performance and user interfaces would be detrimentally affected. Let's come back to that.

2.3 Consistency – Sophisticated, Repeatable Analysis

Point 3) is the holy grail of information analysis. The goal is that no matter who the analyst is, given the same real world concepts, we will always get the same information model. In practice though, the model is always tainted by the user's view of the world, the analyst's view of the world and the methodology used. What if we had a methodology that allowed us to cut through opinion and only deal with fact? Even if such a methodology could get us to the point where 80% of concepts are dealt with in the same way, it would offer enormous potential to businesses.

2.4 Implementation – New Ways of Storing

Now let's get back to implementation – particularly the thorny issue of storage. The relational database rules the roost at the moment, and has done for over 20 years. Its dominance is similar to that of the internal combustion engine – a lot of people think there are probably better technologies out there, but so much has been invested into making the incumbent technology perform that the bar to market entry is very high indeed. In recent years, the relational database has become part of the infrastructure – developers only ever care about physical data models these days if transactional performance is paramount or the queries to be used are complex (e.g. in data warehouses and BI/MI systems). Object-relational projections are now taken for granted in IT. There is no reason why the ideas about accuracy, flexibility and cross-community support outlined above can't also be dealt with in currently available database systems. In fact, in the case of flexibility, a lot of work has already been done. The adaptive object model approach (8)(9) has gone some significant way towards achieving this, but what is often overlooked is the work that went on in Shell in the early nineties on developing flexible data storage systems. This work has now been realised as a successful commercial product – Kalido™ (10).

Furthermore, the *No SQL* movement is producing large numbers of novel data storage approaches, each of which is usually suited to specific types of application. There is great potential for using these technologies to store next generation information structures.

3 A New Approach to Information Systems Development

It's all been bad news up to this point. If this chapter has done what it was intended to do, you will have either thrown the book on the fire in disgust (if you're a systems engineer), or be in floods of tears (if you're a customer). The real question is whether you agree with the points made. Few people would argue that there is *something* wrong in information systems development, but do you agree that the problems lay in the analysis approach? The other argument you need to buy is that to fix the problem we need the three points outlined before:

1. *An accurate, repeatable and defensible way to ascertain the business information requirements*
2. *A way to build flexibility into our information models without making them over-complicated*
3. *A way to ensure that the same concept is treated the same way by different analysts so that when systems are integrated there is less need for triage at the interfaces.*

3.1 Introducing the BORO Method

If not, then it's probably best you don't read on. If you agree with some or all of the points, then there's a different approach to analysis that you might like to consider. The BORO Method(5) is a forensic approach to re-engineering information systems. It relies on there being legacy data. The older, dirtier and more convoluted this data is, the better. Even so-called green-field IT projects are replacing some form of information system, even if it's paper-based. At the start of any information systems delivery project, there *will* be some legacy data. Traditional approaches tend to ignore this in favour of process modelling – the argument being that the new system has to reflect the new processes. To a certain extent, this is true. The *behaviour* of the system certainly does have to reflect this, but the information is unlikely to have changed much from before.

The premise of BORO is that dirty data is a symptom of an information system that doesn't properly support its users. If you look at the data (not the data model) then you will find out what is really needed to support the business. The challenge is how to get from this data to a specification for a new information system. This is where BORO delivers on point 3). The methodology is designed to be simple, precise and repeatable. Hence, if done properly, two different BORO analyses of the same data (even conducted by different people) should result in broadly similar (sometimes even identical) information models.

The BORO method results in what are called *extensional* ontologies(10). This means that the elements specified in the ontology are not primarily identified by their names, they are identified by their extents. [*Technically, the ontologies have extensional criteria of identity. Often data models have no criteria for identity, so no mechanism for consistently generating agreement about what is being modelled.*

Clearly criteria of identity are useful – what is more difficult is devising suitable criteria. For those interested in criteria of identity, an extensional criteria approach is one of the few that meets the requirement.] For individuals, this means their spatio-temporal extent – two things are the same if they occupy the same space for the same time. For types of things (classes), two classes are the same if they have the same members. Although this looks very simplistic, it is reliable and repeatable. It ensures that the same thing doesn't end up in the ontology twice, and that different things don't get mistaken as the same simply because they have the same or similar names. It's also extremely counter-intuitive (especially for computer scientists, it seems), but once you've used the approach long enough, everything else looks primitive, woolly and imprecise.

Once you've identified the things you're dealing with, it is of course useful (and often necessary) to give them a name. The BORO approach separates "name-space" and "object-space". There are things, and there are the names of things. Each name has a context, so there could be a set of names used by one community and another set used by a different community. For a given thing (in object-space) each community might use different names, and BORO allows this. Although the use of multiple names runs counter to the systems engineer's drive for simplicity, it does what the users want. Those communities have probably developed their own terminology for good reason – let them carry on using it. The IT should not dictate terminology to the business.

The ontologies BORO produces allow new classes to be added. This flexibility, combined with the precision of the analysis technique allows applications to be created that are both semantically precise *and* flexible.

3.2 Managing Time

As well as the benefits outlined above, BORO also gives the analyst a new way to model time. The handling of temporal information in data models is at best inconsistent and often frankly incoherent. BORO offers a better alternative.

The requirement that BORO places on the analyst to think in terms of extension has a number of interesting consequences. First of all, if you have to consider identity of individual things (buildings, countries, people, etc.) in terms of their physical extent, you are forced to also consider their temporal extent. Two different things may occupy the same space at different times, so the criteria for identity also has to consider the temporal dimension. Philosophers call this "four-dimensionalism". It has its origins in physics, mathematics(11) and philosophy(12) from the early twentieth century, with mainstream philosophy catching up about 90 years later(10)(13)(14). Theoretical and philosophical subjects tend to be ignored by engineers (the author of this chapter included). However it turns out that for information systems, the 4D approach makes life very simple indeed. Data models usually end up having to deal with time in different ways – the time things happen, the duration of things, the time between occurrences, times of the day, days of the week, etc. The 4D approach

provides a single, consistent approach to time that covers all the bases a data modeller would ever need – compare this with relatively byzantine structure of, for example, (19). It also provides a very sophisticated way to model how things *change* over time. A whole book could be devoted to the things you can do with a 4D approach – see Partridge(5) and Hawley(14) to get you started – but it's worth spending a bit of time on it here.

If you use a 4D approach, the first thing you have to get used to is that other dimension. Let's get the criticism out of the way first. By now, a few readers are going to be thinking this is all theoretical nonsense and of no practical use. Terms like "extensional ontology" and "four-dimensionalism" are hard to get past the CIO, and will probably send the CTO scuttling back to his cave to read Dr Dobb's journal. The ideas behind it are sound, practical and useable, however. In terms of time, the first thing to get used to is using whole-part relationships – technically called "mereology" (15) – in a temporal setting. We are all familiar with the concept of composition – one thing being part of another. In a 4D ontology though, the individual also has a temporal extent. This allows us to express quite complex situations in a very simple way. If we want to take about cars coming and going in a car park, or in a particular parking place, all we need to do is work out the composition. The example below is a "space-time map"; three dimensions (x,y and z) are compressed and shown on the vertical axis, with time on the horizontal:

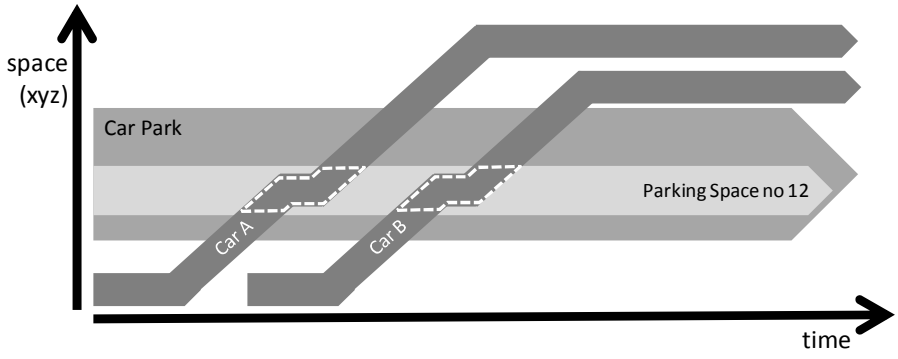


Fig. 2. Space-time map for car parking

Although these diagrams don't allow us to be precise about space, they do help enormously in visualising how things change over time. In the simple example above, we can see immediately that cars A and B come pull into the car-park, park in space number 12 and then leave. In 4D, a (temporal) part of the car (shown by the dotted line) is part of the parking space. The parking space is part of the car park. If we want to be more specific:

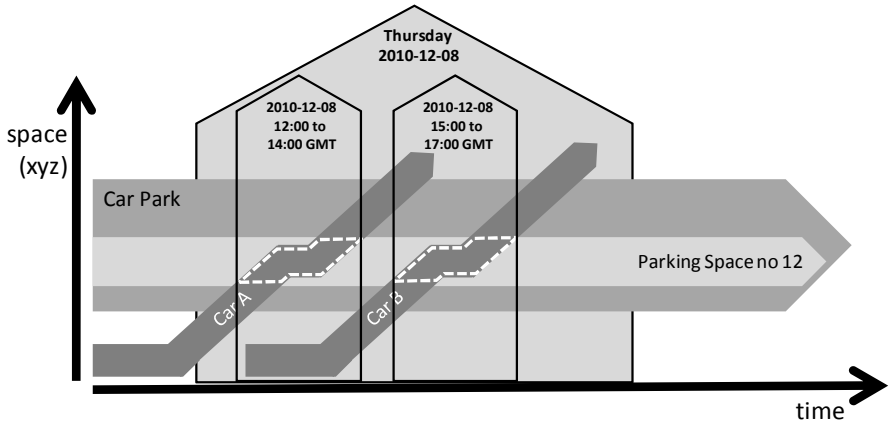


Fig. 3. Space-time map with periods overlaid

In this case, we are stating that the temporal part of Car A while it was in space 12 is part of the period of time 12:00-14:00. Again, all that has been used here is mereology. Although this is a little counter-intuitive, it is consistent. Traditional data models might be able to express what is shown above more concisely – e.g. it might store a set of key pairs for the parking space and the car registration, along with the begin and end times:

Space	Car	In	Out
12	A123ABC	2010-12-08-12:00	2010-12-08-14:00
12	B234XYZ	2010-12-08-15:00	2010-12-08-17:00
12	etc.		

Fig. 4. Flat-earth data modelling

The problem with this is that it is data *about* the car and car park. The data model for this is not an ontology, because it doesn't model the real world, it models some specific data about the real world. The opportunity for re-use of this data is minimal at best, and the opportunity for re-use of the data model is probably zero. A system built around a model like this can do one thing and one thing only – track parking times. A system built around the more sophisticated 4D ontology understands about cars, car parks, days, times and parking spaces. Furthermore, the simple mereological patterns used to deal with this are also equally applicable to the car-key store in the attendant's hut, the car-park ventilation equipment tag numbers, and the clothes the attendant wears. This, in a nutshell, is why it's worth going to the trouble of doing the analysis properly using precise, repeatable method like BORO.

4 Addressing Arguments against Ontology

The counter-argument to formal ontology is usually that it is complex, hard to implement and will never perform. Let's look at those points one by one.

Firstly, yes a formal ontology is not simple. It is complex, and that is a good thing. However, an ontology need not be *complicated*. There are some horrendously complicated relational data models (ER) out in the wild. They are rarely flexible or extensible, simply because they *are* complicated. They are hard to maintain, and very costly to change once up and running. Ontologies suffer from none of these problems and only seem complex to people who are not familiar with formal logic and set theory. That said, something strange starts to happen when these approaches are adopted. Code starts to shrink. This has been the experience of a number of projects now. People who have used formal patterns such as those defined by the Gang of Four(16) or Martin Fowler(8) have experienced improvements in code acreage. Those that have gone a step further and started to use formal semantics and ontologies are seeing even further gains. Most of these seem to stem from the establishment of even more general patterns – for example the re-use of whole-part composition for temporal matters covered in the previous section.

In terms of being hard to implement, this is an education issue. For a systems engineer who is steeped in the traditional analysis and development methods then yes, ontology is going to be difficult. For organisations that are entrenched(17)(18) in these traditional methods, then it's going to be impossible other than by acquisition of smaller, more agile companies. If you don't carry all that baggage with you, or you're prepared to un-learn what you know then it's actually not that bad. Ontology has been used a lot by the artificial intelligence community, and this has tended to add to its reputation for complexity and opacity. Ontology has utility outside of AI, and implementations (even in relational databases) can be relatively straightforward. Where it *can* get difficult though is in the user interface. When the underlying data model can change without recourse to software re-builds, the user interface needs to be adaptive enough to cope with these changes. This either means building data-driven interfaces, using a service-oriented approach with fine-grain presentation services, or being prepared to change the UI code regularly...none of which are show-stoppers. The same goes for middleware in n-tier implementations.

Performance is another old chestnut. One of the major users of ontologies over the last couple of decades has been the artificial intelligence community. Reasoners and inference engines gained a reputation for poor performance and this seems to have stuck. Although there has been some good work in AI, some of the reasoners can spend days churning through data, only to present an answer that shows all the deductive skills of a Labrador puppy. They may well have achieved a great logical feat, but to most IT professionals it is apparent that the same answer could have been arrived at with a couple of queries in a couple of seconds. Ontology doesn't have to be an academic pursuit though. Ontology storage systems can perform well and scale massively. Even triple stores are starting to perform well – a company called Garlik has developed a system called 5Store that they use to manage vast quantities of data with transaction rates of 700,000 triples per second.

5 Conclusion

The theme of this chapter is that traditional analysis techniques rarely go beneath the surface appearance of process models and information flows to find out what is actually going on. This means that the systems that are developed are often not sophisticated enough to cope the problem they were supposed to solve, and are rarely able to extend their functionality or adapt to changing business environments without huge disruption and cost.

If the analysis method used is more rigorous, defensible and repeatable, the information models produced will be more robust. If the models produced are flexible, extensible and closely follow the real world (instead of data about the real world) then the systems produced are more agile. An ontology approach (such as that provided by the BORO method) has the potential to provide the flexibility *and* the accurate real-world modelling.

5.1 Literature Search

As you've probably noticed by now, this chapter contains a lot opinion and observation from a practitioner's point of view. It is the result of two decades of dealing with systems engineers who think interoperability is simply a case of having a network. In many cases, the references have been retro-fitted in order to justify some of the more controversial points. But, I would hope that most of the points are *painfully* familiar to anyone who has worked on a large-scale systems.

The process of going through these references was a revelation though. To my horror, I discovered that the problems described in this chapter have been around since the 60s (20), and the 4D ontology approach that seems to solve so many problems in information modelling has its roots in early twentieth century mathematics (12) and philosophy (13). None of this is new.

Acknowledgements

Thanks go to Chris Partridge for helping with some of the references – especially the Mealy one (20) – and reviewing the chapter (even though he described my writing style as "shock jock"), and to Dr Graham Bleakley of IBM for reviewing and sanity-checking.

References

1. Honour, E.C.: Understanding the Value of Systems Engineering. INCOSE, Pensacola (2004)
2. Bar-Yam, Y.: When Systems Engineering Fails — Toward Complex Systems Engineering. IEEE, Cambridge (2003)
3. Gruber, T.R.: A translation approach to portable ontologies. Knowledge Acquisition 5(2), 199–220 (1993)
4. Kent, W.: Data And Reality: Basic Assumptions in Data Processing Reconsidered (1978)

5. Partridge, C.: Business Objects: Re-Engineering for Re-Use, 2nd edn. BORO Centre, London (2005) ISBN 0-9550603-0-3
6. Barwise, J.: Vicious Circles: On the Mathematics of Non-Wellfounded Phenomena. Cambridge University Press, Cambridge (1996) 978-1575860084
7. Henney, K.: From Mechanism to Method: Total Ellipse. Dr Dobbs, s.l. (2001)
8. Fowler, M.: Analysis Patterns, Reusable Object Models. Addison-Wesley, s.l. (1997) ISBN 978-0201895421
9. Yoder, J.W., Balaguer, F., Johnson, R.: Architecture and Design of Adaptive Object Models. In: OOPSLA 2001 (2001)
10. Heller, M.: The Ontology of Physical Objects. Press Syndicate of the University of Cambridge, Cambridge (1990) 0-521-38544
11. Minkowski, H.: Raum & Zeit (Space & Time). In: Einstein, A., Minkowski, H., Lorentz, H.A. (eds.) Das Relativitätsprinzip. Eine. Teubner-Verlag, Leipzig (1915)
12. McTaggart, J.M.E.: The Unreality of Time. Mind 17, 457–474 (1908)
13. Sider, T.: Four-Dimensionalism: An Ontology of Persistence and Time. Clarendon Press, s.l. (2003) 978-0199263523
14. Hawley, K.: How Things Persist. Clarendon Press, s.l. (2004) 978-0199275434
15. Simons, P.: Parts: A Study in Ontology. Clarendon Press, s.l. (2000) 978-0199241460
16. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison Wesley, s.l. (1994) 978-0201633610
17. Wimsatt, W.: Generative Entrenchment and the scaffolding of individual development and social institutions. In: International Society for History, Philosophy, and Social Studies of Biology, ISHPSSB (2005)
18. Wimsatt, W.: Re-Engineering Philosophy for Limited Beings. Harvard University Press, s.l.; 978-0674015456. 978-0674015456
19. Griethuysen, J.v.: ISO/TC97/SC5/WG3-N695 - Concepts and Terminology for the Conceptual Schema and the Information Base. ANSI, New York (1982)
20. Date, C.J., Darwen, H., Lorentzos, N.: Temporal Data & the Relational Model. Morgan Kaufmann, s.l. (2002)
21. Mealy, G.H.: Another Look at Data. In: Proceedings of the Fall Joint Computer Conference, Anaheim, CA, November 14-16, p. 565 (1967)

Author Index

Adams, Kevin MacG. 1
Al-Zoubi, Khaldoon 129

Bailey, Ian 327
Bandar, Zuhair 201
Blanco-Fernández, Yolanda 285

Crockett, Keeley 201

Diallo, Saikou Y. 49, 259

Fumarola, Michele 107

Hunt, C. Anthony 233

Jannach, Dietmar 75
Jurdak, Raja 309

Keating, Charles B. 1

López-Nores, Martín 285

Mittal, Saurabh 159

Nagy, Miklos 75

Obst, Oliver 309
O'Shea, James 201

Padilla, Jose J. 259
Pazos-Arias, José J. 285

Rosalind Wang, X. 309

Seck, Mamadou D. 107
Sousa-Poza, Andres A. 259
Sowa, John F. 23, 49
Szabo, Claudia 49

Tolk, Andreas 1

Valencia, Philip 309
Vargas-Vera, Maria 75
Verbraeck, Alexander 107

Wainer, Gabriel 129

Yilmaz, Levent 233