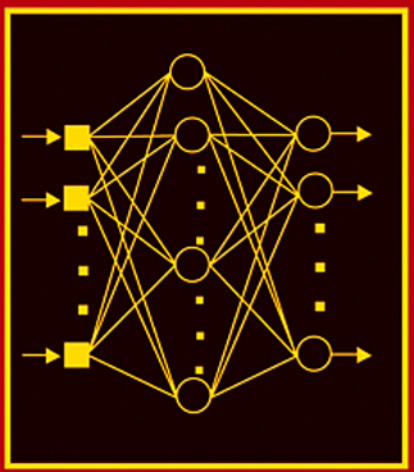
Advances in

COMPUTERS

Volume 70



Edited by

MARVIN V. ZELKOWITZ

Advances in

COMPUTERS

VOLUME 70

This page intentionally left blank

Advances in
COMPUTERS

EDITED BY

MARVIN V. ZELKOWITZ

Department of Computer Science
and Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland

VOLUME 70

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON • NEW YORK • OXFORD
PARIS • SAN DIEGO • SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO
Academic Press is an imprint of Elsevier

ACADEMIC
PRESS

Academic Press is an imprint of Elsevier
84 Theobald's Road, London WC1X 8RR, UK
Radarweg 29, PO Box 211, 1000 AE Amsterdam, The Netherlands
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
525 B Street, Suite 1900, San Diego, CA 92101-4495, USA

First edition 2007

ISBN-13: 978-0-12-373747-2

ISSN: 0065-2458

For information on all Academic Press publications visit our website at books.elsevier.com

Printed and bound in USA

07 08 09 10 11 10 9 8 7 6 5 4 3 2 1

Contents

A Tour of Language Customization Concepts

Colin Atkinson and Thomas Kühne

Advances in Business Transformation Technologies

Juhnyoung Lee

Phish Phactors: Offensive and Defensive Strategies

Hal Berghel, James Carpinter, and Ju-Yeon Jo

Reflections on System Trustworthiness

Peter G. Neumann

This page intentionally left blank

Contributors

Prof. Colin Atkinson heads the chair of Software Engineering at the University of Mannheim in Germany. Prior to that he was an associate professor at the University of Kaiserslautern and project leader at the affiliated Fraunhofer Institute for Experimental Software Engineering. From 1991 until 1997 he was an Assistant Professor of Software Engineering at the University of Houston – Clear Lake. His research interests are focused on the use of model-driven and component based approaches in the development of dependable computing systems. He received a PhD and MSc in computer science from Imperial College, London, in 1990 and 1985 respectively, and his BSc in Mathematical Physics from the University of Nottingham in 1983.

Prof. John Atkinson is an associate professor in the Department of Computer Science at the Universidad de Concepcion, Chile. His research topics focus on Natural-Language Processing, Text Mining and Intelligent Agents. Atkinson received a PhD in Artificial Intelligence from the University of Edinburgh, Scotland, UK. He is a member of the American Association for Artificial Intelligence (AAAI), Association for Computer Machinery (ACM), and the IEEE Computer Society. Dr. Atkinson is also Vice President of the Chilean Computer Science Society. Contact him at atkinson@inf.udec.cl.

Prof. Hal Berghel is currently Associate Dean of the Howard R. Hughes College of Engineering at the University of Nevada, Las Vegas and founding Director of both the Center for Cybermedia Research and Identity Theft and Financial Fraud Research and Operations Center. He has held a variety of research and administrative positions in industry and academia during his twenty-five year career in computing. His current research focuses on computing and network security and forensics, digital crime, and technologies to anticipate network security events-of-interest. He is also a popular columnist, author, keynote speaker and frequent talk show guest in the network and security areas. He is a Fellow of both the IEEE Computer Society and Association for Computing Machinery, has been both an ACM Distinguished Lecturer and an IEEE Computer Society Distinguished Visitor several times over the past two decades. He has received the ACM Outstanding Lecturer of the Year

Award four times and was recognized for Lifetime Achievement in 2004. He has also received the ACM Outstanding Contribution and Distinguished Service awards. He is also the founder and owner of Berghel.Net, a full-service information, computing and security consultancy. He is also Founding Director of UNLV School of Informatics.

James Carpinter completed his honours degree in computer science at the University of Canterbury in 2005 and now works as a software engineer for Unisys New Zealand. His research interests include machine learning and internet security.

Prof. Anita Ferreira is an associate professor in the Department of Linguistics (Spanish) at the Universidad de Concepcion, Chile. Her main research topics include Natural-Language Processing, Intelligent Tutorial Systems, and Computer-Assisted Language Learning. Ferreira received a PhD in Artificial Intelligence from the University of Edinburgh, Scotland, UK. She also received a PhD in Linguistics from Catholic University of Valparaiso, Chile. She is a member of CALICO and EUROCALL. Contact her at aferreir@udec.cl.

Prof. Ju-Yeon Jo received the PhD degree in Computer Science from Case Western Reserve University, Cleveland, Ohio. She is an assistant professor in the School of Informatics at the University of Nevada, Las Vegas, which she joined in August 2006. From 2003 to 2006, she was an assistant professor of computer science department at California State University, Sacramento. Prior to that, she spent several years in the communication, networking, and software industry. She was a member of the technical staff at Lucent Technologies, Bell Labs, in Holmdel, New Jersey, and a software engineer at Coree Networks, a New Jersey based start-up company. Her current research interests include information security, network security, networking protocol design and performance analysis, and Internet traffic characterization.

Prof. Thomas Kühne is an Assistant Professor at the Darmstadt University of Technology in Germany. Prior to that he was an Acting Professor at the University of Mannheim (Germany) and a Lecturer at Staffordshire University (UK). His interests are centered on object technology, programming language design, model-driven development, component architectures, and metamodeling. He received a PhD and MSc from the Darmstadt University of Technology, Germany in 1998 and 1992, respectively.

Dr. Juhnyoung Lee is a Research Staff Member at the IBM T. J. Watson Research Center in New York. He is currently working in the Business Informatics group. He finished his PhD at the Department of Computer Science in the University of Virginia

at Charlottesville in 1994. He received his BS and MS in Computer Science from Seoul National University in 1985 and 1987, respectively. Since joining IBM Research in 1997, he has worked on e-commerce intelligence, electronic marketplaces, decision support systems, semantic Web technologies, and ontology management systems. Before joining IBM, he was a researcher at Los Alamos National Lab in New Mexico and at Lexis-Nexis in Ohio. His current research interests include cost and value estimation for service engineering and management, business and IT modeling, model-driven business transformation, and semantic Web.

Dr. Peter Neumann is Principal Scientist in the Principled Systems Group of the Computer Science Laboratory at SRI International, where he has been since 1971. He was at Bell Telephone Laboratories in Murray Hill, NJ throughout the 1960s, participating in computer-communication system research and in the design and development of Multics. He has doctorates from Harvard and Darmstadt, and taught courses at Darmstadt, Stanford, U.C. Berkeley, and the University of Maryland. His website (http://www.csl.sri.com/neumann) contains papers, reports, and federal/state testimonies relating to trustworthy computer systems, networks, security, reliability, survivability, safety, voting-system integrity, cryptographic policy, social implications, and privacy. He is a Fellow of the ACM, IEEE, and AAAS, and is also an SRI Fellow. He received the National Computer System Security Award in 2002 and the ACM SIGSAC Outstanding Contributions Award in 2005.

Dr. Charles M. Patton, a mathematician and senior researcher at SRI International, focuses on enabling technologies for mathematics teaching, learning, and exploration. Prior to joining SRI, Dr. Patton pioneered the concept of handheld symbolic/graphical/numeric computation, coauthored an NSF-sponsored calculus reform text widely used in AP Calculus, and initiated the idea of a wireless networked classroom. At SRI, Dr. Patton's projects include incorporating mathematics into curricula via student engagement with data, designing cell-phone-based supports for students with learning disabilities and tuplespace-based systems for social coordination, and creating the Group Scribbles collaborative learning system. Dr. Patton received a PhD in mathematics from SUNY Stony Brook, the first American Mathematical Society Post-Doctoral Research Fellowship, and a membership at the Institute for Advanced Study in Princeton.

Dr. Jeremy Roschelle, Director of the Center for Technology in Learning at SRI International, conducts research examining the design and classroom use of innovations that enhance learning of complex and conceptually difficult ideas in mathematics and science. Through cognitive science-based research on the "Envisioning Machine" and later "SimCalc," he has explored how computer-based representations

can make the mathematics of change and the related physics of motion accessible to many more students. Two running themes in his work are the study of collaboration in learning and the appropriate use of advanced or emerging technologies in education.

Prof. Deborah Tatar is currently an Associate Professor of Computer Science and, by courtesy, Psychology at Virginia Tech. Her work falls into three categories: Making Mechanisms (designing new ways to do things with technology), Making Meaning (analyzing complex new systems), and Making Methods (creating new ways of coming to know about phenomena of interest). Recent mechanisms explore the potential of handheld connectivity to help classroom learning. Recent meaning includes the analysis of mediated argumentation, unlocking the concept of "place" in design, and exploring the scaling up educational technology to real world conditions. Recent mechanisms focus on Design Tensions as an analytic frame, and sophisticated conceptualization of the Third Paradigm in HCI. She received the PhD degree in Psychology from Stanford University.

Preface

This is volume 70 in the **Advances in Computers** series of books. This series that began in 1960 is the longest continuing series of volumes that covers developments in the information technology field. Annually we present three volumes that offer 18 to 20 chapters that describe the latest technology in the use of computers today. This present volume contains six chapters covering a wide variety of topics in the ever-changing landscape of information technology.

New technology has often promised to revolutionize the way education is delivered to students, but just as often has failed to live up to that promise. Computers in the classroom were touted as ways to vastly improve the delivery of instruction to children, but such machines have had little impact to date. But the situation may be changing with the advent of new inexpensive powerful ubiquitous handheld devices, connected wirelessly to the Internet. In Chapter 1, "Designing Networked Handheld Devices to Enhance School Learning," Jeremy Roschelle, Charles Patton, and Deborah Tatar explore this new technology and present details of new hardware and software developments which should have a major impact on the public school curriculum in the years to come.

In Chapter 2, "Interactive Explanatory and Descriptive Natural-Language Based Dialogue for Intelligent Information Filtering" by John Atkinson and Anita Ferreira, the authors address Internet web searching. What has made the Internet so important today is the ability to find websites containing the information you need. This is made possible by the many search engines, where you type in a query, and they return the set of web pages, out of the billions they have indexed, that probably contains the information you need. For example, I just typed "web searching" into one such search engine, and it responded saying there were about 842,000 web pages that were relevant. How can I intelligently narrow this search in order to reduce this number to a more meaningful number? That is the theme of this present chapter, which uses a natural language dialogue to filter the information.

"A Tour of Language Customization Concepts" by Colin Atkinson and Thomas Kühne is the title of Chapter 3. Computer languages have been classified as general purpose or domain specific. A language like FORTRAN or C++ is considered general purpose since it can be used to solve a wide variety of problems. But we can also extend the design of a language with domain specific features, a *domain-customized*

language, in order to solve specific problems. As the authors state "Typically, the base language will be a widely-known standard and the amount of customization will be minimized in the sense that as much as possible of the base language will be left unaltered." This chapter explores the world of such domain customized languages.

In Chapter 4, Juhnyoung Lee discusses "Advances in Business Transformation Technologies." Similar to the domain customized languages of Chapter 3, this chapter discusses domain specific languages in the realm of business applications. The general approach of this chapter, as stated by the author, is "Business transformation is a key executive management initiative that attempts to align the technology initiatives of a company closely with its business strategy and vision, and is achieved through efforts from both the business and IT sides of the company." The author presents a series of technologies that implements this vision.

The last two chapters unfortunately present technology that was not foreseen during the formative years of the computer, some 50 to 60 years ago. Early computer pioneers were mostly altruistic and the needs for security in such systems was not an important requirement. Today we are all too aware of hackers, email viruses, spyware, and other malicious software invading our cyberspace. In Chapter 5, "Phish Phactors: Offensive and Defensive Strategies" by Hal Berghel, James Carpinter, and Ju-Yeon Jo, the authors discuss the relatively recent phenomenon of *Phishing*, or the masquerading by an illegal website to look like a valid website in order to get the user to reveal personal information, such as passwords, social security numbers, and other private information. Various phishing methods and protection against them is the theme of this chapter.

In the final chapter by Peter G. Neumann "Reflections on Systems Trustworthiness" (Chapter 6), Dr. Neumann presents an overview of his many years of collecting information on computer-based risks. How has the computer been misused and what protections should we be placing on such software and hardware to ensure that the issues of the previous chapter, as well as many other problems, do not occur? What makes a system untrustworthy and what can we do to avoid such risks and increase system trustworthiness? He gives several examples of good design rules that help increase security and trustworthiness of systems.

I hope that you find these chapters to be of value to you. I am always looking for new material to present. If we have not covered an important area for several years, or if you wish to contribute a chapter, please let me know. I can be reached at mvz@cs.umd.edu. Draft chapters are due June 30 of each year, with final versions submitted during October, for publication generally the following spring.

Marvin Zelkowitz
University of Maryland and Fraunhofer Center, Maryland
College Park, Maryland

Designing Networked Handheld Devices to Enhance School Learning

JEREMY ROSCHELLE AND CHARLES PATTON

SRI International
USA

DEBORAH TATAR

Virginia Tech
USA

Abstract

Handheld devices, especially networked handheld devices, are growing in importance in education, largely because their affordability and accessibility create an opportunity for educators to transition from occasional, supplemental use of computers, to frequent and integral use of portable computational technology. Why and how might these new devices enhance school learning? We begin by discussing a simple but important factor: networked handhelds can allow a 1:1 student:device ratio for the first time, enabling ready-at-hand access to technology throughout the school day and throughout the learner's personal life. We argue that designers need to understand the capabilities of the new generation of handheld computers and wireless networks that are most relevant for learning. We follow this with a discussion of Learning Science theories that connect those capabilities to enhanced learning. The capabilities and features feed into design practices. We describe a set of example applications that are arising from the capabilities, theories and design practices previously described. Finally, we close with a discussion of the challenge of scale.

ADVANCES IN COMPUTERS, VOL. 70
ISSN: 0065-2458/DOI: 10.1016/S0065-2458(06)70001-8

1. Introduction

Handheld devices, especially networked handheld devices, are growing in importance in education, largely because their affordability and accessibility create an opportunity for educators to transition from occasional, supplemental use of computers, to frequent and integral use of portable computational technology [110,118]. Yet educators have been excited about many waves of technology, from film projectors to audio tapes to personal computers and most waves of technology have failed to make a substantial impact in school learning [24]. Given the disappointing history of technology in education, why should we expect networked, handheld devices to be different?

We begin by discussing a simple but important factor: networked handhelds can allow a 1:1 student:device ratio for the first time, enabling ready-at-hand access to technology throughout the school day and throughout the learner's personal life [17]. However, we will argue that merely increasing access to technology in schools and in students' lives is not enough. Time and time again, educational studies have shown

that those technologies that make an impact in learning do so by changing how and what students learn [101]. Further, successful technologies must be integrated into the social practices of schools, which requires integration with teaching practices, curricula, assessments and school leadership. This is a difficult but very important challenge. It is difficult because schools are complex institutions with a dynamic of technology adoption that is quite different from enterprise or consumer markets. It is very important because 21st century societies are increasingly organized around knowledge work and innovation, both of which depend mightily on the high quality of school learning. Without utilizing technology in learning, it is hard to imagine how societies might produce sufficient gains in student learning to continue on successful paths of innovation and improvement in quality of life.

A rather large community of research has grown around this challenge, most recently calling itself the "Learning Sciences" [103]. While networked handhelds present opportunities for learners of all ages, we focus here on the experiences, opportunities, and challenges of using handheld technology in K-12 education. To introduce readers to the broad scope of research relating to networked, handheld computers in K-12 education and falling under the rubric of the Learning Sciences, our article takes the following approach. First, we describe how a new generation of networked handheld technology is enabling students to have greater access to technology in their everyday lives, including school learning. We next review three historical examples of learning success utilizing high levels of access to handheld and/or networked technology. From this review, we draw the conclusion that handhelds can (and actually are already) making a huge difference in student learning. In addition, we observe that handhelds are not simply smaller personal computers. Indeed, these successful examples of technology-enhanced learning drew upon properties of networked handhelds that do not particularly characterize personal computers. Further, the historical success stories drew upon rich integration with social practices, suggesting that successful designers must think about more than the technology—they must understand how people learn and how schools work.

Having set this stage, our logic flows as follows. We argue that designers need to understand the capabilities of the new generation of handheld computers and wireless networks that are most relevant for learning. We follow this with a discussion of Learning Science theories that connect those capabilities to enhanced learning. The capabilities and features feed into design practices. We describe a set of example applications that are arising from the capabilities, theories and design practices previously described. Finally, we close with a discussion of the challenge of scale. Can new designs for technology-enhanced learning surpass the level of success already experienced with the three historical cases?

1.1 Increased Access Enables Frequent, Integral Use

Traditional desktop technology is expensive, and as a result, limited computer resources must be shared amongst many teachers and students. Today the typical student–computer ratio is 5:1, and computers are most often located in special computer labs rather than in ordinary classrooms [16]. The logistics of scheduling class time at the lab—and the time required to move students between rooms—greatly interferes with teachers' abilities to integrate computers into regular learning practices [5]. Thus, despite school's enormous effort to acquire computer resources, there is often a gap between a school's advertised computational facilities and those that a teacher can realistically access [5]. This situation supports occasional, supplemental computer use at best and presents a challenge to integrating technology with other learning materials and activities in the classroom. Further, perfunctory use of technology limits the overall possible impact of computing in education: if an instructional resource is used infrequently, it is unlikely to have a large effect.

In contrast to traditional desktop computers, handheld devices are relatively inexpensive, allowing for each student to own a device or for teachers to have a classroom set with enough for every child. In addition, handhelds are mobile and flexible, allowing for easy use in and across classrooms, field sites, and home environments. Because of these unique characteristics, handhelds hold the promise of enabling many more students to experience integral uses of learning technologies. Indeed, graphing calculators—which are a well-established and effective handheld device—have reached far more K-12 learners than computers. Approximately 40% of high school mathematics classrooms use graphing calculators, whereas only 11% of mathematics classrooms use computers [79]. Finally, because handhelds can be used much more frequently than traditional computer labs, they drastically increase the potential of computational technologies to positively impact the learning process [19].

Importantly, two qualities that have been most associated with successful learning through technology are frequency of use and integration of the technology into the classroom teaching experience. Wirelessly interconnected handhelds provide a unique opportunity to create a learning environment where technology is a transparent, non-invasive support to group learning [21]. Use of technology in the classroom should ideally extend beyond productivity tools and web browsing, to tools that allow more learners to master difficult concepts as they explore and interact with data and ideas. For example, computer simulations can enable 6th grade students to master Newtonian physics concepts at level that surpasses ordinary 12th graders [125]. Early evaluations suggest teachers and students respond to handhelds favourably. In a study of 100 Palm-equipped classrooms, 90% of teachers reported that handhelds

were effective instructional tools with the potential to impact student learning positively across curricular topics and instructional activities [22,120].

Researchers have used a variety of synonymous terms in referring to the use of digital technologies to support human learning. Terms used in the literature include: computer-assisted instruction, educational technology, educational computing, information and communication technology in education, and more recently, e-learning, distributed learning, asynchronous learning, and networked learning. In this chapter, we use the term technology-enhanced learning (TEL), where technology refers specifically to digital technology with graphic displays and keyboards, styluses, buttons or other affordances for hands-on input.

The notion of one-to-one computing (a ratio of at least one computing device for each student) was coined by Elliot Soloway and Cathie Norris. In their keynotes addressed in the IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE2002) and International Conference on Intelligent Tutoring Systems (ITS2004), they argued that today's "personal computer" is not personal to students at school, since students most often have to share with others at a computer lab. The researchers also pointed out that the process of learning changed when all students were able to afford a pencil and again when all students obtained their own books [85]. A similar change can happen if everyone owns and regularly uses a personal computing device as an integral aspect of their learning experience.

Over the next 10 years, we anticipate that personal, portable, wirelessly-networked technologies will become ubiquitous and pervasive in the lives of learners, both in and out of school. The rapid advancement of these technologies is already changing the lives of students outside of school [30,115,54,60,83,96]. Indeed, in many countries, devices like mobile phones or graphing calculators already have a high adoption rate among school-aged children.

As these devices become affordable for the majority of parents and college students, mobile, connected, and personal devices will increasingly come to the attention of educational institutions. For example, the Massachusetts Institute of Technology (MIT) has proposed that $100 laptop computers be purchased for school-aged children by states [12]. At the same time, wireless services and Internet access in many countries will become available in most schools and universities and in public areas, from coffee shops to libraries. For example, Google has offered to bring free wireless access to the entire city of San Francisco.

While the expanded presence of mobile technologies is widely accepted, the specific form of personal, educational computing that will become most available to students is still controversial. Educators today talk about everything from mobile phones and notebook computers to Tablet PCs and personal digital assistants (PDAs). In addition to these general-purpose computing devices, many researchers advocate specialized designed-for-learning devices. For example, graphing calculators

are commonly used in high schools in North America and many European countries. Electronic English dictionaries are commonly used throughout Asia (upgraded with wireless communication capability). With growing interest in the relationships between gaming and learning, students will also be able to use portable gaming devices for learning [42,111]. In the near future, we can expect to see new types of devices emerging as well. The prices of these computing devices and network access will drop, according to Moore's Law and its corollaries [75].

As students increasingly use personal devices for learning outside of school, a new pressure in the adoption of learning devices in schools will emerge. Will students who come to expect mobile, connected, personal devices outside of school demand to use them within school? How will learning in classrooms and everyday life be transformed?

2. Historical Large Scale Successes

It is always difficult to predict how an emerging technology will take shape in a new area of application. Prediction is especially difficult without a sense of history. The biggest misconception about handheld, networked technologies in education is that they are new. To the contrary, at least three handheld and/or networked technologies are already in successful, widespread use. To design for the future, we ought to begin by understanding the past. Hence, we begin with a review of three historical successes: graphing calculators, classroom networks, and probeware.

2.1 Graphing Calculators

Graphing calculators have become one of the most widely adopted handheld technologies in education. In the United States, for example, about 40% of high school students own graphing calculators, and even higher percentages use school-owned devices in the classroom [79]. Graphing products are now integrated with national and state standards (e.g., National Council of Teachers of Mathematics [80]) and they are supported in some curricula. Furthermore, best practices of instruction are well documented [14,107] and teacher professional development offerings are widely available.

2.1.1 Pedagogical Affordances of Graphing Calculators

Like other hand-held instructional technologies, graphing calculators are inexpensive, mobile, and readily adaptable to existing classroom practices. These qualities—combined with the instructional affordances of the technology itself—mean that

graphing calculators have a powerful potential to help students master important concepts in mathematics. Employed as an instructional technology, graphing calculators can enable teachers to foster a problem solving approach to mathematics and help students to reason mathematically. The unique contributions of graphing calculators to problem solving and reasoning include:

- increasing attention to conceptual understanding and problem solving strategies by offloading laborious computations;
- examining the related meanings of a concept through the display of multiple representations, such as exploring rate of change (i.e. slope) in a graph and table;
- engaging students with interactive explorations, real world data collection, and more authentic data sets;
- giving students more responsibility for checking their work and justifying their solutions;
- introducing topics that were previously too difficult for many students (e.g. modeling); and
- providing a supportive context for productive mathematical thinking.

Students with calculators can take on traditional tasks in new ways and also tackle new topics that would otherwise be inaccessible. Rather than laboring over tedious calculations, classes that use calculators can devote more time to developing students' mathematical understanding, their number sense, and their ability to evaluate the reasonableness of proposed solutions. Students can also use calculators to explore concepts and data sets that would otherwise be too complex or cumbersome. For example, students can easily investigate the effects of changing a, b, and c on the graph of $ax^2 + bx + c$, which can be quite tedious using paper and pencil graphing techniques.

Research has also shown that students can often reason best when they experience mathematics through related representations, such as equations, tables, and graphs [35,61]. Graphing calculators can make constructing and using multiple representations easier, allowing students to spend more of their time and intellectual energy exploring the underlying concepts. In addition, technology can link the representations, enabling students to make conceptual connections, such as understanding how a change in an equation links to a change in a graph. Standard mathematical representations can also be linked to other visualization aids, fostering further conceptual understanding.

Research has also shown that students using graphing calculators change their approaches to problem solving: they explore more and their attempted solution

strategies are more flexible [35,61]. In general, students who use calculators better understand variables and functions and are better able to solve algebra problems in applied contexts than students who do not use calculators. Similarly, students who use calculators use graphs more often and interpret graphs better than students who do not regularly use the technology. Finally, students who use calculators are better able to move among varied representations—that is from graphs to table to equations—than students who do not have access to the technology. Clearly, students who regularly use calculators have an advantage over those who do not.

2.1.2 Research on Graphing Calculators

When it comes to instructional technologies, educators and policymakers want to do more than merely identify potentially beneficial tools. In addition, they want concrete guidance on how to achieve an effective implementation and confidence that large-scale implementations will also be successful. Fortunately, strong graphing calculator research is available to address these concerns.

In the United States, the National Assessment of Education Progress (NAEP) samples both 4th and 8th graders throughout the country and measures how many students perform at proficient and advanced levels in mathematics. This research has consistently shown that frequent use of calculators at the eighth grade level (but not at the fourth grade level) is associated with greater mathematics achievement, stating:

> Eighth-graders whose teachers reported that calculators were used almost every day scored highest. Weekly use was also associated with higher average scores than less frequent use. In addition, teachers who permitted unrestricted use of calculators and those who permitted calculator use on tests had eighth-graders with higher average scores than did teachers who did not indicate such use of calculators in their classrooms. The association between frequent graphing calculator use and high achievement holds for both richer and poorer students, for both girls and boys, for varied students with varied race and ethnicity, and across states with varied policies and curricula [79, p. 144].

A study by Heller et al. [50] corroborates the NAEP findings. Heller examined a model implementation, which included a new textbook, teacher professional development, and assessment tools—all aligned with the graphing technology by the theme of Dynamic Algebra. This study shows that daily use of graphing calculators is generally more effective than infrequent use, and establishes that the teachers and students who used graphing calculators most frequently learned the most.

2.1.3 An Example from New Zealand

Researchers in different settings have investigated the effectiveness of graphing calculators in relation to students, teachers, and schools with diverse characteristics. Alan Graham and Michael Thomas, for example, examined the effectiveness of graphing calculators in algebra classrooms in New Zealand [44]. The study compared pretest and posttest scores for students in treatment and control group classrooms in two schools. In all of the classrooms, the regular classroom teacher taught the "Tapping into Algebra" curriculum module. In treatment group classrooms, each of the students received a graphing calculator to use throughout the module; in control group classrooms, students did not use graphing calculators. Students in all classrooms had similar background characteristics and math abilities. Graham and Thomas found that students in the treatment groups performed significantly better than students in the control groups on the post-test examination.

2.1.4 Meta-Analyses Show the Effectiveness of Graphing Calculators

Meta-analysis is a technique that enables researchers to statistically summarize the results of a large body of experimental studies, yielding a robust estimate of true effectiveness. A meta-analysis by Ellington [35] summarized 54 classroom experiments, of which 80% employed some form of random assignment of students to experimental groups (using calculators) and control groups (not using calculators). Random assignment is a key component of true experimental studies, as it allows social scientists to make strong causal inferences with the fewest threats to experimental validity [20]. Ellington's analysis shows a positive effect of graphing calculator-based interventions on student achievement. The effects are substantial, often increasing an average student's achievement by 10 to 20 percentile points [35]. In addition, the studies suggest that when graphing calculators are allowed on tests, gains extend from calculations and operations to conceptual understanding and problem solving. Ellington's summary includes a wide variety of grade levels, socio-economic backgrounds, geographic locations, and mathematical topics, suggesting that the effectiveness of calculators holds true in a variety of contexts.

A second meta-analysis looked specifically at Algebra. Khoju et al. [61] screened available research using stringent quality-control criteria published by the US Department of Education's What Works Clearinghouse. They found four suitable studies that examined the impact of graphing calculators on Algebra learning. Across a wide variety of student populations and teaching conditions, use of graphing calculators with aligned instructional materials was shown to have a strong, positive effect on Algebra achievement.

2.1.5 Why have calculators been so successful?

A number of key features contribute to the success of graphing calculators in bolstering math learning. Graphing calculators are relatively simple, robust and cheap; they are also remarkably free of much of the complexity that accompanies full-featured computers. More importantly, there is a deep scientific linkage between the capabilities of the technology and how people learn. Students learn best with increased learning time, scaffolding, formative assessment, and opportunities for reflection and revision—qualities that can be achieved with graphing technology.

Two less readily obvious factors also contribute to the success of graphing calculators. First, the adoption of the technology has been led by practicing teachers, who function as the key champions and influencers in a professional community [38]. Second, efforts to integrate graphing calculators into classrooms did not begin with the expectation of a rapidly transformed classroom, but rather provided a context to support a long, steady trajectory of continuous improvement [31]. In this way, teachers can begin with one or two relatively simple applications of the technology, and gradually increase the depth and breadth of their calculator integration as they grow more comfortable with the technology. At each stage, graphing calculators can provide concrete enhancements for teaching and learning math.

In summary, the evidence for the impact of calculator use on student achievement is robust and consistent. Graphing calculators are inexpensive and can align with curricula, instructional practices, and assessments. In addition, teacher professional development for integrating graphing calculators into classroom practices is widely available. Collectively, the combination of curricula, pedagogy, assessment, and technology—aligned through professional development—creates the circumstances for sustained improvement in deep conceptual learning.

2.2 Classroom Response Systems (Feedback)

A second effective handheld learning technology is the networked response system. The first notably successful classroom response system, Classtalk, was patented in 1989 [2], and similar product concepts have since been re-implemented many times.[1] A major benefit of these networked response systems is that they have enhanced classroom communication between the teacher and the students. Employing a combination of networking hardware and software, networked response systems provide displays that reveal what students are doing, thinking, and understanding.

[1] Eleven current commercial products have been identified. Examples include: eInstruction (http://www.eInstruction.com), TI-Navigator (http://education.ti.com/us/product/tech/navigator/features/features.html), and Discourse (http://www.ets.org/discourse/).

Teachers can use the information provided through classroom networks to augment the natural communication flow of the classroom.

2.2.1 Instructional Processes Using Networked Response Systems

From a technological viewpoint, a classroom network can be thought of as a tool to augment the interaction loop between teacher and students. The concept of interaction loops builds upon Weiner's pioneering work in cybernetics [124]. A traditional loop opens when the teacher assigns an activity to a student and continues when the student turns in the assigned work to the teacher. The loop is completed days later when the teacher returns the graded assignment to the student. In this model, only a few students are involved in the process of sharing their work, there is very little discussion after a question has been answered, and for the majority of students there is a long delay before they receive any response from the teacher.

In contrast, the networked loop demonstrates classroom interaction occurring much more rapidly and with a smaller-sized task. In this context, an activity might be a request to answer a question, solve a problem, state a position, write an equation, or give a reason. Students provide their responses as input into a personal computing device, such as a graphing calculator, palm-sized computer, laptop, or even a special-purpose device similar to a TV remote. Then, the teacher's desktop machine collects and aggregates the student work, and presents it in a meaningful graphic that teachers and students can interpret quickly.

2.2.2 Pedagogical Affordances of Networked Response Systems

Teachers and researchers have found that the ability to harvest students' work immediately has a range of applications. In the simplest case, a teacher poses a multiple-choice question, and the classroom network rapidly produces a histogram showing the distribution of responses in the classroom. Seeing the histogram makes it easier for both teachers and students to focus on what needs to be learned and to engage in discussion around those topics. In slightly more sophisticated cases, students mark a point on an image or show the line they graphed. These points, lines, or even motions can be aggregated instantly to reveal higher-order patterns [48]. In some of the most advanced uses of networked response systems to date, students engage in a participatory simulation ("part-sim"). For example, each student controls a traffic light in a classroom simulation of traffic patterns shown on the public display. Then the class collaborates to identify some of the principles of operation that would allow traffic to flow smoothly [127].

Even the most basic usages of networked response systems can profoundly impact teaching and learning in the classroom setting. Teachers and students can use the readily interpretable data generated by the network to observe patterns and differences among student responses. By revealing how students are thinking, the response comparisons also enable teachers to drive all students to explain their thought processes more thoroughly. The shared points of reference provided by the system, in conjunction with inquiries from the teacher, can in turn catalyze class discussions of complex concepts. Harvard's Eric Mazur, an early leader in developing the pedagogical use of networked response systems, calls this approach "Peer Instruction" [70], suggesting that the real heart of the learning occurs when students engage with each other conversationally on the basis of the dissonances revealed by the shared display.

In spite of the fact that networked response systems execute a fairly simple function, early adopters have consistently described the technology as a catalyst for a significant, powerful shift in the classroom climate, pedagogy, and resulting learning [3, 28,34,84]. Formative assessment is known to be a very powerful intervention [7] and these systems enable students to receive much more feedback than normal. In addition, students can see where classmates share their misunderstandings and recognize that they are not alone. It is important to note that the overall impact of the lesson need not be at all test-like. Student work can be displayed anonymously, so that embarrassment is essentially eliminated [84]. Finally, real-time information about students' comprehension enables teachers to modify instruction to meet the needs of learners.

The above discussion suggests that effective implementation of networked response systems requires integrated roles for the teacher and the technology, as well as a combination of pedagogical technique and computational capability. The role of the technology in transforming classroom learning is small but extremely valuable. In particular, the technology provides anonymity, speed of response collection, and the ability to produce a shared visualization that enhances mutual pattern recognition. But non-technological social processes still carry much of the burden of teaching and learning: asking questions, explaining, clarifying, summarizing, etc.

The most recent and thorough examinations of this technology (using Texas Instruments graphing calculators and a supplementary networking product) emphasize a virtuous cycle of changes that result in a classroom that uses the system [28, 84]. The cycle maps onto the four factors of successful classrooms identified in a groundbreaking summary of learning science research [33]. The classroom becomes more learner-centered, assessment-centered, knowledge-centered, and community-centered. These are powerful and apparently robust effects from a fairly simple use of networked response technology. Further they do not appear to be limited by subject matter, and can be significantly extended beyond the range of multiple choice

and short answer questioning. For example, 'image map assessments' have been proposed in which students' marks on images are aggregated [100]. Others are working at Cartesian aggregation spaces of contributed mathematical functions [58]. Many more kinds of classroom response aggregation are possible. Clearly, pedagogical application underlying networked response systems deserves much more research attention in the coming years.

2.2.3 Mazur's Peer Instruction

In an effort to improve his students' gains-scores on an introductory physics assessment, Mazur pioneered a new style of classroom practice that relied on augmented teacher-student communication and increased classroom discussion around important concepts. Mazur's new practice began with what he termed a "ConcepTest"—a challenging question designed to foster thinking and discussion that gets at the heart of the target concept. Mazur would pose a ConcepTest to his students, allow them to ponder the answer, and have them submit a response via the classroom network. Based on the percentage of students who answered correctly, Mazur adapted his subsequent instruction by moving on to the next topic, or by spending more time on the subject until mastery was achieved. If a question uncovered lots of misconceptions, Mazur would facilitate discussion, encouraging students to explain and debate their understandings. These kinds of discussions focus on understanding the correct conceptual structure because, while the questions are superficially simple, they are laser-like in their ability to demarcate misconceptions and stimulate valid reasoning.

The process of discussing possible responses, misconceptions, and conceptual understandings amongst peers, through ConcepTests and networked response technology, lies at the crux of Mazur's pedagogical approach. He calls this process Peer Instruction. In his written work on this topic, Mazur emphasizes the strategic planning and classroom practice that are required for a teacher to implement Peer Instruction. As with other instructional technologies, classroom networks can provide critical enablers for enhanced teaching and learning, but it is pedagogical talents that make the innovation successful.

Mazur shows that in the year he first implemented his Peer Instruction methods, the distribution of scores on the Force Concept Inventory (a standard assessment of student understanding in Physics) shifted markedly from pretest to posttest, suggesting that his new approach was effective [70]. Impressively, after the posttest only 4% of students were below the threshold of mastery as defined by the Force Concept Inventory (FCI). There was a steady improvement in scores in each subsequent year over a decade, even though students' incoming scores did not change [23]. Mazur also compared the scores of his 1985 class with those of the 1991 class by giving

them the same final exam. He found that the scores shifted upwards about 10% and manifested a much narrower distribution, showing that he had achieved an important impact on his students' conceptual understanding [70]. He also examined students' responses to more conventional physics problems (stressing mathematical manipulations) versus conceptual problems and argued that while conceptual understanding improved, students' ability to solve the more conventional problems was not compromised [70].

TABLE I
REPORTED BENEFITS OF NETWORKED RESPONSE SYSTEMS

Claimed benefit	Research studies citing benefit
Promotes greater student engagement (16)	Boyle and Nicol [10], Burnstein and Lederman [13], Crouch and Mazur [23], Cue [25], Dufresne et al. [34], Fagen et al. [37], Kaput and Hegedus [59], Horowitz [53], MacDonald [69], Poulis et al. [90], Ratto et al. [93], Robinson [98], Scheele et al. [104], Webking [123], Wilder Foundation [126], Woods and Chiu [128]
Increases understanding of complex subject matter (11)	Abrahamson [1], Boyle and Nicol [10], Crouch and Mazur [23], Fagen et al. [37], Hake [45], Hartline [46], Kaput and Hegedus [59], Poulis et al. [90], Sokoloff and Thornton [108], Wilder Foundation [126], Woods and Chiu [128]
Increases interest and enjoyment of class (7)	Abrahamson [1], Boyle [9]; Boyle and Nicol [10], Burnstein and Lederman [13], Cue [25], Dufresne et al. [34], Woods and Chiu [128]
Promotes discussion and interactivity (6)	Boyle and Nicol [10], MacDonald [69], Robinson [98], Scheele et al. [104], VanDeGrift et al. [121], Woods and Chiu [128]
Helps students gauge their own level of understanding (5)	Dufresne et al. [34], Ganger and Jackson [41], Piazza [89], Robinson [98], Webking [123]
Teachers have better awareness of student difficulties (4)	Abrahamson [1], MacDonald [69], McNairy [71], Robinson [98]
Extending material can be covered beyond class time (2)	Ratto et al. [93], Truong et al. [119]
Students do more thinking in classrooms (2)	Boyle and Nicol [10], Cue [25]
Improves quality of questions asked (1)	Ratto et al. [93]
Overcomes shyness (1)	Truong et al. [119]
Saves time (1)	Boyle [9]
Simplifies record keeping (1)	Webking [123]

2.2.4 A Review of the Effects of Networked Response Systems

Table I illustrates the primary impacts of networked response systems, with the most frequent findings listed first. As the table indicates, the two most commonly reported effects are increased student engagement and improved conceptual understanding. More than half of the studies reported that students were more engaged—as evidenced by their attention and attendance in class—in classes where networked response systems were used. Another eight studies reported increased student interest and enjoyment in these classes, which in turn probably contributed to higher levels of engagement. Just under half of the studies also reported gains on learning outcomes, as measured by end-of-course objective tests and standardized achievement tests. Although few of these studies used any comparison group and none placed statistical controls on comparison groups, increased student understanding of complex subject matter was widely reported. Other commonly reported outcomes included increased "interactivity" and group discussion, as well as increased awareness among teachers and students of the difficulties students faced in mastering subject matter. In addition to the impacts highlighted in this chart, a handful of studies also reported "more pressure on students to think" [3], students coming to class more often [70,27], and reduction in student anxiety [27].

2.2.5 Why Have Networked Response Systems Been so Successful?

A number of key features contribute to the success of networked response systems in bolstering learning. Importantly, classroom response technology accommodates common teaching practices while also offering new enhancements for classroom teaching. Like graphing calculators, response systems are also relatively simple, robust, and cheap, and there are purposeful linkages between the technology and insights from the learning sciences. In addition, response systems address a specific classroom need—namely, enhanced communication and feedback among teacher and students. Of course, effective teacher implementation of classroom communication practices is a critical component of improved student performance. Yet network technology can be a key enabler of this improved communication and feedback by facilitating rapid cycles of assigning, collecting, interpreting, and discussing student work.

2.3 Probeware

Of all the instructional technologies for science classes, the technology with the longest track record is that of electronic probes and sensors and associated software (hereafter, "probeware"). Used in classrooms, probes instantaneously gather and graph data from live experiments [74,78]. It is hard to overstate how important this is in classroom learning. Before probes were readily available, students typically gathered data in class, graphed it at night, and analyzed it the next day. An enormous "gulf of evaluation" [82] thus lay between collecting the data and making sense of it. By the time students were interpreting data, they often could barely remember what it was about. Further, students are often careless in collecting data, leading to a 'garbage in, garbage out' problem. Because of the pace of school, teachers must often ignore these bad data sets and simply move on to the next topic. This sends students a very poor message about the nature of scientific inquiry. Consequently, the ability to collect, graph, and analyze a whole series of experiments in one class period is a radical innovation. By closing the gulf of evaluation, probes support the long-term pedagogical drive towards 'inquiry-centred' science classrooms [118].

Teachers can now purchase a variety of kits that enable students to collect data from experiments using a computer or graphing calculator with attached probes or sensors. Probeware can be used in all areas of science: pH sensors in Biology, pressure probes in Chemistry, and motion recorders in Physics. They can be used in the classroom, but are also commonly used in the field. A very popular scenario for using handheld probes is water quality evaluation [120]. Students take their handhelds and probes to a nearby stream and each student takes measurements at different points along the streambed. The students combine their data by beaming or aggregating onto a common teacher machine. Back in the classroom, students use their handhelds to graph and analyze the combined data set.

Thus, the major benefits of probeware include:

- ease of collecting and recording accurate data;
- ability to collect time series data;
- use of the computer or calculator for instant graphing and analysis of data;
- possibility of exchanging or pooling data sets among students.

Probeware, used well, can open avenues of exploration for students as they use it to experiment with phenomena of their choosing. Students using probeware experience less tedium in setting up their experiments and focus more on the research questions and the data. This can focus students on the science meaning behind the data instead of the rote procedures for collecting data in a valid way. Students can also

collect more observations than manual methods allow, permitting more comparisons and hence, better generalizations. Through the immediacy of data collection and data display, probeware directly connects observation in the real world to abstract representations and allows for the investigation, variation, and play that provide better understanding of important scientific concepts.

Probeware has also been shown to support data collection by students for scientific research at a world-wide scale. The GLOBE environmental monitoring program teams scientific researchers studying local, regional, or global questions with students and community groups who provide on-the-ground data collection in a scientifically valid way. Data is reported to a central database and can be analyzed by both students and scientists for their own investigations.

Studies [118] indicate that students who learn with curricula that include probeware take a more active role in their learning and are taught to use skills of observation. Collaborative learning is naturally supported by the ability to share displays of data and integrate others' findings. Finally, students who learn with these manipulatives show better retention of concepts than those taught in a lecture format.

2.3.1 Research

While there has been a significant volume of research showing that probeware can help students learn to do inquiry, there has been much less research documenting the impact of probeware on science content learning. In their study of a probeware-supported inquiry unit on water-quality, Krajcik and Starr (in [118]) aimed to examine the impact of project-based learning and probeware on science content learning. The water-quality unit was taught by two science teachers, to about seventy 7th grade students in four classes at Greenhills Middle School. The Greenhills School devotes ample attention to integrating innovative teaching methods, through professional development, technology and curricula—thus making them distinct from other schools and teachers. As a result, the researchers decided that control classrooms, either outside the school or with different teachers at the same school, would not provide a valid comparison group for the study. Instead, the researchers used assessments aligned with national standards in order to show that an innovative learning environment employing project-based learning and probeware technology can result in important learning outcomes for students [118].

In order to get an understanding of student learning, the researchers used three tools to assess students' content knowledge and their ability to apply target concepts to new and unique situations. The first tool was a test consisting of multiple-choice, short-answer, and extended-answer items which students took before and after the unit. Students were also required to draw pre- and post-unit 'concept maps' illustrating their understanding of the factors and linkages related to water-quality. Finally,

students created booklets, which they added to at each of three visits to the field. The extended-answer questions, as well as the concept maps and science booklets, were evaluated using precisely defined assessment rubrics.

Statistical analyses comparing the pre- and post-test results for each of the various types of assessments revealed that students made significant gains in both basic content knowledge and deeper conceptual and analytic understandings of water quality [118]. Furthermore, large effect sizes associated with the mean-score increase from pre- to post- measure indicate that the scores were not only statistically different from one another, but that the gains themselves were substantial. Because students were not concurrently participating in other related science interventions, it can be surmised that the project-based, probe-ware program was responsible for the student gains.

The two teachers who implemented this science curriculum wrote about their perspectives on the benefits to students of using probe-ware in a project-based learning environment. First of all, teachers felt that students' relationship with the technology improved over time: students learned to trouble-shoot problems on their own; they began to understand instrument calibration; and they learned to properly care for and maintain their tools. In addition, use of the technology benefited students' attitude toward science learning. Because the project was rooted in students' real-world environment, they viewed the process and the data as meaningful. In turn, students felt ownership in their own work, were excited to interpret their data, and were intent on finding ways to support their research and findings. Finally, teachers felt that the experience also had a positive impact on students' in-depth understanding of water quality, and on their thinking and learning skills in general. Teachers found that students improved in their ability to analyze and synthesize data, to understand the implications of data, to create more comprehensive pictures of the situation under investigation, to make connections between data and key environmental factors, to identify patterns and trends in data, and to understand relationships in the field (in [118]).

2.4 Discussion

The three historical examples presented above are important, in part, because they demonstrate that networked handhelds have produced worthwhile improvements in school learning. In the case of graphing calculators, the improvements have occurred at a scale seldom seen in educational technology: around 40% of high school students have a graphing calculator and the data from numerous studies is consistent and positive. We see graphing calculators as representational tools; they allow students to engage with mathematics in both linguistic (e.g. algebraic symbols) and graphic ways. Research shows that providing students with multi-

ple, linked representations—and especially a combination of linguistic and graphical representations—can produce powerful learning gains. Networked response systems, on the other hand, work through a different mechanism. They are participatory and feedback tools, which in the presence of a capable teacher can transform classroom dynamics to increase student engagement in learning. Probeware incorporates elements of representation (instantly graphing data) and feedback (students can quickly see if they are getting bad data) but also change how students experience physical place—students can more easily measure and quantify their world and can thus more readily engage in scientific inquiry. In the sections that follow, we will continue to build upon these aspects of historical success.

We draw a second important set of conclusions from these examples as well—namely, that our imagination of the meaning of networked handheld devices should not be overly constrained by a vision of anytime/anywhere access to school information or a vision of handhelds as small computers. Some researchers are drawn to thinking about how school information could be made available anywhere. They imagine how learning might be improved if students could see their class schedules online, review their grades, retrieve homework, submit assignments, ask questions of their instructor, and so on . . . all with a personal, low-cost, mobile, wireless handheld device. There are two problems with this vision. First, it relies on placing an interface to instructional management and learning tools on a small screen with limited input possibilities. Second, access to administrative information is unlikely to have an impact on how students learn.

Other researchers imagine handhelds as small, inexpensive computers. They seek to package all of the complexity found in computer applications into a handheld format. It is important to note, however, that the most successful historical examples involve rich social practices built around rather simple (but uniquely functional and reliable) technology. A number of important lessons emerge from the realization that instructional technologies tend to be most effective when simple and efficient technologies are combined with sound pedagogical techniques [15]. This finding indicates, for instance, that restraint is a valuable quality when designing instructional technologies. Rather than trying to create an all-encompassing system, we utilize technology for specific, targeted affordances for tasks that are not well served with the process and pattern already in place in the classroom. Talking and passing out papers are integral parts of handheld-based activity. Students should not struggle to read instructions about an activity on the same tiny screen on which they are conducting the activity when paper-based or white-board based instructions work very well. Furthermore, teachers we have studied prefer students handing in work on paper, for later grading in their easy chair.

The examples of graphing calculators, networked response systems, and probeware emphasize this direction towards simple, well-honed technology and rich, ped-

agogically developed social practices. In each case, the technology performs a small, precisely-defined function uniquely well, but much of the rest of teaching and learning is left to social practice. Probeware excels at collecting and sharing data, and little else. Yet it supports a transition from routine, unexamined scientific practice to inquiry-based practice in the classroom. In networked response systems, the questions are often authored and posed to the class offline, and the technology performs only the essential functions of gathering responses anonymously and instantly summarizing them in a publicly displayed histogram. Yet the system promotes critical thinking as students and teachers compare, elaborate, explain, critique, and debate about the patterns of response. The participatory simulation variants of networked response systems excel at exchanging small, extremely simple data messages among spatial neighbors. Yet students can readily become socially involved in designing experiments. For example, they can try to slow the spread of disease by quickly assigning those infected to quarantine, or by having fewer social partners. As evidenced in all of these examples, the technology serves to enable new types of instruction, yet much of the actual learning occurs in lesson-design and debriefing phases that are not mediated directly by the technology.

This fairly weak coupling of informatics and social practice results in many seemingly ironic outcomes. For example, the essential pattern of the classroom can become one of 'peer learning' despite a technology that has no peer-to-peer communication capabilities (such as Classtalk). The sense of community in the classroom can evolve rapidly, despite the lack of any 'online community' tools. Students become more involved in designing and interpreting controlled experiments, despite software that has no sense of variation in parameters. (To perform an experiment with participatory simulations, students only change their spatial movement; the software has no knowledge that it is involved in a 'different' experiment. The parameters are embodied by the students and are not explicit constructs in the software.) Students can perceive receiving much more individualized assessment feedback, despite the fact that all they ever see is a shared, anonymous public representation of the group's thinking. As these apparent ironies indicate, the causal arrow from technology affordances to social practice is often quite crooked. Consequently, *research attention should be directed at identifying those learning niches in which simple technology could fit in extremely and uniquely well and to understanding the social practices by which those new affordances become powerful.*

3. Technology Context for Learning Applications

The world of new hardware and networking capabilities deployed beyond the classroom is constantly expanding. With each new capability the realm of potential

learning opportunities also widens. New computational functions, graphs, animations, and other representational capabilities can potentially augment the ways in which information can be presented to students, thereby creating opportunities for more students to grasp complex concepts in math and science. Similarly, these new representational and networking functions can potentially expand the variety of ways in which students can express, share, and demonstrate their knowledge. Together, these capabilities could potentially increase opportunity for peer-to-peer learning, and improve teachers' capacity to assess their students' understanding and tailor instruction accordingly. To better understand these prospects we examine some important technology developments from the perspective of their potential fit with classroom practice and their potential uses to improve learning.

As we move into this future of new capabilities and new opportunities, a number of important realities and choices will shape the potential for learning devices to make a difference in education. First, the success or failure of particular classes of devices to attain market dominance may affect viable choices for the classroom. Second, details of the physical network infrastructure and of the communication protocols enforced between machines have huge implications for classroom potential. Third, the stance that designers take about the relationship between systems affordances and the Internet has implications for the complexity, or "weight," of the system, and therefore its flexibility and how often it gets used. Finally, in spite of efforts to use inexpensive devices such as cell phones, or to create inexpensive versions of more pricey devices (as with the $100 laptop), the cost of technology is still a significant hurdle for many schools. As technological innovation and learning-science insights create momentum and a strong rational for the use of educational technology in schools, cost remains as a social brake on technology adoption in education.

3.1 Market Dominance

The question of general versus specific functionality will be paramount in the evolution of educational technologies. Until recently, it has been taken for granted that handheld devices and computers are the same, learning-wise, and that therefore general-purpose functionality would be most useful. Yet the body of research on specific-use devices (like graphing calculators) paints a drastically different picture. Experience has shown that devices with specific, targeted applications can integrate smoothly with existing classroom practices and effectively enhance classroom learning. We have argued that this capacity is crucial for use and success in the classroom. The question remains, though, of whether design-intensive point products can prosper or even survive in the new economic environment. In the larger market for small devices, for example, there is a movement towards internetworking. *Internetworking* is when the functional usages and adaptations of devices are inter-related and

inter-dependent, for example, when a PDA is also a cell phone. The fit of seamlessly inter-networked devices into classroom practice is, at best, unclear at this time. Thus, on one hand, we have products that have demonstrated educational effectiveness, but may not have broad market viability. On the other, while there is great hope for general multi-purpose devices, the educational effectiveness of such general-purpose devices has yet to be documented.

3.2 Physical Networks

One arena in which these larger economic issues will play out and overlap with technical and design issues is in the infrastructure and topology of the network—characteristics that are fundamental in determining how easily and effectively networked handhelds can be used in classrooms. A major choice is between infrared (IR) based beaming and radio-frequency (RF) connectivity. Our primary work has been with machines that use infrared (IR) beaming, affording spatially-directed, point-to-point communication.

Infrared technology has some strong advantages in classrooms. First, IR requires no fixed infrastructure and no configuration, which allows teachers to adopt the technology without becoming or employing network administrators. It avoids dependencies on the uptime of other network components.

Second, IR simplifies the designation of communication targets: users spatially specify to whom they are beaming, and thus do not need to pick user names from lists. The appropriateness and timing of a particular beam is negotiated in the social realm with little technical overhead.

Indeed, IR fits ad hoc ensembles of students that frequently occur in classrooms: Teachers expect to be able to say "Everyone who is done, come to the front of the class and bring your Handhelds" and then create pairs from the students who are ready for the next task. Teachers may design tasks for pairs, but when there are an uneven number of students in the class, they expect the students to form trios. Finally, students do not lose communication functionality when on a field trip.

Furthermore, because of its punctuated nature, IR does not create as substantial a need for power as does radio-frequency communication. Special purpose devices provide far more opportunities than general purpose ones to optimize power consumption without compromising performance.

These issues that promote infrared connectivity turn out to be very important in practice in the classroom. However, radio-frequency communication is coming to dominate the market and it too has advantages. It can, in theory, create simple mechanisms for a teacher to get everyone's device to a particular state, or to collect work from all students simultaneously. It can support in a straightforward fashion the powerful image map assessment and aggregation activities we listed above. It can support

access to the Internet. Hence, we predict great interest in radio-frequency networking in classrooms in the future.

3.3 Connections to the Internet

However, radio-frequency connectivity is not itself a full classroom solution. In practice, wireless Internet use on small handheld screens has been problematic. Despite intensive design and strong hype, Wireless Application Protocol (WAP) has been a spectacular failure [62]. Equally surprising, Short Messaging System (SMS) has succeeded beyond anyone's expectations [96]. However, even common Internet applications can be quite problematic in classrooms. Schools, for example, have been tempted to ban instant messaging because it enables cheating and disruptive behavior [91]. Further, attention is a teacher's most precious commodity, and no teacher wants her students' attention focused on messaging with friends outside of class [106].

There are other reasons we may not want the most general form of radio-frequency Internet connectivity as currently conceived for classroom-based applications. For moderately loaded shared media networks, contention-based access to the medium by individual nodes is typically most efficient. In systems where collisions can be detected by the sender (such as in wired Ethernet) this efficiency holds up even to highly loaded situations. However, in wireless networks, nodes cannot both send and receive simultaneously. Consequently, collisions between senders cannot be directly detected. Connectivity may degrade disastrously as more users attempt to share the medium. The sudden and apparently unpredictable loss of connectivity due, perhaps, to users in a neighboring classroom beginning to access their own wireless access point, creates a lack of trust in the reliability of wireless and consequently less than full-scale adoption.

In classrooms and other situations where proximal social interaction is a key component of communication, wireless range beyond perhaps a decameter ceases to be a benefit and is, instead, a liability. In addition, physical proximity to other users and to network resources such as access points provides an opportunity to greatly simplify the usually daunting process of configuring. Combining the use of point-to-point communications for selection ("this is the access point I want to connect with") and automated configuration, with the use of short-range, high-bandwidth radio communications would appear to provide much higher reliability as well as greater ease of use in classroom-like situations.

In other words, the notions of RF communication, and "connection to the Internet" turn out to need much more definition and elaboration as a concept in order to be useful in a classroom context. Any design wants the benefits of all choices and the costs of none. What makes this more challenging is that many features that are clear benefits in non-educational situations are not really benefits in the classroom.

TABLE II
A PARTIAL LIST OF HANDHELD PROJECTS CLASSIFIED BY CLASSROOM FUNCTIONALITY AND TECHNOLOGICAL STATUS

	Personal/ background tools	Central representation devices	Controllers of other devices	Communica-tors	Teacher management devices
Clicker systems	Boomerang (Tatar et al. [116])	ClassTalk (Dufresne et al. [34])	PUC, CPoF (Myers et al. [77])	I-Guides (Hsi [55])	EduClick (Liu et al. [67])
Graphing calculators	PIGMI (Hennessy [51])	Gridlock (Wilensky and Stroup [127])	LabWorks (Morgan and Amend [76])	Match-My-Graph, Slot Machine (Tatar et al. [116])	HubNet (Wilensky and Stroup [127])
PartSims	Cooties (Soloway et al. [110])	Chemation (Bobrowsky et al. [8])	Thinking Tags (Colella et al. [18])	HEARTS (Jipping et al. [57]), Geney (Danesh et al. [26])	MathWorlds (Hegedus and Kaput [47])
System sims	CritterVille (Soloway et al. [110])	NetCalc (Tatar et al. [116])	MRSCL (Mitnik et al. [72])	Sketchy (Bobrowsky et al. [8])	Environmental Detectives (Klopfer et al. [64])
Awareness devices	StudySpace (Schnase et al. [105])	Data Doers (Tatar et al. [116])	Symbiotic Environment (Raghunath et al. [92])	Awarenex (Tang et al. [114])	Information Aware System (Wang et al. [122])
Focused practice	VeGame (Belloti et al. [6])	Who's who? (Moher et al. [73])	Probeware (Tinker and Krajcik [118])	Electronic Guidebook (Bannasch [4])	Code It! (Goldman et al. [43])
Active document exchangers	FreeWrite (Bobrowsky et al. [8])	Plantations Pathfinder (Rieger and Gay [97])	Campus Mobile (Demeure et al. [32])	NERTS (Jipping et al. [57])	Quizzler (Penuel and Yarnall [87])
Formative assessment	WHIRL project (Roschelle et al. [102])	Palm sheets (Soloway et al. [110])	SLiC project (Soloway et al. [109])	ImageMap (Roschelle and Pea [100])	Gradebook (Penuel and Yarnall [87])
Information delivery/ storage	Fling-It (Soloway et al. [110])	MCSCL (Zurita and Nuss-baum [129])	Cornucopia (Rieger and Gay [97])	PiCoMap (Luchini et al.[68])	NotePals (Davis et al. [29])

Notes. Citations may be found at http://www.manleyhopkins.cs.vt.edu/handheldlist.pdf (reprinted from Kim and Tatar [63]).

In this section, we have discussed background properties of existing handheld connected technologies that condition the design of learning activities. A great deal of research, as mentioned in Table II, has attempted to utilize different aspects and properties of these systems. In the next few sections, we will present some examples of the many kinds of promising connected classroom programs and projects that are currently under investigation. Although some are currently downloadable, each should be taken as an indicator of what may come in the future rather than a plan, as none of them rise to the level of influence attained by our earlier examples. We will then use these examples to return to the importance of guidance from the learning sciences in creating handheld, wirelessly connected classrooms and discuss some of the implications of this on-going work for designers of the future.

3.4 Emergent Classroom Connectivity

3.4.1 SimCalc: Connectivity and Dynamic Representation

The SimCalc project is investigating new applications for graphing technology that will enable more students to develop a conceptual understanding of key concepts in the mathematics of change and variation. SimCalc builds on the strengths of graphing calculators by using robust and inexpensive handheld devices, enabled with connectivity and animation. However, compared to the normal use of graphing calculators, SimCalc substantially increases the student's interaction with multiple, linked, dynamic representations. That is, the student's focus is brought to bear on the relationship between behavior in the simulated world of cars, elevators, or dots, and of the position or velocity graphs that describe their motion as the different representations are animated. Networking capabilities allow teachers to deepen math content and increase student participation in their classrooms by permitting the focused and mathematically relevant contrast between what individual students see on their own screens and the aggregated behaviours on the large screen at the front of the classroom. Thus, students might "count off" so that each student has a unique ordered pair of his/her table and individual number, then create a "car ride" that reflects those parameters (e.g. the student from Table 1, position 4 would create a car ride in which $y = 1x + 4$; the student from Table 4, position 1, would create $y = 4x + 1$), then aggregate the "rides" at the front of the room. Visible patterns emerge. SimCalc researchers Hegedus and Kaput are excited about more than test scores: "classrooms that integrate dynamic software environments with connectivity can dramatically enhance students' engagement with core mathematics beyond what we thought possible . . ." [48, p. 54].

Under these conditions, technology becomes a pervasive medium in which teaching and learning take place in the social space of the classroom. Thus, as more work

happens through collaborative interaction, learning increasingly occurs in the social space [113]. This collaborative learning drastically augments the learning that occurs through individual interaction with technology devices. With careful pedagogical guidance by teachers, students can progress through a trajectory of understanding in which their focus advances "from static, inert representations, to dynamic personally indexed constructions in the SimCalc environment on their own device, to parametrically defined aggregations of functions, organized and displayed for discussion in the public workspace" [49, p. 135].

3.4.2 *ImageMap: Aggregating and Presenting Student Responses to Enhance Learning*

ImageMap is an assessment feedback system for supporting media-rich learning conversations. In a classroom lesson using ImageMap, an image (e.g. graph, map, photo) is distributed to each student with a handheld, networked device; the teacher poses a question about the representation; and each student annotates the image with a response. Next, a server receives the responses from the pool of students, aggregates the responses by superimposing the student annotations on the image that was distributed, and projects the aggregated responses on a public display, allowing students and teachers to see the distribution pattern of different answers. Thus, students might mark the Confederate States during the Civil War on a map of the United States. Many might choose Alabama or Georgia, but fewer would hazard guesses about Delaware or Maryland. In this way, the ImageMap assessment represents degrees of student understanding through a direct spatial mapping of individual contributions to an aggregate representation.

In planned extensions to the ImageMap, developers are extending the strategy of aggregating individual responses so that an exploration can occur simultaneously with all students participating. The idea is that an unknown shape (perhaps a phase plot of a chaotic motion) can be generated by having many students each exploring different portions of the parameter space. As the plot fills in with different contributions, students can start to see regions that have not been explored and ones where something interesting might be happening. This intermediate representation can then direct students' continued exploration, as they see what they are building together [86].

As is the case with a number of networked handheld applications, the teacher role during ImageMap engagements is like that of a "conductor of performances" for an orchestra, with the students contributing to an overall performance. In the ImageMap application (and especially the extended version described above), students contribute to a joint performance, verbally and with input technology, generating an overall aggregate representation, with a coherent visual gestalt. The teacher at-

tends primarily to group performance, not to each individual student. Moreover, the teacher, like the conductor, has responsibility for choosing and sequencing the material to be performed (the curricular activities), interpreting the performance, and guiding it toward its desired forms. As in rehearsal, the conductor might direct groups of students to practice something alone, or in small groups. During performance, the teacher will work to ensure that all parts are heard and that everyone gives their best performance—directing attention towards the students who need the most encouragement while keeping the overall performance moving forward.

3.4.3 Classroom Presenter: Flexible Presentations using Digital Ink

In moving from manual presentation systems (such as overhead projectors) to computer-based presentation systems (such as power-point slides), instructors saw some benefits and some losses. Computer based systems offer many conveniences: instructors can easily prepare their lectures in advance, switch back and forth between the presentation and other computer-based tools (including web applications), and save, re-use, and share their classroom materials. The trade-off, however, was a drastic loss of flexibility, as instructors could no longer annotate their presentations in writing to correspond with real-time events throughout lecture and discussion. Classroom Presenter is a tablet-PC-based application that combines the advantages of computer based presentation tools with the flexibility afforded by traditional systems. By using tablet-PCs as a platform for lecture presentations, teachers can use digital ink to write directly on their slides. In addition, Classroom Presenter supports multiple separate—but linked—views of the presentation: the teacher view, the projector view, and the students' views. This structure enables all students to have access to the instructors notes, enables students to beam their markings to the teacher and/or the projector, and also provides "private" annotation space for both students and the instructor on their own respective devices.

The affordances of the Classroom Presenter system support active and collaborative learning, student engagement, real-time feedback for the instructor of students' understanding, and the integration of student materials into classroom discussions. Thus far, reactions to the system have been highly positive: in a survey of students from over 200 science and engineering courses in which the instructor used Classroom Presenter, 55% of students said that the use of the system positively impacted their understanding of lecture material, and 69% of students would encourage other professors to use the system as well.

3.4.4 Boomerang: Capturing Student Generated Questions

Often when we approach learning, we think about teachers quizzing or asking questions of students. This activity orients students and teachers towards the body of knowledge for which the student is accountable. Student generated questions are also valuable for learning. However, standard classroom practice may permit only a small number of student questions, students with good questions often feel discouraged because other people's questions are so different from theirs and students often are discouraged by teachers who do not recognize the importance of question asking.

Boomerang is a tool designed to support students asking questions. It allows all students to submit questions privately which can then be posted and discussed by the group as a whole. When students ask questions in their own words, they reveal gaps in understanding that may not be elicited by the teachers' use of standard terminology and phrases. By asking questions, students not only fill in gaps in their knowledge base, but they also open up possibilities for wonderment.

3.4.5 Match My Graph: Language Games for Math Learning

One of the reasons that graphing calculators and SimCalc are so important is that middle school math students typically have difficulty remembering the meaning rep-

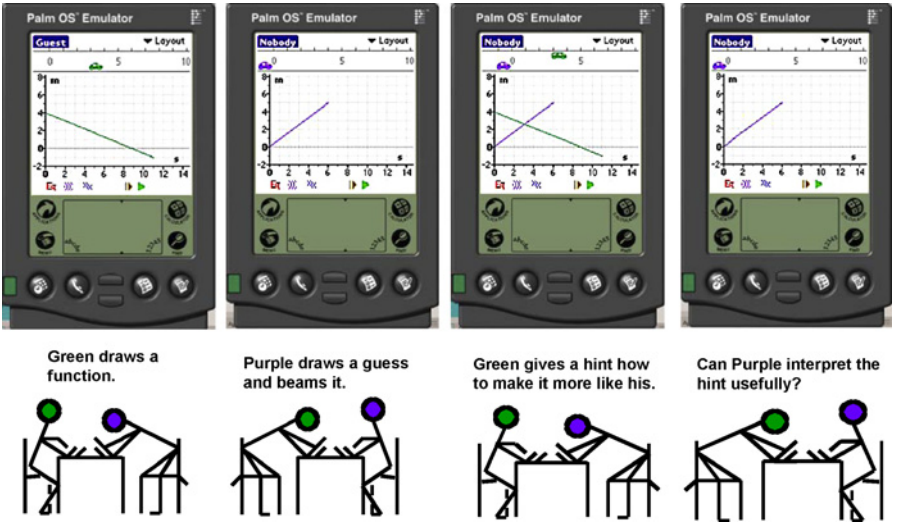


FIG. 1. Activity sequence in Match-My-Graph (reprinted from Tatar et al. [116]).

resented by a graph. Indeed, students may interpret a position graph that shows a line with an upward slope (see Fig. 1) as a representation of a car going up a hill. Another SimCalc variant, Match-My-Graph, targets the meaning of the graph by asking students to put it into words. Students work in pairs. One student, the grapher, draws a function over a domain. Only they can see their graph and its relationship to the motion of the car. The other student, the matcher, has the job of creating the same function over the same domain, by making successive guesses and interpreting hints. The matcher must use math language with sufficient care to convey precise meaning to the grapher. Over multiple rounds, each student takes turns as grapher and matcher. Several metrics of student engagement—including impermeability to distraction—confirm that students involved in Match My Graph activities are very focused on the task at hand.

An important part of Match-My-Graph is running the animation to understand whether a steeper graph represents a faster or slower car. What does an increase in the slope of the line mean for the animation? Four variations of this task stress the student's ability to create and interpret mathematical language. In the most complex version, a student looking at two velocity functions and no representation of the car motion must give hints to a student looking at a position graph.

Note that the teacher in this case is "untethered," roaming the classroom. She often observes groups and makes a decision about whether to engage with them more actively. Yet a challenge to this is her lack of direct access to the screen states of the participants. Sometimes students fail to realize what is important about their own activity, and therefore give accounts that ignore important phenomena. To support the teacher's observation, in the context of untethered teaching, work has been done exploring the parameters of sharing necessary to support a networked functionality "Look" which allows the teacher to gain a lightweight snapshot of a student's screen. In contrast to much work in Computer Supported Cooperative Work (CSCW) on the importance of shared screens, this work asks "how little knowledge of the visual workspace can still be useful to resolve differences?"

3.4.6 Data-Doers: Helping Students to be "Minds-on" During Hands-on Science Activities

In science classes, students face a number of difficulties in grasping scientific concepts and understanding hands-on demonstrations. Data-doers is a tool that aims to facilitate science learning by supporting students in being "minds-on when they are hands-on". Even though many teachers have near miraculous powers of knowing what is going on behind their heads, they cannot be everywhere in the classroom at once. Consequently, students engaged in hands-on work may sometimes become distracted or confused about the tasks at hand and their relationship to the larger point

of the lab experience. "What were those numbers?" "What was I supposed to do with them?" "What are we doing anyway?" Their confusion may last only a few minutes, and yet in the fast-paced world of the classroom, those few minutes can put them at a disadvantage.

Data Doers allows teachers to create handheld-based worksheets for labs or demonstrations to help students with data collection activities. Students can then work individually or in small groups, preparing materials that may eventually be shared in a class discussion. Data Doers reminds students to think about what they are doing in two *direct* ways:

(1) Based on teacher-set upper and lower bounds for measurements, it gives students feedback about when a result that they report is not plausible and needs to be reconsidered and possibly re-measured.
(2) It allows students to beam to their data, thus enabling them to make comparisons and contrasts more quickly.

It also provides more occasions for student thought in a three *indirect* ways:

(1) Teachers can collect student values and respond in a more timely fashion.
(2) Students will not be copying over data tables during class discussion, because they will either take the Handhelds home or take printouts home.
(3) Teachers can use the presentation of the Data Doers spreadsheet for a lab as a means of "pre-flecting," creating classroom discussion about the lab and its goals before it starts.

3.4.7 eXspot: Using RFID Technology to Enhance and Extend Museum-based Learning

Although the learning that takes places at science museums is typically highly motivating, it is also unstructured and unsystematic, and occurs in short spurts of engagement. Studies have shown that at museums such as the Exploratorium—an interactive, hands-on museum in San Francisco with over 100 exhibits covering a wide variety of topics—visitors typically spend only about 30 seconds at any given exhibit [56]. The key challenge for museum educators, then, is to find ways to deepen visitors' learning and extend their experiences, without shattering the motivation that comes from un-structured, self-directed exploration.

Researchers at the Exploratorium have been working on a new application called eXspot that uses RFID technology to support, record, and extend exhibit-based informal science-learning experiences. The eXspot system is supported by a wireless network linking stationary RFID readers to portable, individual RF tags. Visitors to a museum each receive an RF tag, which they first register (using an ID number and email address) at a registration kiosk. As visitors progress through the

museum, RFID readers stationed at each exhibit read RF tags that come within a certain range. Visitors can then use their ID information at museum kiosks or on home computers to login to personalized web-pages that capture their museum experience. Each visitor's museum trajectory, as well as photographs taken at various exhibits and links to additional resources and activities, appears on the personalized sites.

A bonus application of the eXspot system is that it can serve as an embedded evaluation tool for museum research. Information captured from RF tags as the visitors progress through a museum can provide valuable information regarding exhibit interaction patterns, use of RFID-supported exhibits, preferences for online content and learning activities, and repeat-visit information. All of this data serves to inform museum strategies and strengthen the learning value of community science museums.

3.4.8 Crickets

Crickets are tiny programmable devices that students can embed in their own physical creations. Using sensors and actuators, students can write simple programs telling Crickets how to behave. In this way, Crickets make it possible for students to control both the physical design for the structures and mechanisms of a creation as well as the computational design of behaviors [94]. In particular, they build on the notion of physical design (with blocks, tiles, Lego© parts, etc.) as a learning domain in which the consequences of design decisions and explorations are immediate and transparent. They then add to that context the ability to explore behavioral design in a similarly transparent fashion [95].

3.5 Discussion: Capabilities, Tensions, Potential Resolutions

Certain features emerge from this discussion: *a local messaging topology* among participants in mostly face-to-face settings; variations in *teacher control* and expectations for teacher behavior; the potential of *spatially directed communications*, for example, in beaming in Match-My-Graph, in the placement of probes in particular spatial locations in the stream, and implicitly by posting a mark in a certain place in image map assessment; in the predominance of *short, asynchronous structured data* over long general purpose text messages (such as in email) or long-term conversations (such as telnet or Napster); and in *aggregation* of data and experience by student and teacher. Finally, a *shared public display* is often important in making these aggregates available for discussion. We ask next how best to think about the status of each of these features. What factors, if any, should be treated as de-

sign principles? What factors are necessary for learning success and success in the world?

4. Overarching Ideas from the Learning Sciences

In discussing historical successes, we noted that successful applications of learning technology do not insert technology into schools in isolation from other factors. To the contrary, behind every successful learning technology we described an integration of simple, focused technologies and transformed teaching practices. We next considered emergent technological capabilities and applications. Describing the technology features of these examples is not enough because the designs have been informed by dual technological and learning perspectives. Readers would be unlikely to be able to generalize to powerful new designs of their own without also understanding the learning theories that informed each design. Therefore, in this section, we discuss recent advances in learning theory that have been most closely associated with design of networked, handheld learning technologies.

Recall from the historical examples that they drew upon the capabilities of networked computers to *represent* scientific and mathematical ideas in profoundly new ways, to provide enhanced *feedback* to teachers and students, and to enable students to engage in *inquiry* that is more aligned with scientific practice. The field of empirical research and theory that has been most closely associated with representational, feedback and inquiry approaches is called the "Learning Sciences." Although researchers have been active in this field for 20 years, they have only recently pulled together to form a society (the International Society of the Learning Sciences, http://www.isls.org) with two journals (*Journal of the Learning Sciences*, *International Journal of Computer-Supported Collaborative Learning*), and two important reference works (*How People Learn* and *The Cambridge Handbook of the Learning Sciences*).

In this section, we selectively highlight aspects of the Learning Sciences that we have found to be most relevant to the issue at hand: the design of handheld and wireless learning technologies to enhance school learning. We approach this task in three stages. First, we describe two overarching perspectives in the Learning Sciences, one that emphasizes cognitive augmentation and the other that emphasizes social participation. Many successful projects draw upon the tensions that lie between these perspectives. Second, we discuss a set of design factors that link with Learning Science theories and cut across many example applications. Third, we note that the Learning Sciences has been concerned not only with theory and results, but also with the practice of design. Hence our third section discusses the technology design practices emerging in this field.

4.1 Perspectives

4.1.1 Cognitive Augmentation

Inspired by Vannevar Bush's classic essay, "As We May Think",[2] Douglas Engelbart defined a program of research that linked advances in computing to a vision of augmenting human intelligence [36]. Engelbart's innovations—including the mouse, the use of graphical user interfaces to communicate ideas, and collaboration across a network—have become commonplace affordances of personal computing technology over the past 30 years. The power of cognitive augmentation is so strong that many of us would feel unable to work without our accustomed computational tools.

Engelbart's innovations also occurred in the midst of a cognitive revolution in psychology, which moved beyond earlier stimulus-response theories to focus on human problem solving as mediated by representations [81]. These mediating representations could be both knowledge within the mind and cultural symbol systems outside the mind, such as notations, diagrams, and visualizations. Computers turned out to be good both for testing theories of cognition (e.g. the field of Artificial Intelligence) and for creating novel symbol systems that could better mediate human problem solving performance.

The joint onset of the augmentive and cognitive revolutions had powerful effects on theories of learning. Before this time, learning was largely seen as a matter of applying effort to forming the right stimulus-response bonds (a theory called "behaviorism"). A student had learned something when they could quickly answer a question correctly. Many applications of computer technology in learning still build upon this old way of looking at things; these applications try to motivate students to put more effort into question-answering by making it more fun (e.g. by introducing a game-like reward structure). Applications of computer technology learning also reflect this old way of looking at learning when they see "interactivity" as a stimulus-response cycle: the student does something and the computer responds (or visa versa).

After the cognitive revolution, learning was seen as a process of transformation of a student's knowledge that is highly related to problem solving. A student now had to use knowledge in solving a complex problem to demonstrate learning. Further, "interactivity" now was described in reference to an extended problem solving process: how could "interacting" with a computer-based model or simulation (that itself was a dynamic system of symbols) be a powerful resource for a student to re-think their ideas about physics, for example? Computers now were seen as tools that could augment learning and problem solving processes by providing conceptual tools that were more appropriate to the learner's challenge [65].

[2] http://www.theatlantic.com/doc/194507/bush.

Arising within this history, key innovations in the learning sciences were cognitive tools that augmented the learning process. The programming language Logo provided "gears for the mind" ([85] which could make abstract geometric ideas more concrete (as instructions to a "turtle" to draw graphics) and help students learn new ways of thinking mathematically (such as recursion). At Xerox PARC (an institution that hired many people from Engelbart's lab), Alan Kay imagined a "Dynabook" which would enable students to manipulate scientific models and instantly see the consequences. Knowledge Forum, a collaborative learning tool for science classrooms, helped students to visualize and improve the structure of their ideas about a scientific phenomena. This transformed science classrooms from being more about memorizing facts (per earlier behaviorist theories) to being about scientific inquiry. Similarly, "The Adventures of Jasper Woodbury"—an early expression of Learning Science theories in multimedia format provided an extended context in which students could learn mathematical concepts while using those concepts to solve realistic problems.

As the cognitive revolution continued, much more was learned about representation. Of particular importance to learning is the transition from emphasizing general problem solving abilities to discovering that expertise depended on domain-specific knowledge and skills. Hence, a mathematician does not solve problems using the same representations or approaches as a biologist. Each field of inquiry has its own cultural tools and problem solving approaches that are particularly tuned to its subject matter. In Learning Sciences, this insight led to a search for ways of using computers to produce new ways of representing knowledge so that students might learn more easily. Many of the representations took advantage of the graphical, manipulable and dynamic aspects of modern computer technology, resulting in modeling, simulation, and visualization tools appropriate for learning a range of topics in mathematics, science and other subjects.

This history underlies the cases of handheld learning technology we have drawn out for consideration in this review. Graphing calculators, for example, provide students with a visual representation of functions that makes mathematical reasoning more fruitful for most students. A "killer application" of graphing calculators occurs in a typical calculus sequence when students need to understand that functions appear locally linear. By zooming in progressively closer in the graph of a function, students can experience local linearity in a way to that leads to great understanding of this fundamental idea within calculus. Simultaneously, graphing calculators offload laborious calculations, thus enabling teachers and students to allocate their cognitive resources to the more important aspects of the mathematics at hand. Within the newer applications we discussed, we also see a focus on handheld devices as representational tools that augment students' capabilities as they learn.

TABLE III
PRECOGNITIVE AND COGNITIVE PERSPECTIVES ON LEARNING TECHNOLOGY

	Precognitive perspective	Cognitive perspective
View of learning	Acquiring correct associations between a stimulus and a response	Transforming prior knowledge to be able to solve authentic problems in a domain of expertise
View of interactivity	Receiving quick and informative responses to each action can accelerate students' acquisition of appropriate associations	Interacting with a dynamic notation or simulation model during problem solving can lead students to transform their knowledge
Role of technology	Motivating students to apply more time and effort to learning through entertaining reward structures	Providing representations that are especially conducive to building a conceptual model of a particular area of mathematical or scientific investigation
Key features of technology	Allows designers to create a controlled sequence of interactions that optimizes formation of correct associations and motivation to apply effort to learning	Allows designers to create new forms of representation, using dynamic, manipulable graphic notations and models
Prototypical handheld example	Flashcards on a handheld allow students to practice associations anytime and anywhere	Graphing calculators enable students to explore local linearity of a function

Although we have been drawing a somewhat strict line between precognitive and cognitive approaches (see Table III for a summary), we should acknowledge that this is an oversimplification. The view of learning as embedded in problem solving and the role of technology as supplying cognitive tools for inquiry also goes back to seminal 20th century thinkers such as Dewey, Piaget, and Vygotsky—and these thinkers have figured large in the Learning Science approach to supporting learning through cognitive augmentation. Dewey, for instance, offers a clear philosophical account of what problem solving looks like. His account is helpful in distinguishing superficial problem solving (e.g. where a multiple choice test question is termed a "problem" if it contains a framing story) from inquiry-oriented problem solving. Dewey described the inquiry as being rooted in students' feelings of confusion and frustration with an inability to make something work, involving the application of both conceptual and physical tools over time to transform the students relationship to the situation, with a resulting cathartic realization of a new approach to the situation that removes the confusion and results in ability to work effectively in a range of similar situations. Hickman [52] gives a compelling account of Dewey as having foreseen the cognitive augmentation view of cultural tools. In learning, one role of

cognitive tools is to extend the time and intensity with which teachers and students are able to engage with complex and extended problem solving before frustration or confusion blocks progress.

Piaget also figures large in the cognitive augmentation view of tools for learning. In Piaget's theory, one important line of child development is from more embodied to more concrete to more abstract ways of solving problems. Learning technologists often look at this not as a fixed sequence but instead as a map of children's resources for learning; children have relatively few and fragile resources for learning through a purely abstract modality but relatively robust resources for learning through embodied and concrete experiences. Hence, one way to enable children to work more productively with abstract ideas (such as mathematical or scientific concepts) is to provide a range of well-designed concrete embodiments that bridge to the abstractions. The historical success of probeware, for example, can be seen as enabling students to participate in scientific inquiry by connecting to their ability to engage in embodied sensory exploration of the world. Likewise, cognitive tools running on handhelds (such as mathematical models) give students an opportunity to interact with a more concrete realization of abstract theoretical concepts.

We also note continuity from precognitive to cognitive perspectives. For example, our introduction emphasized the importance of frequent and integral access to technology. The theoretical roots of this concern reside in the concept of "time on task." Students learn more when they spend more time learning; Learning Scientists agree that arranging for teachers and students to spend more time engaged in learning is really important. Indeed, one of the drivers towards simpler technology is a desire to avoid introducing complexities (and logistic problems like running out of batteries) that steal precious minutes from engaged learning. Another area of continuity regards the importance of feedback. Both precognitive learning and cognitive augmentation perspectives highlight the importance of giving students timely and supportive feedback as they learn. The main difference, as would be expected, is whether the feedback is on the correctness of associations or related to a more extended problem solving process. In a learning sciences approach, "formative assessment" (the technical term for feedback) is often directed at helping students understand how experts judge the quality of explanations and problem solutions and at giving guidance to students on the fruitfulness of particular problem solving approaches.

4.1.2 Social Mediation of Participation

Like all revolutions, the cognitive revolution also had its excesses. Approximately midway in our timeline running from Engelbart's 1968 demonstration to the present, scholarly concern with the excesses of the cognitive revolution culminated in a

counter-revolution that stressed a more social and contextual view of learning. Some of the excesses that scholars raised included:

- A emphasis on problem solving in overly logical situations, like playing chess, rather than more common but fuzzy practical situations, like doubling a recipe.
- A tendency to treat context as yet more information to be represented, rather than a more phenomenological, embodied, situated account of context as cultural, historical and physical.
- A neglect of the social and collaborative dimensions of learning and an overemphasis on learners as isolated individuals.

In one way of viewing this history, the cognitive revolution supplied a thesis (the problem solving view of learning), which was then countered with an antithesis (a more social, cultural and situated view of problem solving). One can see the field of the Learning Sciences as the emerging synthesis which brought together these tensions, especially as applied to the design of learning technology. Therefore our narrative below explores the consequences of a social participation view of learning for the design of networked handheld applications.

One early and important center in which the social participation view of learning technology developed a particularly full theoretical and practical realization was the Laboratory of Comparative Human Cognition at the University of California, San Diego. Here, in work that started in 1972, Mike Cole and colleagues began attending to the problems students had learning school subjects in particular cultural settings, such as the problems Liberian students experienced in learning mathematics. But rather than developing a "deficit" view that emphasized cultural impoverishment, the social participation view often undertook anthropological studies to understand how children (or adult workers) in cultural settings develop powerful mathematical competence. This led to a long-term experiment in the development of after-school settings in which students use technology in support of learning, called the "Fifth Dimension." In parallel with the development of this technology-enriched setting, researchers developed a cultural–historical view of learning that drew upon Dewey, developmental psychology and especially the work of Russian psychologists such as Vygotsky. Below we describe some key features of the Fifth Dimension design and theory, especially as these features contrast with the cognitive augmentation perspective and lead to a broader Learning Sciences perspective (Fifth Dimension Clearinghouse website).

The Fifth Dimension was designed as an after-school setting for learning within community centers such as Boys' and Girls' Clubs, YMCAs, recreation centers, libraries, and public schools. A central element of the design was play; whereas most school settings beyond Kindergarten contain learning in a very formal structure, the

Fifth Dimension uses play as a primary activity system for developing mastery of a subject.

The heart of the Fifth Dimension is a wooden Maze divided into twenty rooms. Each room provides access to two kinds of activities, computer and non-computer. About seventy-five percent of the activities utilize educational software and computer games. Included are telecommunications activities for searching the Internet, and tools for computer-mediated, and video-mediated conferencing. The remaining activities are non-electronic and include board games and arts and crafts. The software represents the curricular content of the Fifth Dimension. Subject matter includes social development, communications, reading, writing, math, geography, social studies, health, technology, language, and problem solving. In all, the Maze contains over 120 educational and computer games and non-electronic activities (from http://www.education.miami.edu/blantonw/5dClhse/childs.html).

In addition to play, another critical aspect of the design was a sense of place. While the Fifth Dimension did not design for learning in formal school settings, it also did not design for learning in complete ad hoc settings either (as some researchers in handheld and networked learning do when they emphasize learning "on the bus" or "in the park"). The cultural and historical location of their design in places that were organized and designed for learning is critical to their successful design. Below we draw out three additional aspects of the Fifth Dimension that are central to the social participation perspective on the design of learning technology.

First, the Fifth Dimension was conceptualized as a set of activities in which students participate. This may seem superficial but is actually a deep point. The cognitive augmentation perspective often did not question the school-like nature of many of the tasks students were asked to do, and how that artificiality of school-like tasks prevents students from gaining a deeper appreciation and understanding of a subject. The point is theoretically deep because activity is a central construct in Russian psychology and a hinge-point in linking that theory to design. Activities are theorized as the minimal unit of meaningful engagement in a cultural practice, and include an analysis of goals, mediating tools, roles, and involvement of a community. Likewise, when Learning Scientists design applications of technology to learning, they do not merely design cognitive tools, but rather design coherent activities in which those technologies are used in social practice.

Second, the Fifth Dimension conceptualized a new role for a leader often called "the Wizard." In an allusion to the Wizard of Oz, the wizard only appears through the technology (and is a role often played by a group of people in a backroom). The wizard may act as a knowledge expert but may also be a prankster or curmudgeon who prods students' development forward. Theoretically, the wizard relates cultural prototypes of adults who guide and support learning but not merely through the narrow role that teachers often take, the role of a "sage on the stage." Likewise, in many de-

signs for the use of networked handheld in learning, the teacher is no longer expected to simply be the expert who delivers correct knowledge, nor is the device seen as an anytime/anyplace gateway to authoritative knowledge. In many successful designs of networked handhelds for learning, teachers are expected to prod students into more active roles in learning and the network enables students and teachers to enact rich social forms of participation in learning that go far beyond accessing "learning objects" on the web.

This brings us to a third point: the Fifth Dimension emphasized social participation of learners in activities. In particular, learners were expected to establish goals and strategies for their own participation, to collaborate in small groups, to reflect and communicate their emerging understandings, and to transition from roles as novices to roles as masters of their environment who could help others. Peer interaction was considered especially critical. Theoretically, this relates to the perspective that learning involves a transformation of participation in a community of practice. Complementing a knowledge-centered view, this view sees learners progress from "legitimate peripheral participation" in a cultural practice (such as making scientific arguments) to become more central participants in the real work of science and eventually to becoming masters who organize the learning environment for new students to sustain and grow the practice.

Drawing upon the work of Vygotsky, Learning Science researchers and technology designers often build on the idea that learners often first enact complex new practices in a social group and only later internalize the social performance as a cognitive capacity. In comparison to the cognitive view, in which cultural symbol systems primarily mediate thinking within an individual mind, the Vygotskian view sees the same cultural symbol systems as primarily mediating social practices of thinking—by allowing practitioners to think together more effectively. With respect to learners, Vygotsky conceptualized a "zone of proximal development" that was the conceptual area that lay between what a child could do individually and what they could do in a social group. Researchers could design mediating tools for the zone of proximal development to extend and deepen this zone to enable more fruitful social interaction around complex problems. Through social use of powerful mediating symbol systems, students could come to internalize expert use of those symbol systems and move along the trajectory toward expertise.

The contrast between the cognitive augmentation and social participation views of learning technology can clearly be seen in work around a modeling tool called the Envisioning Machine [99]. The Envisioning Machine provided a simulation environment in which students could learn Physics concepts by manipulating objects in a "Newtonian World" to match the motion of a ball in an "Observable World." Students used a mouse to drag velocity and acceleration vectors in the Newtonian World; when they started the simulation, the corresponding motion would be produced in-

cluding a display of the changing velocity vector over time. As a tool for cognitive augmentation, the Envisioning Machine was carefully designed to suggest and constrain students' thinking towards a vector addition model of kinematics—a powerful symbol system for reasoning about motion that is abstract and non-intuitive for most students. However, the eventual analysis of how students learned with the Envisioning Machine came to emphasize a social participation perspective. Students found the Envisioning Machine engaging because it encouraged a playful engagement with a core physics challenge—modeling the real world with theoretical concepts. As students participated in this activity, they initially could not see the symbol systems of the Newtonian World in the way an expert physicist does. However, the symbols mediated their use of language and enabled them to use available metaphors such as "pulling" to gradually construct a shared understanding of how the Newtonian World worked. This understanding was not identical to an expert view but constituted progress from very peripheral understanding of what physicists do and how they think towards a greater ability to participate in conversations with scientists about motion. One can see the Envisioning Machine not just as augmented cognition but also as mediating students' collaborative participation in scientific modeling practices—enabling them to engage in scientific modeling for a longer time period than they might have without the mediating tool and without each other.

Within the historical examples of networked handhelds introduced earlier, the case of networked response systems is the best fit to a social mediation perspective [88]. One can look at these systems merely in terms of providing more feedback to teachers and students. However, doing so misses a great deal of the action in successful classroom implementations. Students and teachers both report that the use of these systems changes how they participate in class. On the good side, students report less anxiety in sharing ideas in class; on the bad side, some students resent that their participation in class can now be measured by the teacher and adopt positions of resistance. Either way, these systems evoke strong feelings from the students about participation.

Penuel and his colleagues propose that concepts from the sociocultural theory of learning can explain how, when, and why audience response systems effectively transform and enhance learning [88]. First, the sociocultural perspective of learning posits that people learn when they practice using the tools of a given discipline, with the guidance of an expert. As people become more comfortable with these tools, their participation within the community of learning transforms, resulting in a transformation, too, of the interpersonal and group interactions that take place within the community of learners. Second, the sociocultural perspective argues that actions are mediated by discourse and symbolic representations. Extended practice at using the discourse of a discipline and at conceptualizing and communicating answers to open-ended questions aids students in developing deeper understandings of science

concepts. This emphasis on the importance of discourse provides an understanding of how and why the discussions resulting from the use of audience response systems are so influential in fostering increased conceptual understanding among students. Finally, a sociocultural perspective on student interest, motivation, and identity illuminates the ways in which audience response systems—and the features of the tasks and classroom dynamics associated with their use—might influence student engagement and student participation. Taken together, ideas from the sociocultural perspective of learning shed light on the mechanics of change taking place in classrooms that implement audience response systems.

4.2 A Learning Sciences Synthesis

Both cognitive augmentation and social mediation moved beyond the view of learning as mastery of a simple association of stimulus with response and the view of motivation as a simple impetus to put in enough effort to master the associations. Between stimulus and response, the cognitive revolution inserted mental representations. The social mediation view instead inserted cultural tools. While the theoretical differences between these moves is huge, the resulting learning technologies can look quite similar: a representational tool for augmenting mathematical cognition looks pretty much the same as a mediational tool for social participation in mathematical practices. Hence rapprochement and synthesis is achievable.

As one simple starting point, we can synthesize across settings. Researchers in the cognitive augmentation perspective have been more attuned to design tools for formal learning settings. Researchers in the social mediation perspective have been more attuned to the use of learning technology outside the formal school day. With the advent of handheld and networked technology, there seems to be little reason to make a sharp distinction between these two settings. Students will be able to carry their handhelds between formal classroom settings and settings in clubs, museums, and homes. In place of sharp distinctions, networked handhelds will likely encourage a sense of "seamless learning spaces" [17] that enable learning to move among and across different settings.

A synthesis can start with the view that the purpose of technologies in learning is to provide symbolic tools that enable students to engage effectively in extended episodes of thinking, communicating, and reflecting—not simply to make the mastery of simple associations more fun (see Table IV). A more cognitive view is often helpful in analyzing the relationship of a proposed tool to disciplinary knowledge: how does this tool embody or distort the ways in which mathematicians think about mathematics? A more mediational view is often helpful in analyzing the relationship of a proposed tool to disciplinary practices: how does this tool enable or block

TABLE IV
COMPARISON OF PERSPECTIVES ON LEARNING

Cognitive augmentation perspective	Social mediation perspective	Learning sciences perspective
Formal learning	Informal learning	Linking formal and informal learning
Symbolic tools (representations) mediate mental operations	Symbolic tools (representations) mediate social practice	Symbolic tools (representations) can fruitfully link cognitive and social dimensions of learning
Symbolic systems as the unit of design	Activities as the unit of design	Activities with symbolic systems as the unit of design
Successful learners are able to solve complex problems and undertake scientific inquiries	Successful learners are able to participate in disciplinary community of practice	Successful learners can participate in collaborative inquiry in a domain

students from participating in social practices which successfully approximate the practices of real scientists?

When designing a learning tool, designers working from a cognitive augmentation point of view often start by thinking about the symbol system that expert mathematicians or scientists use to solve problems. How could abstractions in the mind of experts become more tangible and concrete in the hands of learners? How could the abstract concepts be represented in a way that directs and constrains student thinking towards the ways in which experts think about the concepts? How could students attention be focused on the really important aspects of thinking in this domain, offloading less important, distracting parts to automation?

Designers working from a social mediation point of view start by thinking about designing an activity system that engages students in an approximation of the social practices of actual scientists and mathematicians. Designers think about appropriate goals for the activity and how technology can mediate students' involvement in it. What roles and rules will be necessary to structure playful engagement so as to enhance the learning potential of the activity? What new roles will teachers and other adults take? How does this activity fit as a transition between the cultural practices of childhood and the cultural practices of a specific discipline? How will it build children's affiliation with each other and enable them to see themselves moving along a trajectory towards greater participation in a community of practice?

In the Learning Sciences, these two design points of view come together by thinking about activities with domain-specific symbol systems as the fundamental unit of design. This synthesis balances the need to think about how technology can make an expert's abstract operations on symbols more tangible for learners and with the

recognition that the smallest meaningful chunk of social practice that can be designed is an activity.

Finally, each perspective offers a slightly different view of what success looks like. In the cognitive augmentation perspective, researchers celebrate success when students can solve complex problems or undertake scientific inquiries that reflect deep understandings of a particular domain of expertise. In the social mediation perspective, researchers see success more in terms of increasing ability to participate in the core practices of a professional community. A Learning Sciences synergy suggests seeing successful learning as an ability to participate in collaborative inquiry in a particular mathematical or scientific domain.

5. Design of Instructional Technologies

5.1 Design Factors

When discussing themes in the creation of historical and new handheld, connected activities, we identified local messaging, teacher-control, spatially-directed communication, short, asynchronous data messages and aggregation as factors. The crucial issue for computer scientists to bear in mind when designing handheld-based, wirelessly connected systems for learning is that, ultimately, nothing is more important than the learning. As in medicine, failure is a tragedy in the context of education, a waste of society's resources in general but a loss to every child who fails to learn what he or she ought. And the key to success in education is maintaining the primacy of the meaning of the experience to the learner over the design principle.

If we re-examine the technological features that we drew out as important before in the light of the knowledge of the Learning Sciences just presented, we see that four themes provide underlying causes for those features to attain importance: shared attention, rich representation, the role of public and private work, and the importance of control. Importantly, for example, local messaging is not significant in itself but rather because of the situation it creates in the design context. Local messaging is an important technique to consider insofar as it helps maintain shared attention among relevant participants and therefore promotes deeper interaction between peers or between student and teacher.

5.2 Design Practices

In designing handheld-based, wirelessly connected activities for the classroom, we must balance the considerations of producing an entire system, emphasizing user or

user group experience [117]. Stroup and Petrosino [112] characterize the classroom as an eco-system, and they rightly emphasize the limitation on and inter-dependence of resources.

Our goals as engineers must be not to set immediate boundaries nor to simplify the problem at all costs, but rather provide a "space of relevance," that is, a framework in which to think about the tradeoffs. Earlier, in discussing the technological context, we talked about the relationship between RF, IR, the market place, power usage, teacher control, connections to the Internet, and aspects of network performance at small scales. These are fundamentally incommensurate considerations. There may develop a societal consensus about how to handle them, but that consensus will just be another factor to take into account in the decision process that any project goes through.

This kind of design calls for a new paradigm for proceeding, a design tensions paradigm [117]. Like design rationale (Carroll; McLean) or scenario based design (Carroll and Rosson), design tensions focuses on reflection. However, unlike these paradigms, it does not carry the weight of capturing reflection in a form of interest to all subsequent designers. It attempts to assess the issues for the current design in situ. Our goals guide us through the design of handheld-based wirelessly connected systems, but we may not fully understand our goals until they run into conflict during the process of design. Thus we might hazard a guess that most designers have the twin goals of producing something useful in learning and having it adopted widely. Both of these goals might appear to converge on the choice of a particular platform for delivery because of its potential for widespread frequent integral use. However, if it turns out that the devices cannot be recharged in time to use in class after class, suddenly the creators must choose or find a third option. This approach to design differs from the usual engineering problem-solving in two small but highly significant ways. First, it emphasizes the integration rather than the separation of design decisions, suggesting that design decisions must be revisited during the design and building process in the light of subsequent discoveries. Second, as called for in the Value Sensitive Design movement [39,40], it focuses us on the values inherent in the design decision. The design of learning technologies is a high stakes, value laden enterprise.

One useful way of categorizing classroom systems is via the dichotomy of vertical vs. horizontal technology (Table V, adapted from [112]): tightly focused use (vertical) or general purpose use (horizontal). Other characteristics that appear to be tightly clustered with this characterization include whether it is designed primarily in reference to a teacher's pedagogical needs (vertical) or a student's personal needs (horizontal), whether it is designed to be a fixture of a classroom (vertical) or travel with the student from class to class (horizontal).

TABLE V
CATEGORIZATION OF CLASSROOM TECHNOLOGIES

Characteristic	Horizontal technology	Vertical technology
Designed for whom	Student	Teacher
Focus of functionality	Just-in-case	Just-enough
Physical movement between contexts	Portable across physical contexts	Fixed within use context (e.g. math classes)
Inter-device communication	Peer-to-peer/neighbor	Networked/flexible group
Content domain interaction	Interdisciplinary by nature	Domain-focused

However, while this table characterizes some portion of a state of affairs in the world, note that it does describe potential for new designs to solve the problem of what "good" is in the context of a particular project. Yet we have argued that values are central to the design process. Let us return to the examples of connected SimCalc and Match-My-Graph/NetCalc to illustrate how a designer might think about the different levels of decisions and goals in the system within the same project. Both the connected SimCalc and the Match-my-graph projects started with desktop SimCalc and its benefits. Indeed, they were funded by the same grant. As summarized in Table V, in other words, the projects had deep and broad agreement at the vision level. Project personnel agreed that the current state-of-affairs in the world was that desktop SimCalc was highly successful, that it could promote both excellence and equity in math knowledge by giving students a better grasp of the math of change and variation. They also agreed, at the vision level, that it ought to be cheaply available in a form that would get it to more students. The project approach, also agreed upon, was to utilize inexpensive networked, handheld computers, building on the example of graphing calculators. Furthermore, they agreed in their pedagogical commitment to designing for learning through supporting the use of multiple, linked representations and through promoting participation and feedback. These agreements have considerable consequence. For example, they suggest that there might be a minimum capacity for the display of representations. Cell phones, for example, fall under the "inexpensive" and "wide spread" criteria, but perhaps not the strong representation. I-pods (which were not yet known quantities) may have sufficient representational power, but not sufficient manipulation.

However, when it came to the level of project tensions, of what goals really to support, the project agreed to diverge (see Table VI). One portion of the project chose to emphasize an approach that maximized the installed base of particular technology (graphing calculators) over its display and manipulation capacities. The other portion chose to emphasize the display and manipulation capacities over the installed base by choosing PDA's. Both attempted to have some elements of the other. That is, the graphing calculator based portion of the project choose to work with high-end

TABLE VI
DIVERGENT APPROACHES TO SIMCALC RESULTING FROM PROJECT TENSIONS

Vision	Is: "SimCalc, when used well, teaches the math of change and variation, in a way that promotes both excellence and equity in math knowledge"	Ought: "It ought to be cheaply available to more students"
Approach	Project drivers: The state of cheap handheld devices; the prevalence of graphing calculators in schools already.	Values: Pedagogical commitments to multiple, linked representations; participatory and feedback tools; HCIcommitments to graphical-user-interfaces (GUI's).
Project tensions	Use PDA's with their larger screens and flexible input devices vs. use graphing calculators with their wide-spread prevalence.	
As created dilemmas	On-going usability challenges vs. a longer lead-time to demonstrations of effectiveness.	
	Tailoring of success to teacher controlled vs. student-controlled classrooms.	

calculators not currently possessed by every student. The PDA based portion of the project chose to go with low end IR PDA's, the least expensive on the market. While the power utilization of graphing calculators cannot be beat, the Palm IIIc used less power and lasted longer (more than 90 minutes of active use) than many of the alternatives.

Ultimately, both ends of the project were successful in developing usable learning systems. Neither was a waste of time nor bad. Both advanced the field. However, they were different in ways that must be anticipated and understood during the design process.

Working with SimCalc on a graphing calculator, with its small, low-resolution screen and repurposed buttons, presents on-going usability challenges. In part for reasons of pedagogical commitment by Hegedus and Kaput, in part because of the kinds of classrooms in which the graphing calculator version of SimCalc has been used, and in part because of these UI drawbacks, the graphing calculator version of SimCalc has emphasized the relationship between the individual and whole class activities. At the same time, the capacity to continue the research and for it to spread spontaneously by teacher downloading is a substantial advantage.

Working with PDA's led to the exploration of more forms of networked connectivity, of which Match-my-graph is one example. It led to a more attractive and more usable interface. It led to a focus on peer-learning, and to a different image of student-teacher relationships, the "untethered" teacher model. It also led to subsequent on-going research on the architecture of connectivity [66,11] and may in the long run feed back into the push into the classroom of devices that are more user friendly than graphing calculators.

6. Looking Forward

In this section, we examine some examples of devices that are not yet widely and effectively integrated into classroom learning. With respect to both meshing well with current practice and incorporating key lessons from the Learning Sciences, we consider the potential for new constellations of technology-enhanced learning to meet or surpass the level of success already experienced with the three historical cases.

6.1 I-Pods

Apple Computer's iPod digital music player, though it is early in its evolutionary sequence, illustrates the way in which changes in the technology ecosystem open niches for new species of technology. If the PalmPilot enables users to "break off a piece of their personal computer and take it with them in a very usable form," then the iPod enables users to "break off a piece of the Internet and take it with them in very usable form." In this case, the piece of the Internet in question is access to digital audio resources, such as mp3 files. The iPod is typically characterized as a personal device, and can be said to be very special purpose (in that its core function is centered on doing one thing: playing media). However, the fact that media are domain context neutral (an iPod plays pop music and recorded lectures on quantum physics equally well) opens the possibility that the iPod may expand to be a much more general purpose device than originally imagined.

Viewed with an eye toward its potential integration with existing practice in such a way as to incorporate learning science principles, there are a number of intriguing potential iPod applications to explore. In particular, modelling inquiry at the point of instruction both meshes well with current practice and has the potential of transforming practice through incorporating a rich, shared representation. For example, providing video snippets illustrating data collection procedures and issues *in situ* at the field site of a water quality experiment may be as convenient as providing written or oral directions but may also be far more effective in focusing discussion on the key inquiry issues of the experimental process.

6.2 Gaming Devices

In contrast to the I-pod trajectory, gaming devices such as Gameboy, PlayStation, Nintendo, Xbox, etc. are continuing to evolve toward extreme specialization. Many of these devices have hardware facilities that far exceed those of typical "horizontal" devices (such as laptops or personal computers), facilities that could, in principle, be employed in a much more general purpose way. However, the well-defined, but very

large, niche these devices occupy (and battle for dominance in) tends to preclude trading off any aspect of gameplay performance for either adaptation to current practice or incorporation of learning science results.

6.3 Mobile Phones

The evolution of mobile (cellular) phones is greatly complicated by the regulatory and infrastructure constraints of their ecosystem. Though they are clearly personal devices and could, in principle, be useful in a broad range of contexts, the supporting service model militates against their evolution toward utility in classrooms and other face-to-face contexts (since those uses would, sensibly, skip the trip to the cell tower and back.) But especially, the current inability of local institutions (such as schools) to control their usage in any kind of fine-grained way (the most common current control is to outlaw their use in classrooms) militates against any integration with current practice. The emergence of Wi-Fi-based mobile phones, for which the local wireless network infrastructure can be used as an alternative to carrier-provided infrastructure, could significantly alter this situation. Given that many of the applications described here might provide as much or more utility when implemented on mobile phones as on other classes of handheld devices, and that mobile phones are rapidly becoming the most ubiquitous of networked handheld devices, the emergence of locally controllable mobile phones could occasion a broad class of new, widely deployed, learning applications.

6.4 Projectors, SmartBoards, and Other Large Format Displays

In the context of technology for teaching and learning, a very interesting emerging genus of devices comprises stationary, shared, interactive displays such as projectors, SmartBoards, and wall-mounted display panels. In classrooms today, the image source for these displays is typically a laptop or personal computer and they are used as the shared focus of attention for the entire class. In this sense, they become a general purpose, hybrid system with a classroom, rather than personal, focus. Clearly, this breed of technology is not simply adapting to a pre-existing niche but also altering the ecosystem in a way that may allow novel hybrids to flourish (personal special-purpose devices sourcing the displays in aggregate; special purpose functionality like graphing built into the displays; etc.).

6.5 Laptops and Tablet PCs

Laptop computers and their Tablet PC variants provide an illuminating contrast with the other networked handheld devices considered here. In contrast to the vertical, special-purpose nature of the devices described above, laptops and tablets are archetypical general purpose, "horizontal" devices. In addition, rather than the device application evolving to meet a learning need, current educational practice is evolving in response to the inclusion of laptops in the classroom context ("now that every student has a laptop, what do we do with them?").

Except for the specific instance of enabling more, and more frequent, re-writing and re-visioning (for which there are successful, more "vertical" alternatives such as the popular AlphaSmart devices), effective learning applications of such general-purpose devices are not as well established as effective learning applications of more specific-use devices. That said, the direct and natural interaction through stylus drawing provided by Tablet PCs strongly suggests a rich realm for exploration of potential linkages with powerful learning experiences and careful meshing with current—and emerging—practice.

6.6 The $100 Laptop

The $100 laptop, devised by Nicholas Negroponte and his colleagues at MIT, is a response to concerns that technology is too often prohibitively expensive for widespread use in schools. Because it is still fundamentally a general-purpose, horizontal device, the $100 laptop fills a niche in the learning and technology ecosystem similar to that of regular laptops and tablet PCs. However, some important differences derive from the relative low-cost and enhanced durability of the $100 laptop. In particular, $100 laptops make it possible for students to not only work with technology on a 1 to 1 basis, but also to be in sole possession of the device, keeping the laptop with them at all times, both at school and at home. As a result, the $100 laptop has the potential to drive a rapid evolution of practices that capitalize on a strengthened home/school connection and a new bridge between informal and formal learning.

7. Conclusion

Handheld devices, especially networked handheld devices, are growing in importance in education, largely because their affordability and accessibility create an opportunity for educators to transition from occasional, supplemental use of computers, to frequent and integral use of portable computational technology. We have offered a view of this trend that is grounded in historical examples of success,

learning theory, an analysis of design tradeoffs and a discussion of design practices.

Our historical review highlighted three examples of handheld and wireless technologies that already have made a significant impact in school learning: graphing calculators, classroom response systems, and probes for collecting scientific data. Each of these has amassed a research literature providing evidence of a positive impact in education and detail on the factors contributing to implementation success.

The review and synthesis of learning theory highlighted two perspectives that can guide successful design of transformed classrooms, a cognitive augmentation perspective and a social participation perspective. We suggested that transformative uses of networked handheld devices will link in-school and out-of-school uses of technology for learning. Transformative uses will position the devices as a symbolic tool that fruitfully links social and cognitive dimensions of learning. To realize this potential, educators will design new forms of learning activity in which learners use symbol systems to participate in collaborative inquiry in a particular academic discipline.

Our analysis of tradeoffs argued that education has different requirements from consumer or enterprise markets and thus technologies that are successful in consumer or enterprise markets may not be a good fit to school markets. We note that all three historical examples of success were not consumer or enterprise products, but rather were specifically designed for the education market. Therefore, innovators should not think of educational handhelds as scaled-down computers but rather as specific appliances designed for a unique ecological niche. We called for designs of new ICT infrastructures that fit the needs of school. Some of these needs include: a local messaging topology among participants in mostly face-to-face settings; variations in teacher control and expectations for teacher behavior; the desirability of spatially directed communications; the predominance of short, asynchronous structured data over long general purpose text messages or long-term conversations; and aggregation of data and experience by student and teacher. A shared public display is often important in making these aggregates available for discussion. As there are no easy solutions, we suggested that a design tensions framework will be useful as designers work to realize the potential of new devices in enhancing the classroom experience.

We now conclude with a few comments on the challenge of scale. In nearly every country, improving the quality of education is seen as deeply linked to improving economic growth and the quality of life. Education, however, is a large-scale system which is slow and difficult to change. Furthermore, the history of attempts to use technology in the service of improving education is not a happy one. Simply put, most technologies have failed to make a noticeable impact in educational quality. Yet,

a few handheld technologies, such as the graphing calculator, are making a positive impact at scale (as affirmed by the correlation between graphing calculator use and mathematics achievement on the National Assessment of Educational Progress). We can make a difference with technology but it is never easy.

Here are a few ways in which handheld and networked technology could fail to enhance education. Wireless networks could prove too complex for schools to maintain or could fail to perform gracefully in conditions where 30 students suddenly ask for the same piece of data. New handheld devices could usher in a new age of incompatible operating systems (how will that iPod interoperate with that PDA and that graphing calculator) leading to a nightmare of system incompatibilities. The antisocial affordances of new technologies—allowing cheating, disruptive behavior, increasing student inattention to school tasks, or access to illicit materials—could outweigh the benefits in the eyes of parents, administrators and other school stakeholders. Purchasers could fail to demand integration of curriculum, assessment and teacher professional development with new technologies, and thus ignore the most important lesson of the past: that no technology improves learning in schools without substantial attention to these complementary components. Case studies could ignore the unusual extra resources made available in a school testbed and attempts to replicate and scale could fall apart in new locations that do not have these unique or extra resources.

We believe that networked handheld technology can overcome these potential downsides, but only if innovators keep the challenges of scale in mind as they design new technologies, activities, and school improvement plans. For a technology to work at scale, it must be quite simple and robust; it must tap complementary technical and social forces in learning; it must integrate with other drivers of school improvement (such as curriculum, assessment, and teacher professional development), it must make low demands on already over-taxed school resources and be effective despite variability among schools, teachers, and students. Interdisciplinary teams will be crucial to overcoming these obstacles and thus we invite technologists to join hands with educators and learning scientists in the quest for applications of handheld and networked technologies that can have a positive impact at necessary scale to improve the lives of children and teachers throughout our vast educational systems.

ACKNOWLEDGEMENTS

Special thanks to Corinne Singleton for her tremendous support in writing this article. This material is based in part upon work supported by the National Science Foundation under Grant Numbers #0205625 and #0427783. Any opinions, findings,

and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Abrahamson L., "An overview of teaching and learning research with classroom communication systems (CCSs)", in: *Samos International Conference on the Teaching of Mathematics, Village of Pythagorion, Samos, Greece*, 1998.

[2] Abrahamson L.A., Hantline F., et al., Electronic classroom system enabling interactive self-paced learning [patent #5002491], Patent and Trademark Office, United States of America, 1989.

[3] Abrahamson L.A., Davidian A., et al., "Wireless calculator networks: Where they came from, why they work, and where they're going", in: *13th Annual International Conference on Technology in Collegiate Mathematics*, Atlanta, GA, 2000.

[4] Bannasch S., "The electronic curator: Using a handheld computer at the Exploratorium", retrieved December 10, 2005, from Concord Consortium Newsletter. Web site: http://www.concord.org/library/1999fall/electronic-curator.html.

[5] Becker H.J., *Internet Use by Teachers: Conditions of Professional Use and Teacher-Directed Student Use*, Center for Research on Information Technology and Organizations, Irvine, CA, 1999.

[6] Belloti F., Barta R., De Gloria A., Ferreti E., Margarone M., "VeGame: Exploring art and history in Venice", *IEEE Computer* **36** (9) (2003) 48–55.

[7] Black P., Wiliam D., *Inside the Black Box: Raising Standards Through Classroom Assessment*, King's College London, London, 1998.

[8] Bobrowsky W., Vath R., Soloway E., "The Palm project: The impact of handhelds on science learning in the 7th grade", retrieved December 10, 2005, from the Center for Highly Interactive Computing in Education University of Michigan. Web site: http://hice.org/aera2004.html.

[9] Boyle J., "Using classroom communication systems with large classes", in: *Taking Advantage of Hand Held Technology and Calculator Network Workshop, University of Strathclyde*, 1999.

[10] Boyle J., Nicol D.J., *The Interactive Classroom and Studio (Powerpoint document)*, Strathclyde University, Glasgow, Scotland, 2002.

[11] Brecht J., Chaudhury R., Davis K., Digiano C., Patton C., Roschelle J., "Coordinating networked learning activities with a general purpose interface", paper presented at mLearn 2006, Banff, Canada, October 22, 2006.

[12] Bullis K., "A hundred-dollar laptop for hungry minds", *Technology Review* (2005, September 28), Retrieved from http://www.technologyreview.com/Infotech/14793.

[13] Burnstein R.A., Lederman L.M., "Using wireless keypads in lecture classes", *The Physics Teacher* **39** (2001) 8–11.

[14] Burrill G., Allison J., et al., *Handheld Graphing Technology in Secondary Mathematics: Research Findings and Implications for Classroom Practice*, Texas Instruments, Dallas, TX, 2002.

[15] Carroll J.M. (Ed.), *Minimalism Beyond the Nurnberg Funnel*, MIT Press, Cambridge, MA, 1998.
[16] Cattagni A., Farris E., "Internet access in US public schools and classrooms: 1994–2000", 2001. Retrieved July 15, 2006, from http://www.usdla.org/html/journal/JUN01_Issue/article04.html.
[17] Chan T., Roschelle J., Hsi S., Kinshuk, Sharples M., Brown T., Patton C., Cherniavsky J., Pea R., Norris C., Soloway E., Balacheff N., Scardamalia M., Dillenbourg P., Looi C., Milrad M., Hoppe U., "One-to-one technology-enhanced learning: An opportunity for global research collaboration", *Research and Practice in Technology Enhanced Learning* **1** (1) (2006) 3–29.
[18] Colella V., Borovoy R., Resnick M., "Participatory simulations: Using computational objects to learn about dynamic systems", in: *Human Factors in Computing Systems (extended abstracts), CHI '98, Los Angeles, CA, USA, April 18–23, 1998*, ACM Press, New York, NY, 1998, pp. 9–10.
[19] Consortium for School Networking, *A Guide to Handheld Computing in K-12 Schools*, Consortium for School Networking, Washington, DC, 2004.
[20] Cook T.D., Shadish W.R., "Program evaluation: The worldly science", *Annual Review of Psychology* **37** (1986) 193–232.
[21] Cortez C., Nussbaum M., et al., "Teachers' support with ad-hoc collaborative networks", *J. Computer Assisted Learning* **21** (3) (2005) 171–180.
[22] Crawford V., Vahey P., "Palm Education Pioneers Program: March 2002 evaluation report", SRI International, Menlo Park CA, 2002.
[23] Crouch C.H., Mazur E., "Peer instruction: Ten years of experience and results", *The Physics Teacher* **69** (9) (2001) 970–977.
[24] Cuban L., *Oversold and Underused: Computers in the Classroom*, Harvard University Press, Cambridge, MA, 2003.
[25] Cue N., "A universal learning tool for classrooms?", in: *First Quality in Teaching and Learning Conference, Hong Kong International Trade and Exhibition Center, HITEC, Hong Kong SAR, China*, 1998.
[26] Danesh A., Inkpen K.M., Lau F.W., Shu K.S., Booth K.S., "Geney: Designing a collaborative activity for the Palm handheld computer", in: *Proc. of the SIGCHI Conference on Human Factors in Computing, Systems, CHI '01, Seattle, WA, April 20–25, 2001*, ACM Press, New York, NY, 2001, pp. 388–395.
[27] Davis S., "Research to industry: Four years of observations in classrooms using a network of handheld devices", in: *IEEE International Workshop on Wireless and Mobile Technologies in Education*, *Växjö, Sweden*, IEEE Computer Society, 2002.
[28] Davis S., "Observations in classrooms using a network of handheld devices", *J. Computer Assisted Learning* **19** (3) (2003) 298–307.
[29] Davis R., Landay J., Chen V., Huang J., Lee R., Li F., Lin J., Morrey C., Schleimer B., Price M., Schilit B., "NotePals: Lightweight note sharing by the group, for the group", in: *Proc. of the Conference on Human Factors in Computing Systems, CHI '99, Pittsburgh, PA, May 15–20, 1999*, ACM Press, New York, NY, 1999, pp. 338–345.
[30] Dede C., "Planning for neomillennial learning styles", *Educause Quarterly* **28** (1) (2005) 7–12.

[31] Demana F., Waits B.K., "The evolution of instructional use of hand held technology. What we wanted? What we got!" in: *Proc. of the Technology Transitions Calculus Conference*, October, 1997.
[32] Demeure I., Faure C., Lecolinet E., Moissinac J.C., Pook S., "Mobile computing to facilitate interaction in lectures and meetings", in: *Proc. of First International Conference on Distributed Frameworks for Multimedia Applications, 2005, DFMA '05, Besancon, France, February 6–9, 2005*, 2005, pp. 359–366.
[33] Donovan M.S., Bransford J.D., Pellegrino J.W. (Eds.), *How People Learn: Bridging Research and Practice*, National Academy Press, Washington, DC, 1999.
[34] Dufresne R.J., Gerace W.J., Leonard W.J., Mestre J.P., Wenk L., "Classtalk: A classroom communication system for active learning", *J. Computing in Higher Education* **7** (1996) 3–47.
[35] Ellington A.J., "A meta-analysis of the effects of calculators on students' achievement and attitude levels in pre-college mathematics classes", *J. Research in Mathematics Education* **34** (5) (2003) 433–463.
[36] Engelbart D.C., *A Conceptual Framework for Augmenting Human Intellect*, SRI International, Menlo Park, CA, 1962.
[37] Fagen A.P., Crouch C.H., et al., "Peer instruction: Results from a range of classrooms", *The Physics Teacher* **40** (4) (2002) 206–207.
[38] Ferrio T., "What year did the graphing calculator get to scale?", 2004 (email correspondence).
[39] Friedman B., Kahn P.H., Jr., "New directions: A value-sensitive design approach to augmented reality", paper presented at the DARE 2000.
[40] Friedman B., Kahn P., Borning A., *Value Sensitive Design: Theory and Method*, University of Washington, Seattle, WA, 2002.
[41] Ganger A.C., Jackson M., "Wireless handheld computers in the preclinical undergraduate curriculum", *Medical Education Online* **8** (2003), art. no. 3.
[42] Gee J.P., *What Video Games Have to Teach us About Learning and Literacy*, Palgrave Macmillan, New York, 2003.
[43] Goldman S., Pea R., Maldonado H., "Emerging social engineering in the wireless classroom", in: Kafai Y., Saldoval W., Enyedy N., Nixon A.S., Herrera F. (Eds.), *Embracing Diversity in the Learning Sciences: Proceedings of the Sixth International Conference of the Learning Sciences, ICLS 2004*, Lawrence Erlbaum Associates, Mahwah, NJ, 2004, pp. 222–230.
[44] Graham A.T., Thomas M.O.J., "Building a versatile understanding of algebraic variables with a graphic calculator", *Educational Studies in Mathematics* **41** (3) (2000) 265–282.
[45] Hake R.R., "Interactive-engagement vs. traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses", *Amer. J. Phys.* **66** (1998) 64–74.
[46] Hartline F., *Analysis of 1st Semester of Classtalk Use at McIntosh Elementary School*, bE Inc., 1997.
[47] Hegedus S., Kaput J., "Exploring the phenomena of classroom connectivity", in: Mewborn D., et al. (Eds.), *Proc. of the 24th Annual Meeting of the North American Chapter*

of the International Group for the Psychology of Mathematics Education, 1, 2002, ERIC Clearinghouse, Columbus, OH, 2002, pp. 422–432.

[48] Hegedus S.J., Kaput J., "The effect of a SimCalc connected classroom on students' algebraic thinking", in: *Psychology in Mathematics Education Conference, Honolulu, HI*, 2003.

[49] Hegedus S.J., Kaput J., "An introduction to the profound potential of connected algebra activities: Issues of representation, engagement and pedagogy", in: *Proc. of the 28th Conference of the International Group for the Psychology of Mathematics Education*, vol. 3, 2004, pp. 129–136.

[50] Heller J.L., Curtis D.A., et al., *Impact of Handheld Graphing Calculator Use on Student Achievement in Algebra 1*, Heller Research Associates, Oakland, CA, 2005.

[51] Hennessy S., "The potential of portable technologies for supporting graphing investigations", *British Journal of Educational Technology* **30** (1) (1999) 57–60.

[52] Hickman L.A., *John Dewey's Pragmatic Technology*, Indiana University Press, Indianapolis, IN, 1990.

[53] Horowitz H.M., "Student response systems: Interactivity in a classroom environment", retrieved 3/17, 2003, from http://www.qwizdom.com/software/interactivity_in_classrooms.pdf, 1988.

[54] Howe N., Strauss W., *Millennials Rising: The Next Great Generation*, Vintage, New York, 2000.

[55] Hsi S., "I-Guides in progress: Two prototype applications for museum educators and visitors using wireless technologies to support informal science learning", in: *Proc. of IEEE International Workshop on Wireless and Mobile Technologies in Education, WMTE '02, Vaxjo, Sweden, August 29–30, 2002*, IEEE Computer Society, Washington, DC, 2002, pp. 187–192.

[56] Hsi S., Fait H., "RFID enhances museum visitors' experiences at the exploratorium", *Commun. ACM* **48** (9) (2005) 60–65 (Special Issue on RFID).

[57] Jipping M., Dieter S., Krikke J., Sandro S., "Using handheld computers in the classroom: Laboratories and collaboration on handheld machines", in: Proc. of the 2001 SIGCSE Technical Symposium, Charlotte, NC, February 21–25, 2001, *SIGCSE Technical Bulletin* **33** (2001) 169–173.

[58] Kaput J., "Implications of the shift from isolated, expensive technology to connected, inexpensive, diverse and ubiquitous technologies", in: Hitt F. (Ed.), *Representations and Mathematical Visualization*, Departamento de Matematica Educativa del Cinvestav, IPN, Mexico, 2002.

[59] Kaput J., Hegedus S.J., "Exploiting classroom connectivity by aggregating student constructions to create new learning opportunities", in: *26th Conference of the International Group for the Psychology of Mathematics Education*, Norwich, UK, 2002.

[60] Kasesniemi E.L., Rautiainen P., "Mobile culture of children and teenagers in Finland", in: Katz J.E., Aakhus M. (Eds.), *Perpetual Contact: Mobile Communication, Private Talk and Public Performance*, Cambridge University Press, Cambridge, 2002, pp. 170–192.

[61] Khoju M., Jaciw A., et al., *Effectiveness of Graphing Calculators in K-12 Mathematics Achievement: A Systematic Review*, Empirical Education, Inc., Palo Alto, CA, 2005.

[62] Kiili K., "Evaluating WAP usability: What usability?", in: *IEEE International Workshop on Wireless and Mobile Technologies in Education*, *Växjö, Sweden*, IEEE Computer Society, 2002.
[63] Kim K., Tatar D., Harrison S., "Handheld-mediated communication to support the effective sharing of meaning in joint activity", in: *WMUTE 2006, The 4th IEEE International Conference on Advanced Learning Technologies, Athens Greece, November 17–18*.
[64] Klopfer E., Squire K., Jenkins H., "Environmental detectives PDAs as a window into a virtual simulated world", in: *Proc. of IEEE International Workshop on Wireless and Mobile Technologies in Education, WMTE '02, Vaxjo, Sweden, August 29–30, 2002*, IEEE Computer Society, Washington, DC, 2002, pp. 95–98.
[65] Lajoie S.P., Derry S.J. (Eds.), *Computers as Cognitive Tools*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.
[66] Lin Y.-M., Laffey J., "Exploring the relationship between mediating tools and student perception of interdependence in a CSCL environment", *J. Interactive Learning Research* **17** (4) (2006) 385–400.
[67] Liu T.C., Kiang J.K., Wang H.Y., Chan T.W., Wei L.H., "Embedding EduClick in classroom to enhance interaction", in: *Proc. of the International Conference on Computers in Education, ICCE '03, Hong Kong, China, December 2003*, 2003, pp. 117–125.
[68] Luchini K., Quintana C., Krajcik J., Farah C., Nandihalli N., Reese K., Wieczorek A., Soloway E., "Scaffolding in the small: Designing educational supports for concept mapping on handheld computers", in: *Human Factors in Computing Systems (extended abstracts), CHI '02, Minneapolis, MN, April 20–25, 2002*, ACM Press, New York, NY, 2002, pp. 792–793.
[69] MacDonald M., *Using Technology to Assist Facilitation*, International Association of Facilitators, Williamsburg, Virginia, 1999.
[70] Mazur E., *Peer Instruction: A User's Manual*, Prentice Hall, Englewood Cliffs, NJ, 1997.
[71] McNairy W.W., "*PRS in the physics classroom: Implementation and course assessment correlations*", paper presented at the American Association of Physics Teachers, San Antonio TX, 2002.
[72] Mitnik R., Nussbaum M., Soto A., *Mobile Robotic Supported Collaborative Learning (MRSCL)*, *Lecture Notes in Comput. Sci.*, vol. 3315, November 2004, pp. 912–919.
[73] Moher T., Ding X., Wiley J., Hussain S., Singh P., Srinivasan V., Conmy D., "Combining handhelds with a whole-class display to support the learning of scientific control", in: *Human Factors in Computing Systems (extended abstracts), CHI '03, Ft. Lauderdale, FL, April 5–10, 2003*, ACM Press, New York, NY, 2003, pp. 882–883.
[74] Mokros J.R., Tinker R.F., "The impact of microcomputer-based labs on children's ability to interpret graphs", *J. Research in Science Teaching* **24** (5) (1987) 369–383.
[75] Moore G., "Electronics", April 19, 1965.
[76] Morgan M.E., Amend J.R., "Making chemical measurements using the lab works interface and a handheld graphing calculator", *The Chemical Educator* **3** (5) (October 1998) 1–7.
[77] Myers B.A., Nichols J., Wobbrock J.O., Miller R.C., "Taking handheld devices to the next level", *IEEE Computer* **37** (12) (December 2004) 36–43.

[78] Nachmias R., Linn M., "Evaluations of science laboratory data: The role of computer-presented information", *J. Research in Science Teaching* **24** (1987) 491–506.
[79] National Center for Education Statistics, *The Nation's Report Card: Mathematics 2000*, US Department of Education, Washington, DC, 2001.
[80] National Council of Teachers of Mathematics, *Principles and Standards for School Mathematics*, National Council of Teachers of Mathematics, Reston, VA, 2000.
[81] Newell A., Simon H., "GPS, a program that stimulates human thought", in: Feigenbaum E., Feldman J. (Eds.), *Computers and Thought*, McGraw–Hill, New York, 1963, pp. 279–293.
[82] Norman D.A., *The Design of Everyday Things*, Doubleday, New York, 1990 (paperback version of The psychology of everyday things, unchanged except for title).
[83] Oblinger D., "Boomers, Gen-Xers, & Millennials: Understanding the new students", *Educause Review* **38** (4) (2003) 37–47.
[84] Owens D.T., Demana F., et al., "Developing pedagogy for wireless calculator networks: Report on grants ESI 01-23391 & ESI 01-23284", The Ohio State University Research Foundation, Columbus, OH; Developing Pedagogy for Wireless Calculator Networks, The National Science Foundation, 4201 Wilson Blvd., Arlington, Virginia 22230, 2002.
[85] Papert S., *Mindstorms: Computers, Children, and Powerful Ideas*, Basic Books, New York, 1980.
[86] Pea R.D., "Seeing what we build together: Distributed multimedia learning environments for transformative communications", *J. Learning Sciences* **3** (3) (1994) 283–298.
[87] Penuel W.R., Yarnall L., "Designing handheld software to support classroom assessment: An analysis of conditions for teacher adoption", *J. Technology, Learning, and Assessment* **3** (5) (2005) 1–46.
[88] Penuel W.R., Abrahamson A.L., Roschelle J., "Theorizing the transformed classroom: A sociocultural interpretation of the effects of audience response systems in higher education", in: Banks D. (Ed.), *Audience Response Systems in Higher Education: Applications and Cases*, Information Science Publishing, Hershey, PA, 2006, pp. 187–208.
[89] Piazza S., "Peer instruction using an electronic response system in large lecture classes", Presentation document presented at the Pennsylvania State University Center for Education Technology Services "Teaching with Technology" series, Departments of Kinesiology, Mechanical Engineering, and Orthopaedics and Rehabilitation Center for Locomotion Studies, 2002.
[90] Poulis C., Massen C., et al., "Physics learning with audience paced feedback,", *American Journal of Physics* **66** (1998) 439–441.
[91] Pownell D., Bailey G.D., "Getting a handle on handhelds: What to consider before you introduce handhelds into your schools", Electronic School.com, 2001.
[92] Raghunath M., Narayanaswami C., Pinhanez C., "Fostering a symbiotic handheld environment", *IEEE Computer* **36** (9) (September 2003) 56–65.
[93] Ratto M., Shapiro B.R., et al., "The activeclass project: Experiments in encouraging classroom participation", 2002.
[94] Resnick M., "Computer as paint brush: Technology, play, and the creative society", in: Singer D., Golikoff R., Hirsh-Pasek K. (Eds.), *Play = Learning: How Play Motivates*

and Enhances Children's Cognitive and Social–Emotional Growth, Oxford University Press, 2006.
[95] Resnick M., Berg R., Eisenberg M., "Beyond black boxes: Bringing transparency and aesthetics back to scientific investigation", *J. Learning Sciences* **9** (1) (2000) 7–30.
[96] Rheingold H., *Smart Mobs: The Next Social Revolution*, Perseus Book Group, Cambridge MA, 2002.
[97] Rieger R., Gay G., "Using mobile computing to enhance field study", in: Hall R., Miyake N., Enyedy N. (Eds.), *Proc. of CSCL 1997, Toronto, Ontario, Canada, December 10–14, 1997*, L. Erlbaum & Assoc., Mahwah, NJ, 1997, pp. 215–223.
[98] Robinson S., *Discourse: Decades of Achievement Results*, Educational Testing Service, Princeton, NJ, 2002.
[99] Roschelle J., *Students' Construction of Qualitative Physics Knowledge: Learning About Velocity and Acceleration in a Computer Microworld*, University of California, Berkeley, CA, 1991.
[100] Roschelle J., Pea R., "A walk on the WILD side: How wireless handhelds may change computer-supported collaborative learning", *Internat. J. Cognition and Technology* **1** (1) (2002) 145–168.
[101] Roschelle J., Pea R., Hoadley C., Gordin D., Means B., "Changing how and what children learn in school with computer-based technologies", *The Future of Children* **10** (2) (2001) 76–101, Packard Foundation, Los Altos, CA.
[102] Roschelle J., Penuel W.R., Yarnall L., Tatar D., "Handheld tools that "informate" assessment of student learning in science: A requirements analysis", in: *Proc. of IEEE International Workshop on Wireless and Mobile Technologies in Education, WMTE '04, Taoyuan, Taiwan, March 23–25, 2004*, IEEE Computer Society, Washington, DC, 2004, pp. 149–153.
[103] K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences*, Cambridge University Press, New York, 2006.
[104] Scheele N., Mauve M., Effelsberg W., Wessels A., Horz H., Fries S., "The interactive lecture: A new teaching paradigm based upon ubiquitous computing", in: *Poster Proceeding of the CSCL 2003*, Bergen, Norway, June 2003, pp. 135–137.
[105] Schnase J., Cunnius E., Dowton S., "The StudySpace project: Collaborative hypermedia in nomadic computing environments", *Commun. ACM* **38** (8) (August 1995) 72–73.
[106] Schwartz J., "Professors vie with web for class's attention", New York Times, New York, 2003.
[107] Seeley C., "Technology is a tool: President's message", *NCTM News Bull.* **42** (7) (2006) 3.
[108] Sokoloff D.R., Thornton R.K., "Using interactive lecture demonstrations to create an active learning environment", in: Redish E.F., Rigden J.S. (Eds.), *The Changing Role of Physics Departments in Modern Universities: Proceedings of ICUPE*, The American Institute of Physics, College Park, MD, 1997.
[109] Soloway E., Grant W., Tinker R., Roschelle J., Mills M., Resnick M., Berg R., Eisenberg M., "Science in the palms of their hands", *Commun. ACM* **42** (8) (August 1999) 21–27.
[110] Soloway E., Norris C., Blumenfeld P., Fishman B., Krajcik J., Marx R., "Log on education: Handheld devices are ready-at-hand", *Commun. ACM,* **44** (6) (June 2001) 15–20.

[111] Steinkuehler C., "Learning in massively multi-player online games", in: *Proc. of the Sixth International Conference on Learning Sciences*, Lawrence Erlbaum, Mahweh, NJ, 2004, pp. 521–528.

[112] Stroup W.M., Petrosino A.J., "An analysis of horizontal and vertical device design for school-related teaching and learning", *Education, Communication & Information* **3** (3) (2003) 327–345.

[113] Stroup W.M., Kaput J., et al., "The nature and future of classroom connectivity: The dialectics of mathematics in the social space", in: *Psychology and Mathematics Education North America Conference*, *Athens, GA*, Educational Resources Information Center (ERIC), 2002.

[114] Tang J.C., Yankelovich N., Begole J.B., Van Kleek M., Li F., Bhalodia J., "ConNexus to Awarenex: Extending awareness to mobile users", in: *Proc. of Conference on Human Factors in Computing Systems, CHI '01, Seattle, WA, April 20–25, 2001*, ACM Press, New York, NY, 2001, pp. 221–228.

[115] Tapscott D., *Growing up Digital: The Rise of the Net Generation*, McGraw–Hill, New York, 1998.

[116] Tatar D., Roschelle J., Vahey P., Penuel W.R., "Handhelds go to school: Lessons learned", *IEEE Computer* **36** (9) (2003) 30–37.

[117] Tatar D., Vahey P., Roschelle J., "Design tensions in the creation of a handheld, networked application for teaching math", *J. Human–Computer Interaction*, 2006, submitted for publication.

[118] Tinker R., Krajcik J. (Eds.), *Portable Technologies: Science Learning in Context*, Kluwer Academic/Plenum Publishers, New York, 2001.

[119] Truong T.M., Griswold W.G., et al., *The ActiveClass Project: Experiments in Encouraging Classroom Participation*, University of California, San Diego, CA, 2002.

[120] Vahey P., Crawford V., *Palm Education Pioneers Program: Final Evaluation Report*, SRI International, Menlo Park, CA, 2002.

[121] VanDeGrift T., Wolfman S.A., et al., "Promoting interaction in large classes with a computer-mediated feedback system", retrieved 2/19, 2003, from http://www.cs.washington.edu/research/edtech/publications/aavwy02-cfs.pdf.

[122] Wang C.Y., Liu B.J., Chang K.E., Horng J.T., Chen G.D., "Using mobile techniques in improving information awareness to promote learning performance", in: *Proc. of Third IEEE International Conference on Advanced Learning Technologies, ICALT '03, Athens, Greece, July 9–11, 2003*, IEEE Computer Society, Washington, DC, 2003, pp. 106–109.

[123] Webking R., *Classtalk in Two Distinctly Different Settings*, University of Texas-El Paso, El Paso, TX, 1998.

[124] Weiner N., *Cybernetics, or Control and Communication in the Animal and the Machine*, MIT Press, Cambridge, MA, 1948.

[125] White B., "ThinkerTools: Causal models, conceptual change, and science education", *Cognition and Instruction* **10** (1) (1993) 1–100.

[126] Wilder Foundation, *Evaluation of Discourse in Saint Paul Public Schools*, Saint Paul, MN: Author (no date).

[127] Wilensky U., Stroup W.M., "Networked gridlock: Students enacting complex dynamic phenomena with the HubNet architecture", in: Fishman B., O'Connor-Divelbiss S. (Eds.), *Fourth International Conference of the Learning Sciences, Ann Arbor, MI, June 14–17, 2000*, Lawrence Erlbaum Associates, Mahwah, NJ, 2000, pp. 282–289.

[128] Woods H.A., Chiu C., "Wireless response technology in college classrooms", *The Technology Source* (September/October 2004).

[129] Zurita G., Nussbaum M., "Computer supported collaborative learning using wirelessly interconnected handheld computers", *Computers & Education* **42** (3) (April 2004) 289–314.

Interactive Explanatory and Descriptive Natural-Language Based Dialogue for Intelligent Information Filtering

JOHN ATKINSON AND ANITA FERREIRA

Department of Computer Sciences and Department of Linguistics
Universidad de Concepcion
Concepcion
Chile
atkinson@inf.udec.cl
aferreir@udec.cl

Abstract

Filtering systems and search engines have traditionally been used in a one-shot mode—the user types a query, a ranking algorithm returns the results, and the interaction ends. However, the input query is at best an imprecise description of the user's information need and that we must engage the user in a dialog to encourage an evolving understanding of what the user was looking for. In this work, a computational linguistics approach for interactive Web-based dialogue interactions aiming at intelligent web search and filtering is proposed.

The model focuses on the user's requests by automatically generating language-driven interactions which take into account the context, user's feedback and the initial web search's results. The different components for natural-language processing in the context of dialogue discourse interactions are described. The main results of a working prototype aiming to decrease both the number of conversational turns and the information overload are finally discussed.

ISSN: 0065-2458/DOI: 10.1016/S0065-2458(06)70002-X

1. Introduction

Users are usually confronted with rapidly increasing amounts of information as epitomized by the buzzword "information overload". While skills necessary for browsing individual websites seem to be available to users after only minimal training, considerably more experience is required for query-based searching and intersite navigation [24].

Much of the information on the Internet today consists of documents made available to many recipients through mailing lists, distribution lists, newsgroups, and the World Wide Web. Common to mailing lists and forums is that the originator of a message need only give the name of one recipient, the name of the group. The messaging network will then distribute the message to each of the members of the group, with no extra effort for the originator. The average effort of writing a simple message is about four minutes, and the average effort of reading a message is about half a minute, so if there are more than about eight recipients to a message, the total reading time is larger than the total writing time, and if there are hundreds or thousands of recipients, the total reading time caused by the originator is many times larger than his effort in writing the message. Because of this, Internet users will easily become overloaded with messages, documents, etc. This issue can also be seen as a quality problem: people want to read the most interesting documents, and want to avoid having to read low-quality or uninteresting messages. Filtering is tools to help people find the most valuable information, so that the limited time spent on reading/listening/viewing can be spent on the most interesting and valuable documents. Filters are also used to organize and structure information.

Future software for the Internet can be expected to employ more advanced and user-friendly filtering functions than today, in order to support less computer-specialist users. Since people download millions of web documents every day, and very often do not immediately get what they would mostly like to get, the gains through better filtering are enormous. Even a filter with a 10% efficiency gain, the gain would be worth billions of dollar a year.

However, filtering is not possible without having suitable search facilities hence search engines are a central part of information access on the Internet. Their efficient

use requires sophisticated knowledge. Investigations on the search behavior of both expert and novice Web users have several practical applications. First and foremost, a model of search behavior can serve as the basis for improving interfaces and functionality of existing search systems. The varied needs of experts and novices can be identified and considered by more sophisticated future systems. Also, help-systems and Internet education can also benefit from a better understanding of users' difficulties with the search process [23].

State-of-the-art keywords-based information search systems can provide us with a fair first approach to efficiently searching on the Web. However, a major challenge is how to perform this kind of tasks accurately and smarter in order to make good use of user's knowledge (i.e., intentions, goals). This aims to improve the searching capabilities with a minimum of user–system interactions.

In order to address some of these problems, this work explores the generation of natural-language interactive dialogues for bibliographical searching on the Web aiming to assist the searching/filtering process with minimum user exchanges. The approach focuses on enhancing the whole information searching paradigm with both a computational linguistics model and a more suitable search agent to filtering. Accordingly, the following issues are addressed:

- Decreasing information overload in searching for information involves "filtering" it in an intelligent way in terms of the context and the user's underlying knowledge.
- Making good use of the user's feedback/knowledge can provide us with more accurate information being delivered.
- Taking into account the linguistic knowledge as main working support can assist us in specifying and restricting the real user's requirements and so capturing user knowledge which is unlikely to be obtained by other means.

Our approach's backbone is made of task-dependent discourse and dialogue analysis capabilities as a part of a major interactive searching system which a user interacts with. While the original approach and its implementation was carried out to deal with Spanish dialogues, we provide a model which can easily be adapted to other languages providing that right grammars and pragmatic constraints are considered.

Our experiments, conducted in the context of a filtering system for Web documents, demonstrate the promise of combining *Natural Language Processing* (NLP) techniques with simple inference methods to address the information overload problem [15].

In what follows, we first motivate our work by discussing previous related work. Next, the novel features of our model for intelligent search and filtering along with the analysis methods and used representation are described. Finally, an analysis of

some experiments in the context of web-based natural-language search and filtering is discussed.

2. Related Work

In the user modeling community, the behavior of Web users has attracted some attention. In [28], for example, Bayesian networks are constructed to model the successive search queries issued by users of a search engine. Augmenting the search engine logfile with manually assigned categories of presumed information goals they are able to predict query modifications. Similarly, [53] propose the use of Markov models to predict a Web user's next request based on the timing and location of past requests. However, these studies do not address personal characteristics of a user's level of expertise.

Many current search engines use automated software which goes out onto the web and obtains the contents of each server it encounters, indexing documents as it finds them. This approach results in the kind of databases maintained and indexed by search services such as *Google*. However, the problems which users can face when using such databases are beginning to be well documented:

- *Relevant Information*: from the retrieved information (i.e. references to documents, snipets), users can spend a long time trying to check whether these results contain what they have been looking for.
- *Information Overload*: the amount of information and wide coverage is often so huge that most of it is usually abandoned.
- *Representation*: the information search is based on documents being represented as "bag of words" and consequently all the further similarity computations are restricted to this kind of user's query.
- *User's Feedback*: there is no interaction with the user in the searching process in order to check whether her/his requirements have been fulfilled. It is assumed that all the information is useful, but it is unclear how.

It seems clear then that users are constantly facing a lot of obstacles in order to effectively search for and filter information on the Web. An analysis of the main issues ranging from experience-related problems (i.e., "*how do I retrieve an existing page*" ') to design issues (i.e., "*the browser is either poorly designed or difficult to effectively use*") can be seen in graph of Fig. 1. Here it can be observed that a significant proportion of web users takes a long time to search for relevant information (documents, webpages, etc.).

Issues in the second and third place in the graph, are directly related with the difficulty to retrieve useful and understandable information. In order to address this,

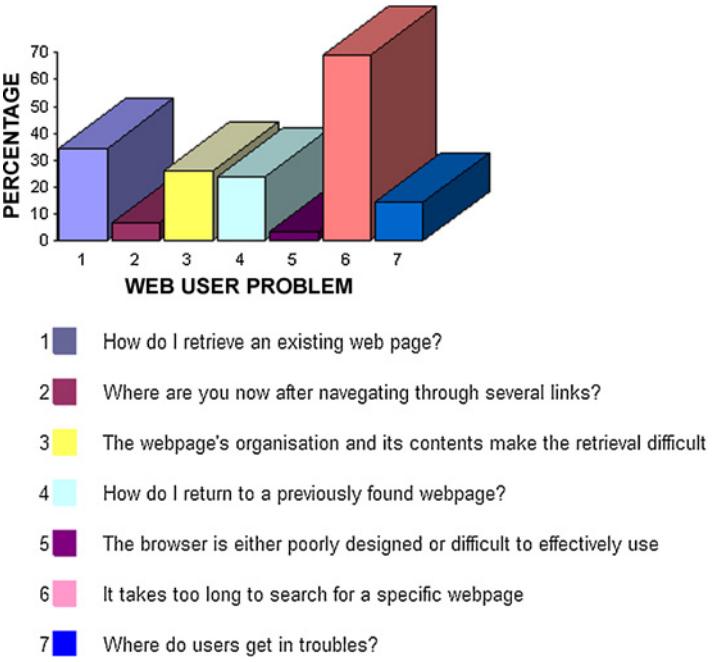


FIG. 1. Web search users' main issues.

two specific problems can be identified:

1. Current search systems are not capable of understanding a web user's behavior, intentions and/or profiles so as to learn from it.
2. A bag-of-words representation used by current search engines seems rather restrictive. Many search systems are unable to capture further underlying relevant information. Users often communicate through implicit discourse-level knowledge which is not actually plausible to be acquired by traditional search systems [4,25].

The first problem is related to adaptation capabilities of a search system, whereas the second one involves the query representation, the interaction with the users and the ability of the system to capture underlying (and implicit) information expressed in natural language.

Intelligent searching agents have been developed in order to address some of the issues above [29]. These agents can use the spider technology, and employ this in new kinds of ways. Usually, these tools are "robots" that can be trained by the user

to search the web for specific types of information resources. The agent can be personalized by its owner so that it can build up a picture of individual profiles or precise information needs. An intelligent agent can also be autonomous—so that it is capable of making judgments about the likely relevance of the material. Another key feature is that their usefulness as searching tools should increase the more frequently they are used. It will learn from past experiences, as a user will have the option of reviewing search results and rejecting any information sources which are neither relevant nor useful. This information will be stored in a user profile which the agent uses when performing a search, and enable the agent to learn from its initial forays into the web, and return with a more tightly defined searching agenda if requested.

Many search systems lack a deeper linguistic analysis of the user's requirements and context to assist him/her in getting a more specific view about what he/she really wants. In order to select and reduce the number of relevant documents obtained from an Information Retrieval (IR) system, Information Filtering (IF) techniques come to place [48]. These can usually be divided into two groups:

1. Automatic filtering is where the computer evaluates what is of value for you.
2. Social filtering (also known as collaborative filtering) is tools where other people help you evaluate what is of most value to read. Just like the publishers and organizations did in society before the Internet.

The most successful social filtering system is *Yahoo* which employs humans to evaluate documents, and puts documents, which are interesting into its structured information database.

The simplest and most common filtering is by organizing documents into groups (newsgroups, mailing lists, forums, etc.) Each group has a topic, and wants only contributions within that topic.

In general, filtering is done by applying rules to attributes of the documents to be filtered. Filtering rules are usually Boolean conditions and often put in an ordered list, which is scanned for each item to be filtered. The order of the items in the list can sometimes influence the outcome of the filtering, in ways, which the user does not understand well. The attributes of documents, to be used in filtering, are words in the titles, abstracts or the whole document, name of author, and ratings on the documents supplied by its author or by other people.

The most common way of delivery of filtering results is that documents are filtered into different folders. Users choose to read new items one folder at a time. Thus, the filter helps users read messages on the same topic at the same time. The user can also have a personal priority on the order of reading documents in different folders. Most filtering services deliver new documents with a list, from which the user can select which items to read or not to read. The user act of selecting what to read from such a list can also be seen as a kind of filtering.

Automatic filtering has been successful only with very simple filters (i.e., spam filters in which a system discovers attributes of an email message and determine relevant messages from spam). Advanced methods for "intelligent" filtering have in general not been very successful. Intelligent filtering is a complex task requiring intelligence which computers are maybe not yet capable of. By intelligent filtering is meant use of Artificial Intelligence (AI) methods [7] to enhance filtering. This can be done in different ways: AI software can be used to derive attributes for documents, which are then used for filtering, it can be used to derive filtering rules, or it can be used for the filtering process itself. With the machine learning approach, the filter will take as input information from the user about which documents the user likes, and will then look at these messages and try to derive common features of them to be used in future filtering.

Such filtering can be done in the background, behind the scenes, with little or no interaction with the user, or it can be done in a way where a user can interact with the filter and help the filter understand why the user likes certain messages. A disadvantage with much user interaction is that it takes user time, and the whole idea of filtering is to save user time. A disadvantage with very automatic filtering is that the user may not trust a filter if the user does not understand how it works.

Several approaches to filtering have been proposed by understanding the documents and their semantics, including *Oval* which uses a keyword-based strategy, *Foltz* which uses Latent Semantic Analysis [27] and collaborative recommenders [22] to filter news articles, *INFOSCOPE* which uses rule-based agents to watch the user's behavior and then to make suggestions, MAXIMS, for collaborative electronic filtering [31,32], WebWatcher, for Web filtering system which learns user's preferences and highlights interesting links on web pages that it visits, and *CinemaScreen*, for collaborative and content-based filtering [41]. Both cognitive and social approaches are suitable approaches to extract documents. Whereas a social filtering is more appropriate when the information obtained is used to keep it updated in some environment, a cognitive approach is better when the gathered information is based on some specific topic, regardless who the users are. Accordingly, learning and adaptation capabilities become more important on an IF context rather than IR because of the underlying environment's features: IF systems are used by huge groups of users, most of who are not motivated information searchers, and so their interests are often weakly defined and understated [38,29].

In order to acquire suitable profiles, current IF systems allow users to specify one or more sample documents as reflective of his/her interests [47], instead of requiring direct explicit definition of interest, whereas others attempt to learn the profiles from the user's behavior. Note that this approach may not be very practical as users do not often focus on their real goals.

As users struggle to navigate the wealth of on-line information now available, the need for automated *Question–Answering* (QA) systems becomes more urgent. We need systems that allow a user to ask a question in everyday language and receive an answer quickly and succinctly, with sufficient context to validate the answer. As previously discussed, current search engines can return ranked lists of documents, but they do not deliver answers to the user. Question–Answering systems address this problem. Recent successes have been reported in a series of question–answering evaluations that started in 1999 as part of the Text Retrieval Conference (TREC) [21,50]. The best systems are now able to answer more than two thirds of factual questions in this evaluation.

Question answering has many applications. We can subdivide these applications based on the source of the answers: structured data (databases), semi-structured data (for example, comment fields in databases) or free text. QA systems address the problem of using linguistic processing into different levels to assist the answer retrieval. However, there is no dialogue at all and the effort focuses on getting precise documents rather than capturing the user's needs.

While the main purpose of QA systems is to extract the correct answer for a question, a lot of sample information needs to be provided in advance. In addition, QA systems know the kind of questions the user can make, there is a predefined set of target documents where the answers are supposed to lie, there is a few constraints which restrict the scope of the likely questions, etc.

Since most of the QA task is fixed, there is only one possible answer for every question and there is no feedback at all. That is, one question must get one likely answer, so one can not go back to refine the *Natural-Language* (NL) question.

To some extent, QA systems succeed to extract "deeper" knowledge from user questions. However, there are no facilities to interactively generate explanatory requests to have users more focused on their interests [36,3]. Although some research has been carried out using NLP technology to capture user's profiles, it has only been in very restricted domains which use *WordNet* or more primitive resources [5].

In other dialogue-related approaches using NLP such as PIES [39], a user's interests and goals are obtained by providing relevance feedback. However, unlike natural-language interaction systems [20], the aim of PIES is to analyze the user's profile to prune part of some "story" in order to reach a final answer.

Approaches making further use of the user's knowledge can be used by enabling other NLP tasks. In particular, *Natural Language Generation* techniques can be used to allow the system to produce good and useful "dialogue" with the user [34,35]. An important aim at this stage is to decrease the number of generated conversation turns in order for the user to obtain the information (i.e., references to documents) the user is looking for. Some research approaches involve producing some kind of dialogue

interaction as a whole structured into different levels [40] in which communication constitutes an indirect action to accomplish the goals.

2.1 Natural-Language Generation

Over the last years, research and practice of Natural-Language Generation (NLG) systems has strongly evolved towards discourse processing and establishment [40]. Generally speaking, NLG research has traditionally been divided into two levels: content generation and surface generation. A clear problem with this is that for the same high-level proposition, different sequential realisations can be performed. In addition, selecting a particular choice is often dependent on the global structure of discourse. Reiter [40] addresses this kind of problem by establishing an intermediate level called *Sentence Planning*. Furthermore, a three-level architecture (see Fig. 2) is suggested to describe properly the entire process of NLG, in which:

- *Content generation* (*also known as content determination*) is responsible to determine the content and the rhetorical planning tasks (i.e., the way the content of appropriate referring expressions can be worked out).
- *Sentence Planning* (also known as text structuring) involves designing suitable strategies to transform abstract semantic representations into constraints on candidate sentences.

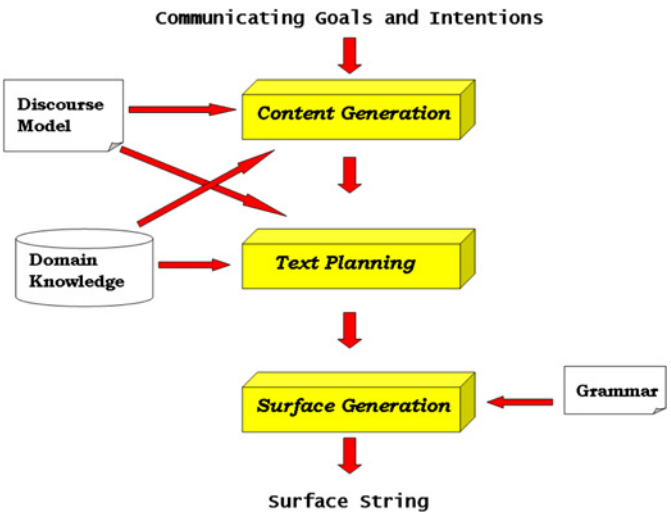


FIG. 2. Typical architecture for NLG.

- *Surface Generation* (*also known as realization*) produces (realizes) a sequence of words representing the surface structure of the utterance from the determined content of the sentences.

Note that in this model, content generation and sentence planning levels share knowledge about the domain and the discourse structure. Accordingly, actual developments of NLG systems include these elements focused on three dimensions:

1. *Generation tasks*: these involve grammar issues to determine how to say something, what to say (content selection) and why to say something (the motivation driving the decisions).
2. *Generation methods*: these have traditionally focused on four different modes: selecting predefined sentences with no changes, selecting predefined items to allow some variation (i.e., templates), selecting patterns that are gradually instantiated until the full text is produced, and selecting individual features from the target text which are combined together to build an specification for the generated text.
3. *Generation Records*: these represent and record the sensibility of an NLG system based on the topic, the interpersonal roles and relations, and the situation of the communication.

One of the most important NLG tasks, *Text Planning*, is based on the fact that human beings "plan" their utterances before saying them. Planning is strongly motivated by the Speech Acts Theory [2] which has been widely introduced to describe a computer system which produces a plan containing a sequence of Speech Acts [8,9,46].

Components of planning tasks usually consider the state of the world, beliefs, domain actions, communicating actions, direct speech acts, goals, and operators [40]. Several successful NLG systems have been built using plan-based strategies [1]. However, the different actions to meet the communicating goals of a NLG system can be realised into different ways ranging from hierarchical planning to reactive planning [36].

Text planning on dialogue contexts has traditionally been tackled from two perspectives. Firstly, a dialogue theory has been developed to include computational and linguistic foundations for goal-oriented dialogues [45] in which participants communicate based on some goal-driven tasks. Secondly, the planner focuses on the participants' mental states and the context required to enable a cooperative dialog [10,3].

Despite the successes, computational processing of dialogues has proven to barely be driven by a similarity with human dialog. For instance, question–answering systems and frame-filling dialogues are just simplifications for human dialogue in which a user makes guided questions and the system generates short answers. Dialogues

between humans are known to be wider and include clarifications, confirmations, reformulations and other communicating actions [14]. Thus, computer modelling of dialogue [52,30] has usually been approached as follows:

1. **Dialogue Grammars** use finite-state phrase-structure grammars on which dialogues are seen as a sequence of adjacent speech acts.
2. **Dialogue Plans** are based on the underlying assumption that utterances are not only word sequences but also speech acts update actions, such as illocutionary acts (i.e., request, inform, suggest, etc). These communicating actions are part of a plan aiming to change the hearers' mental states. Thus, it is the speaker's responsibility to uncover and respond to this plan.
3. **Dialogue as Cooperation** involves shared tasks to be carried out by interacting agents [17,26,35] rather than the effect of interacting plan generators in harmony. At the very least, dialogue participants are required to commit themselves so as to understand each other.

Dialogue discourse is commonly described following a *Communication-based Approach*. Here, the discourse's structural components are established from communicating goals. The focus here is on identifying interrelations between specific components and their uses as planning operators. This approach is strongly due to Mann's early work on the *Rhetorical Structure Theory* (RST) [33,45] being one of the most influential approaches for NLG over the last decade.

Typical interactive cooperative dialogue systems provide general, descriptive answers along with explanations about their answers. Grice's maxims of conversation namely the quality, quantity, relation and style maxims [16] are frequently used as a basis for designing cooperative answering systems. To address these behaviours, specific cooperative techniques have been developed to identify and to explain false presuppositions or various types of misunderstandings found in questions.

Agents participating in a cooperative dialogue share information about the discourse's status, this is, the situation in which the agents themselves are situated. This status usually includes actions performed by the agents [51], the conversation's topics, and their mental states (i.e., beliefs, intentions) [26]. Cooperative collaboration between dialogue agents can involve simultaneous, joint or sequential execution of actions [17]. For this, each agent must be provided with mutual beliefs about others' intentions and capabilities and the way these intentions and actions support the overall goal. Plan representation containing this information is referred to as a *Shared Plan* [26].

NL systems have a passive role in a user-oriented dialogue so these must recognise the user's plans in order to provide answers from a databases or knowledge-based system. However, in recent experimental studies, NLG systems are seen as those having the control of the conversation (i.e., system-oriented dialogue). Thus, when-

ever the system is acting as an expert advisor, it should provide explanations in order for the user to understand and accept the system's advice. In this kind of scenario, a user has (almost) no knowledge on how to carry out the task, so consequently the user communicates her/his plans and the system drives the conversation. On the other hand, in mixed-initiative dialogues [6] the tasks are assumed broad and with no structure. For instance, in air ticket reservation, system-oriented dialogues are not suitable as there is no schemata to suggest the system the sub-tasks a user want to perform and the sequence of these.

2.2 Interactive Discourse Planning

Unlike the traditional discourse models [17], Haller [20] proposes a model for *Interactive Discourse Planning* (IDP) in which a plan captures the discourse's intentional and attentional structure. In addition, information on the relevant objects, properties and relationships at each stage of the discourse plan execution is obtained. Because of its nature, IDP represents the discourse as a structure of events and distinguishes two different rhetorical relations for two kinds of text acts. These can then be used to achieve the discourse's goals as follows:

1. *Discourse Text Acts* (*DTA*): define both rhetorical relations for presentation and Searle's Speech Acts. DTA are executed to get into a state in which the hearer has an attitude (or capability) as defined by the system's discourse goal. Usual implementations of IDP consider four DTAs: recommendation, advice, motivation and granting.
2. *Content-Selection Text Acts* (*CTA*): define operators for content-selection text acts with two types of constraints: constraints on active content's goals and constraints on the domain. The first type restricts the plan and effects deduction of the CTS to those cases in which there is an active content's goal (i.e., the effect of the CTA). The second one involves restrictions on the domain which in turn are satisfied by the IDP planner by applying domain rules. These rules ties the "topic" rhetoric relation to the domain's beliefs.

Recent efforts for implementing IDP strategies include a modified version of *SNePS* (Semantic Network Processing System) [44], an integrated approach for inference and acting that uses a common representation for beliefs, plans and acts, all of which are thought of as entities structured in a semantic net.

A first major modification aims to select criteria for generating a text plan from specific text acts. In *SNePS*, the plan satisfying the higher number of content's goals is selected. A second feature involves looking ahead for the effect that each text plan will have on the hearer's knowledge so to use it for identifying candidates for a hearer's future implicit questions. IDP also makes use of the user's feedback to guide

further discourse plan expansions based on the success or failure of the speaker's communicating plan.

3. A Model for Interactive Web-driven and Dialogue-based Search

In this section, a novel approach to intelligent search and filtering on the web using natural-language feedback is described. For this, some of the previously discussed issues are addressed and suitable strategies are designed in order to capture underlying knowledge from a web user and from the search system. The approach incorporates intelligent agent technology and NLP methods to exploit knowledge regarding the context, interests and goals, and the user's behavior along an adaptive dialogue. The model's backbone is made of task-dependent discourse and dialogue analysis capabilities as a part of a major interactive searching system which user interacts with.

Instead of providing samples or going through the Web looking for the information, search requirements are focused by using a dialog-based search system to learn a user's specific interests. This allows the search system to obtain implicit knowledge so as to refine and filter the search results as the dialogue goes on.

Our model is designed to meet a variety of requirements. This may be scalable to large collections of documents. A system using this approach should be able to quickly determine the user's needs through a combination of user provided information and feedback [11], all of these based on NL interactions.

The overall approach to searching and filtering using NL feedback is shown in Fig. 3. The operation starts from NL questions provided by the user and then passed on to a discourse processing phase which generates the corresponding interaction exchanges (turns) using NL utterances. As the dialogue goes on, the system generates a more refined question which finally is sent to a search agent. The search results are delivered to the user providing that these have been appropriately filtered. This will depend on the previous interactions, the user's context and the features extracted from the queries.

3.1 Experimental Methodology

In order to gather linguistic data so as to build a dialogue model, an experimental technique called *Wizard of Oz* (WOZ) was used [25].

During each interaction, a human (*the Wizard*) simulates a system interacting with the users. We make them believe that they are interacting with a system which handles natural language in a real way. Then, the dialogues are recorded, annotated and

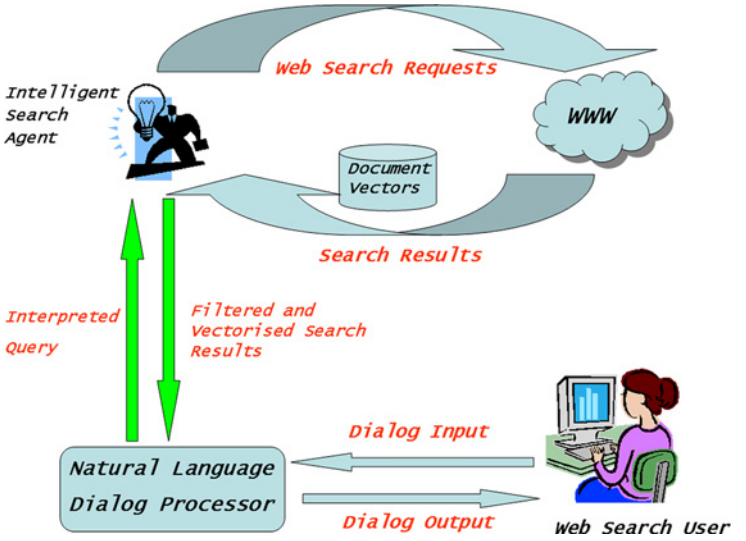


FIG. 3. The overall search-driven NL dialogue agent.

analyzed with the ultimate goal of improving the initial dialogue model and therefore, the interaction. The interactive process continues until the model meets the expectations and the design requirements.

In the actual experiments, WOZ has been used to get a significant dialogue corpus which allowed us to analyze the transcriptions and to establish a dialogue structure based model that will support the planning and generation of interactive *explanatory* and *descriptive* discourse [12,18].

A WOZ simulator with search capabilities was implemented and used by a group of human subjects who were required to search for information on the Web. The interactions between users and system (actually, the hidden expert who interacts with the users) were automatically recorded and used for linguistic analysis purposes. A group of subjects was then required to search for information on the Web using this tool. Thus, the communicating situation using the simulator becomes a three-step activity:

1. A user gets involved in a dialogue with the computer in order to make her/his NL question more refined.
2. The dialogue is monitored by the human expert, and as the dialogue proceeds, the answers are manually built.
3. The answer is sent to the user.

Based on the available information, user and system are able to continue the refining process until the user's need is met or the user retracts from searching any longer.

In practice, a threshold of 20 minutes was considered to check for the user's communicating goal accomplishment with a total number of 20 non-expert sample subjects being involved. The sample was then divided into four groups, in which the first three groups were randomly selected whereas the fourth one was constituted by graduate students of linguistics. Furthermore, they had to perform the search and then to provide definitions for "explanation" and "description" of what users are reading, based on the obtained search results. The aim of the model to be built is to be capable of generating either descriptive and explanatory discourse depending on the context, situation, and search's results.

To this end, the groups were requested to perform the following tasks:

- The first group was required to "talk about" the search's results (i.e., *Could you talk about your search's results*?).
- The second group was required to "describe" the search's results (i.e., *Could you describe your search's results*?).
- The third group was required to "explain" the search's results (i.e., *Could you explain your search's results*?).
- The fourth group was required to do the three above tasks and to answer some questions stated by the system, including:
  - *What do you understand by explaining or describing*? *Give me an example*.
  - *What do you understand by describing the search's results*?
  - *What do you understand by explaining the search's results*?
  - *What is the difference between describing and explaining the results*?

In order to finally validate whether the subjects were able to establish differences in producing their discourses (i.e. explaining, describing), some few advanced students of linguistics were requested to perform the different tasks but for the same session (full details of the experiments can be found in [13]).

In addition to the transcriptions and the produced dialogue structures, several parameters which guide both explanatory and descriptive discourse about the search were produced. These included documents' date, language, source place, kind of WWW page (i.e., research, business, etc.) which were also used in developing the NL generation module.

Descriptive texts were usually located at the beginning of a narration and they are organised into four dimensions: spatial, sensorial, scientific and formal:

- *Spatial*: objects are localised up, down, from inside to outside, or vice versa, from global to details, etc.
- *Sensorial*: different sensations are provided: what is seen, what is heard, what is touched, etc.
- *Scientific*: each human expert has a descriptive schemata of objects in his/her domain (i.e., a plant is not described as a rock).
- *Formal*: a description can be organised according to objects, classes and properties.

In the context of NLG, descriptive texts bring different definition processes into scene aiming to characterise actors, places or processes. In our model, a formal schemata was used to represent descriptive discourse texts (Fig. 4) which is composed of a topic, its parts and its properties. This assumes that it is possible to find sequences of descriptive parts in a narrative text. For instance, there is a huge number of texts on the Internet, whose global function is to describe something so we have a descriptive text or discourse.

The strategy for constructing descriptive propositions can be seen in Fig. 5. Anchoring techniques generate individual-argument ties for predicates obtained either from verb aspectualisation (*Pd.PROPerty* and *Pd.PART*), or from association assignment (*Pd.SITuation* and *Pd.ASSociation*).

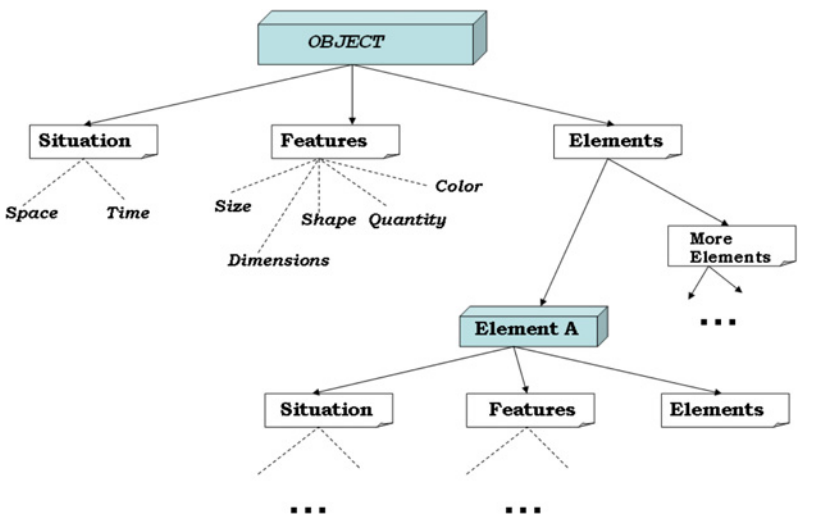


FIG. 4. Formal schemata for descriptive texts.

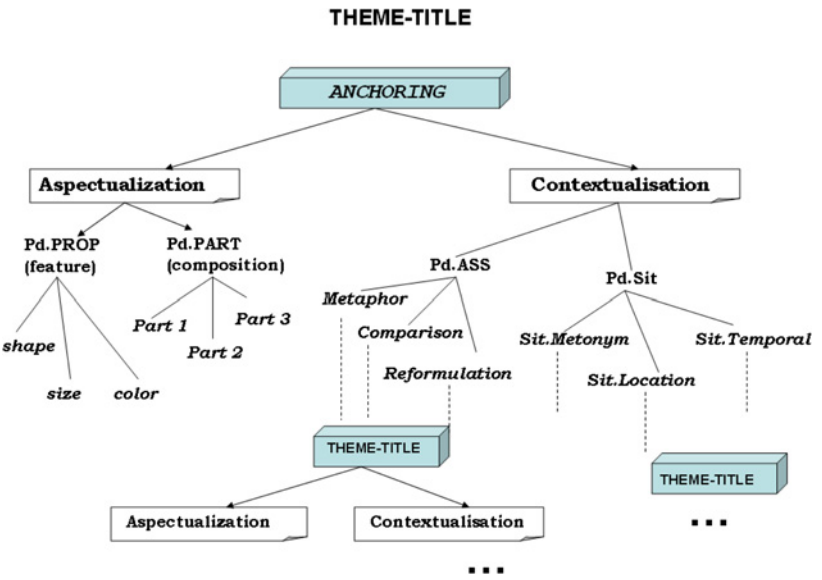


FIG. 5. Macro-structure for descriptive texts.

A descriptive macro-proposition (*Pd*) defines a direct relationship between topic and title which is taken as the macro-proposition's individual-argument. The selected individual's property (i.e., predicate) can usually be due to:

- Quality properties: *Pd.PROP* which can be spatial (*SIT.Location*), temporal (*SIT.Temp*), etc.
- Parts or relationships with the whole (*Pd.PART*)
- A situated event: *Pd.SIT* which can be comparative (*Comp*), metaphoric (*Meta*), or reformulation (*Ref*).
- An assimilation or association (*Pd.ASS*).

Based on these features of a descriptive narration, four underlying operations can be identified for the description's structure:

1. Detecting the subject of the description's topic (i.e., hyperonym): the topic is usually located either at the beginning of the text (i.e., title), or at the end of the text expressing a problem's solution.
2. Enumerating the parts or aspects of an object.
3. Characterising the parts and the object itself.
4. Expanding characterisation by comparing, metaphoring or reformulating.

On the other hand, in order to give account for the generation of explanatory texts, our model is strongly based on Moore's approach to explanatory dialogues [37,36]. Moore focuses on textual explanations given by expert and advisory systems, and she discusses the explanation component of an expert system that advises the user on how to improve their programming skills. However, it is clear how the basic ideas and techniques presented apply to a wide range of applications in which explanations need to be provided, such as help systems, online documentation systems, tutorial systems, and so on.

In our model, explanatory text generation is an essentially interactive process, requiring a dialogue between the system giving the explanation and the user receiving the explanation. Without such a dialogue the chances of the advisee obtaining the information required, and in a form they understand, are much reduced. In order to participate effectively in such a dialogue the system is able to respond appropriately to some follow-up questions after the initial explanation is given. This requires that the system understands the context of these questions, and in particular the context created by the system's previous responses. Accordingly, explanatory text generation has knowledge of how to explain. This includes knowledge of how explanations are structured (discourse structure) and how particular explanation strategies are used to achieve communicative goals. Thus, the main focus here is on how communicative goals can be linked to rhetorical or coherence relations, and how that provides the basis for a principled approach to the planning of coherent texts to address our particular communicative goal.

The developed explanation generation system uses an RST-based text planner to construct explanatory texts, given a particular communicative goal and assumptions about the user. The text plan makes explicit the goal behind the explanation (and the subgoals behind its different parts), the way the different parts of the explanation are related (by rhetorical relations), and any assumptions about the user that are made when planning the explanation.

3.2 Interactive Natural-Language Dialogue Generation

The descriptive and explanatory dialogue generator is based on a number of stages which define the context, the participants' knowledge (user and system) and the situation in which the analyzed dialogue is being conceived (i.e., interaction to search for information on the Web). This also considers a set of components which input and output is based on different stages of linguistic and non-linguistic information processing tasks established from the dialogue.

This phase is strongly motivated by a linguistic model for discourse processing proposed by [49,42] regarding the design of components of interaction and action.

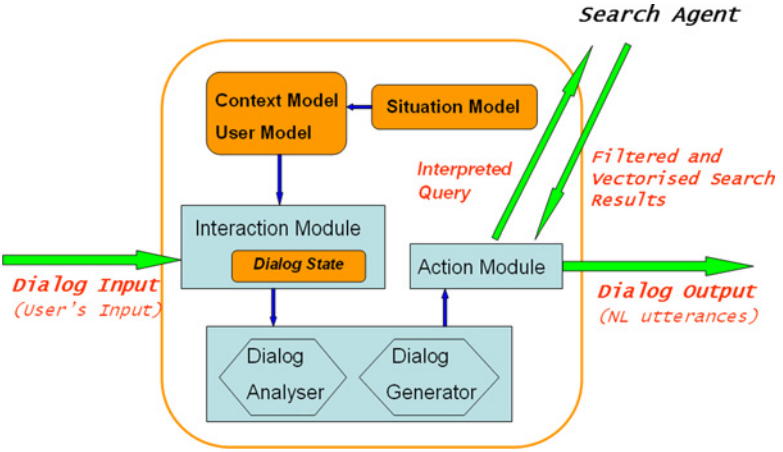


FIG. 6. The interactive natural-language dialogue processor.

In Fig. 6, the design for the proposed Web-based natural-language dialogue model is shown. This generates discourse outputs from the results of a bibliographic search so as to guide further dialogue interactions [13,11,19]. This starts with the user's NL question and produces either an output consisting on a NL conversation exchange to guide the dialogue or a search request being passed on to the search agent.

In order to better understand the approach, the underlying generation model has been divided into different components (Fig. 6) and described as follows:

- **The Context Model** deals with the information regarding the dialogue's participants, this is, the user who needs information from the Web and the system which performs the search. The model states the kind of social situation called "*bibliographical queries on the Web*" and the participants' goals: "*find out information about some topic*" (user) and "assist the user to achieve her/his goal through searching and collaborative dialogue" (the search system). The **User Model** includes knowledge about the user with whom the system interacts. The user model's outcome will become the type of question which the system expects as input (i.e., a kind of bibliographical NL "query" on the WWW).

  Information regarding the communicative situation's characteristics in which the dialogue discourse is embedded, is established on the **Situation Model**. Due to the nature of the model (i.e., enabling communicating interaction), conversations are restricted to requirements and constraints requested by the search agent. This involves using both some dialogue state records and utterance struc-

tures (lexical, syntactic, semantics, pragmatics) so to have the dialogue represented.

- **The Interaction Module** is based on Grice's cooperative principle and collaborative maxims [16] and involves two-position exchange structures such as question/answer, greeting/greeting, etc. These exchange structures are subject to constraints on the system's conversation, regarding a two-way ability to transmit through the keyboard, suitable and understandable messages as confirmation acts. Constraints have also to do with the Speech Acts chosen at some time given of the dialogue process.

  All this information and that related to the different interactions between system and user during the dialogue is stored by a *Dialogue Recording Module* aiming to keep the dialogue coherence between the system and the user's input.

- **The Dialogue Discourse Analyzer** receives the user's question and analyses the information contained in order to define the conditions which can address the system's response generation. This module's outcome is the NL question both recognized and analyzed by the system. Furthermore, recognition and interpretation is controlled by two main analysis modules which process the linguistic knowledge and interact with each other: semantics and pragmatics analyzers.

  Semantic macro-structures (see Fig. 5), which constitute the **semantic module**, allows for the selection of an argument from some property or action associated to them. Thus, defining the propositions is guided by the modules of interaction and action, and so its outcome is an appropriate semantic proposition for a given Speech Act generated from the pragmatic analysis stage.

  An embedded *Pragmatic Analyzer* establishes a Speech Act suitable to both the dialogue structure's component and the stated constraints. This concerns the information coming from the *situation model* and the *Context Model*. In addition, it involves the semantic content suggested by the *Semantic Analyzer* and the discourse coherence based on the *Interaction Control*, so its outcome is the previously defined speech act suitable for the current communicating goal.

  Each obtained pragmatic function makes explicit some specific Speech Act. In addition, each semantic proposition defines the speech act's semantic content. This content is expressed as a set of semantic functions such as agent, event, object, instrument, etc. (see ontology in Fig. 4).

- **The Dialogue Generator** involves the information from the search agent's *information recording module* and that coming from the *dialogue recording module* to produce a coherent utterance on the current dialogue sequence. As a

first output, the module generates a question to the user about the information needed to produce the corresponding utterance to the dialog's conversational turn.

Dialogue begins by generating the kind of utterance "query about information requested by the user" (very general at the beginning). Next, the system considers two possible NLG outputs: an specific question for the communicating situation (`what topic do you want to search for`?) and a general one on the context of the different kinds of information available on the Web (`what kind of information do you need`?). The different kinds of user's requests can then be divided into four general groups: requests for information, positive/negative confirmation, specification of features, specification of topic.

The discourse analyzer processes the user's NL input and gets the information needed for a search agent. From the obtained information (i.e., references to documents), the NLG task is capable of guiding the dialogue towards an explanatory NL generation based on basic criteria identified in preliminary experiments.

As the dialogue goes on, the discourse generator produces its output (NL sentences) based on the results of the search, context information, and the user's feedback. In order to establish the starting point for the NL generation process, high-level goals have been identified. However, since the generation is fully unstructured and too wide, some decisions must be made to better guide the generator from pragmatic levels down to lexical levels. Accordingly, the search agent's actions depend on the results of the search:

1. More than 100 documents were found (i.e., the NL question is too broad).
2. Less than 100 documents were found.
3. Different languages were detected in the retrieved documents.
4. More than 30 and less than 100 documents were found.
5. Less than 30 documents were found (i.e., right enough to display the results to the user).

Next, the discourse analyzer makes some single decisions based on the number of keywords which express the user's topic of interest and their context, among other issues. Then, the NL generator can guide the dialogue towards an explanatory generation or descriptive generation according to the following two simple criteria:

1. If the search agent retrieves more than 100 documents, the system should answer the user question by using the explanatory generation task.
2. If the search agent retrieves less than 100 documents, the system should generate a descriptive utterance.

Explanatory utterance generation aims to get the user more focused either by requesting a more detailed specification of the question ("*Your question is too general,*

could you be more specific?") or by requiring the user to specify some feature of the topic being consulted ("*I found N references to documents about that topic, in which one are you most interested*?").

The discourse analyzer again performs the analysis on the user's specific input in order for the agent to perform the search. In doing this, the information recording module stores the question specification or the thematic topic provided by the user. Then, the agent locally searches again for the information on the requested topic. Now, the generation leads to descriptive sentences. From the information obtained from the intermediate search results and the user's context (confirmation, negation, request, etc), the discourse generator produces three kinds of descriptive utterances providing different choices to the user, including:

1. Providing the results in a specific language. For example: "*the information is written in different languages, do you prefer them in English*?",
2. Displaying all the obtained documents. For example: "*There are twenty items about this topic, do you want to check all of them*?",
3. Displaying the results according to some frequent parameter. For example: "*I found information about research groups, courses and personal pages, are you interested in something specific*?".

The corresponding search action is performed by the **Action Module** in Fig. 6 (also known as *Action Selection Component*) which receives the information analyzed by the dialogue analyzer, so the recording module stores the response provided by the user. This allows the system to perform the corresponding action and to keep the dialogue coherence.

At this point, the (discourse) analyzer processes the user's response in order for the NLG task to produce an output confirming or expressing the action being performed (i.e., "*Did you find what you were looking for*?").

In order to verify whether the communicating goal has been achieved, the analyzer processes and analyses the input. If a positive response is obtained, the system generates a sentence to give the user the opportunity to choose another topic so to perform a new search. Otherwise, the option of searching for another topic related to the recently found one is produced, starting from the highest pragmatic level.

The overall process starts by establishing a top goal to built down the full structure in the sentence level. The discourse generation carried out by the model is a task driven by several linguistic elements from high-level goals to sentences. Actual elements include linguistic functions, conversation turns, speech acts, semantic propositions and syntactic structures. The linguistic functions are the as discussed in the next sections.

3.3 Linguistic Elements for Interactive Dialogues

Discourse generation starts from a set of linguistic functions associated with the different components of the dialogue structure including linguistic functions for initiating the dialogue (*FID*), answering a question (*FRC*), determining whether the discourse goal has been met (*FPM*), requiring a new dialogue topic (*RNT*), and finishing the dialogue (*FD*). Each function is defined by a set of conversation turns as follows:

- Initiating Dialogue (*FID*) includes two kind of (conversation) turns: specific and general questions.
- Answering Question (*FRC*) considers five types of turns depending on the information obtained from the search agent:
  - Answering by prompting the detection of different languages in documents.
  - Answering by prompting the number of references to documents.
  - Answering by prompting the most frequent search's features (i.e., type of webpage, language, etc.).
  - Answering by prompting the huge amount of information available for the requested topic.
  - Answering by prompting a very general question.
- Determining whether the discourse goal has been met (*FPM*) considers a turn for "*Confirm the usefulness of the provided information*".
- Requiring a dialogue's new topic (*RNT*) is composed of two turns: requiring a topic about the same theme or requiring a topic about a new theme.
- Finishing the dialogue (*FD*) considers three kind of turns: gratitude, usefulness of the search and apologies for the results.

Each conversation turn involves specific speech acts from which the NLG generation task starts, including:

- Speech Act for initiating the dialogue:

  $$SPEECH\_ACT \rightarrow ASK\_ACTION.$$

- Speech Act for a descriptive sentence and a request:

  $$SPEECH\_ACT \rightarrow DESCRIPTIVE\_ACT \quad REQUEST\_ACT$$

  (i.e., "*Twenty retrieved documents are written in different languages, do you prefer those in English*?").

- Speech Act for an explanatory sentence and request:

  $$SPEECH\_ACT \rightarrow EXPLANATORY\_ACT \quad REQUEST\_ACT.$$

- Speech Act for finishing the dialogue:

$$SPEECH\_ACT \rightarrow CONCLUSION\_ACT.$$

- Etc.

Each speech act is associated with a specific propositional structure to be generated. For example:

$$DESCRIPTIVE\_ACT \rightarrow DESCRIP\_PROP \quad REQUEST\_PROP,$$
$$EXPLANATORY\_ACT \rightarrow EXPLAN\_PROP \quad REQUEST\_PROP.$$

Furthermore, each propositional structure is built up from semantic functions (also known as cases), including:

- *AGENT*: defines a function which features are attributed to. Example: "`Your question is too general`".
- *ATTRIBUTE*: involves the features assigned to an object or agent. Example: "`.. documents are written in English..`".
- *EVENT*: defines either an action or a significant process. Example: "`What theme do you want to search forotherref`?".
- *OBJECT*: adds information to the event's features. Example: "`Can you question be more specific so that I can assist you`?".
- *GENITIVE_OBJECT*: involves objects defined by their extension. Example: "`I found information on research groups...`".
- *INFINITIVE_OBJECT*: adds information to a verb action. Example: "`Do you need to find out about any other topic`?".
- *MEAN_INSTRUMENT*: defines a means by which an event is established. Example: "`the retrieved information is written in different languages..`"
- *USEFULNESS*: expresses the usefulness of the performed task. Example: "`I hope I was useful in your search requests..`".
- Etc.

Semantic propositions are then defined by using semantic rules as follows:

```
ASK_PROPOSITION
  -> INTERROGATIVE-OBJECT TOPIC EVENT INFINITIVE-OBJECT
   | INTERROGATIVE-OBJECT    EVENT

DESCRIPTIVE_PROPOSITION -> AGENT EVENT MEAN-INSTRUMENT |
                           EVENT GENITIVE-OBJECT
```

```
DESCRIPTIVE_REQUEST_PROPOSITION
  -> OBJECT EVENT MEAN-INSTRUMENT

INTERROGATIVE_REQUEST_PROPOSITION
  -> OBJECT EVENT INFINITIVE-OBJECT |
     INTERROGATIVE-OBJECT  DATIVE EVENT |
     EVENT QUANTITY OBJECT GOAL
EXPLANATORY-PROPOSITION -> EVENT  GENITIVE-OBJECT
...
```

Next, the corresponding syntactical structures must be generated. These consider usual types of groups such as Noun Phrases (*NP*), Prepositional Phrases (*PP*), Adjective Phrases (*ADJP*). Note that each semantic function can enable one or more different syntactical structures. A typical set of grammar rules (according to their semantic cases) for the generation task include:

```
AGENT  -> NP
ATTRIBUTE -> ADJP
EVENT -> V | Vcop | VIMP  | VP
OBJECT -> NP
GENITIVE-OBJECT -> NP
INTERROGATIVE-OBJECT  -> NP
INFINITIVE-OBJECT -> NP
INTERROGATIVE-THEME -> PP
MEAN-INSTRUMENT -> PP
...
```

Each grammar phrase (i.e., syntactical structure) is composed of different grammar categories such as determiners (*det*), articles (*artdef*), determining adjectives (*adjdet*), nouns (*n*), adjectives (*adj*), prepositions (*prep*), verbs (*v*, *Vcop*, *Vimp*), pronouns (*proper*, *pro*), etc. Accordingly, examples of defined grammar rules look as follows:

```
NP ->      DET n    |
           adj  n   |
           n    |
           det n PP |
           proper |
           prop |
           n PP NP CONJ NP
```

```
DET ->   ARTDEF |
             ADJDET |
         ADJINT

ADJP -> INT adj

PP -> prep NP

ADV -> QUANTITY
...
```

Finally, the generation lexicon is defined to include words and features extracted from the experiments' transcriptions. Each lexical entry is specified by its grammar category and the corresponding morphological features. Discourse generation matches the lexical information for each rule and the words in the dictionary so as to select those corresponding to the specified constraint. Different morphological features were considered and included genre (male, female), number (singular, plural), and type (i.e., groupings or semantic fields which specify syntactical-semantics behaviors. For instance, a type-B noun indicates a noun that designates information. Lexical items used for the generation included entries as follows:

```
ARTDEF    :   articles

a (artdef,fem,sing)

ADJDET    : quantifiers

Type-A1 ADJDET: Quantity that meets requirement

much (adjdet,typeA1,fem,sing)
huge (adjdet,typeA1,fem,sing)

Type-A2 ADJDET  : Quantity that meets requirement
...
sufficient (adjdet,typeA2,inva,sing)
...
Type-A NOUN: nouns on information

theme (typeA,np,masc,sing)
information (typeA,np,quantifiable,fem,sing)
```

```
topic (typeA,np, masc, sing)
reference to document (typeA,hiper, fem, sing)
topic (typeA,np, masc,sing)
...

Type-B Noun : on language

language (n,typeB,hyper,fem,sing)
languages (n,typeB,hyper,fem,sing)
spanish (n,typeB,native,male,sing)
english (n,typeB,extra,male,sing)
..

Type-A ADJ : denotes a different feature

different (adj,typeA,inva,plu)
diverse (adj,typeA,fem,plu)
distinct (adj,typeA,fem,pl)
..

VERBS

Type-A Verb : on wishes

prefer (v,typeA,2ps,pre)
want (v,typeA,2ps,pre)
wish (v,typeA,2ps,pre)
..
```

Once all the previous elements are put together, the NLG task can be performed from high-level goals to instantiated utterances. Thus, the overall process of NL sentences generation is guided by the selected turns as follows:

```
CONVERSATION_TURN_1  -> SPEECH_ACT_1 ...
CONVERSATION_TURN_2  -> SPEECH_ACT_2
...
SPEECH_ACT_1 -> INTERROGATION_ACT_1
...
INTERROGATION_ACT_1 -> PROPOSITION_QUESTION_9
...
```

```
PROPOSITION_QUESTION_9 -> REQUEST_FOR_OBJECT + EVENT
...
DESCRIPTIVE_PROPOSITION -> AGENT + EVENT + MEANS
...
AGENT  -> NP
(surface generation starts at this point)
...
MEANS  -> NP | NP OBJECT ...
EVENT  -> VERB  VP
(further linguistic features and constraints, lexicon,
etc).
```

3.4 Adaptive Search Agent

Unlike traditional search engines or IR systems, we have designed a search agent which does not deliver all the obtained information from web search results to the user. Instead, the agent waits until sufficient knowledge about the user's feedback, goals, etc. is acquired. As the interaction goes on, the agent refines the requests and filters the initial information obtained from the user's feedback and the search process, until a proper amount of information can be displayed.

The search agent is composed of three components: the search engine itself, the criteria analyzer which processes the obtained information based on the user's feedback and current context knowledge, and the information status record which manages the obtained information and the knowledge acquired from previous dialogues with the user. The results produced by the agent are later used by the NL generator to produce adaptive utterances which match the current conversation status and constraints.

By "criteria," it is meant the kind of underlying representation stated for the documents and the user's profile which is similar to that used in IR but it has been augmented with vector-based specific purpose features to support the needed expressiveness. Both documents and user's queries are represented in a multidimensional space so that once a question is processed it is then translated into a pattern representing a criteria vector. By using distance metrics and some combination of existing search engines, the appropriate documents are retrieved. A document D so structured is represented as a vector i: $V_d = X_0 X_1 \ldots X_n$ where X_i states the value extracted from the users or the search results for the ith criterion of the documents being obtained.

These criteria represent relevant context information related to web pages which can be useful in training the patterns and filtering the search results. Initially, criterion

X_0 will concern the main theme and the rest of the vector will remain empty. As the dialogue proceeds and new search results are obtained, these slots become filled.

Using the information provided by the criteria above, the dialogue samples and the current context information, it was possible to extract and synthesize the most frequent search patterns, some of which involve *URL Address of the Web page being selected, Documents' author, Language in which the document is written in, Document's source country, Type of document/page* (*commercial, education, etc.*), *documents related to events, Technical documentation, Research groups, Products and Services*, etc.

For example, slot X_4 can store the criterion "language." Thus, if a further interaction learns that the user is interested in getting documents in *Spanish* language, the slot X_4 will be filled with the value *Spanish*, so to take part of the next search attempts. Each criterion has also a weight which represents its "contribution" to a retrieved document.

Whether the criteria are filled with information from the dialogue or from the current intermediate search, the previously trained agent takes the matching vectors and performs the search request onto the Web. If no further filtering can be done, the results are showed, otherwise, new requests from the dialog context are issued to the agent.

When information in the vectors and the user feedback provided is not enough or unavailable, the agent is still capable of making simple decisions by predicting the most likely actions to perform. That is, given some context information, the agent tries to determine the bets action to perform. For this, the agent is provided with a simple trainable Bayesian inference strategy. Its outcome has two basic consequences: one affecting the information filtered and other assisting the sentence generation to look for criteria missed or incomplete. The general strategy to decide the next likely action based on the confidence levels is shown in Fig. 7.

In practice, these actions are translated into high-level goals of pragmatic constraints which cause a particular kind of NL dialogue to be generated (i.e., question, request, feedback, etc.).

4. Analysis and Results

A search model which uses intelligent agent technology and NLP techniques was designed and implemented in order to investigate the extent to which information overloading and intelligent search capabilities can be effective for web search and filtering.

The results produced by a prototype system are defined in terms of two previously discussed issues: one regarding the kind of NL utterance automatically generated by

```
ALGORITHM:  Generate Action for Given Criteria
Input : DocumentVector, Situation
Output: Action

BEGIN
   CF <--  Compute Probability "P(Action/Situation)"
   IF (CF >= 0.5) y (CF<=0.96) THEN
        Agent makes suggestion with confidence CF
        IF (Positive_Feedback_from_User) THEN
           Update frequencies  of pairs (Action,Situation)
           Add new situation to the list of Situations
           Search and Filter documents according to Situation
           (new criterion's value found for DocumentVector)
        END-IF
   ELSE
     IF  (CF >0.96) THEN
        Agent makes question according to context information
           and Action given by higher pragmatic goal
        Update frequencies of pairs (Action,Situation)
     ELSE
        IF  (CF=0) THEN
           Action <-- Agent goes on with Dialog
        ELSE
           Action <-- Agent's request for further information
                      (High-level pragmatic goal invoked)
END
```

FIG. 7. The strategy for action selection.

the system which suggests that the search-driven discourse generation may indeed be plausible. A second issue concerns the benefits of using this kind of interaction to decrease information overloading and so the time spent by users searching on the web.

For the dialogue component, a discourse generator was implemented using the **SnePS** framework [43] which provides NL facilities and knowledge representation tools (i.e., semantic nets, etc.), and medium-size NL interfaces.

The analysis of results was based on the automatic generation of 1000 dialogue structure samples obtained from the discourse processing task performed by the system (original dialogues were allowed in Spanish so some English translations are provided). The discourse processor was able to produce dialogue structures that look like this:

```
Input sentence to nlip: ("hola" ".")
Resulting parse: ("sobre" "que" "topico" "deseas"
                 "revisar" "?")
(What topic do you want to search for)

Input sentence to nlip: ("sobre" "linguistica"".")
```

```
(on linguistics)
Resulting parse: ("tu" "consulta" "es" "muy" "amplia"
                 "puedes" "delimitar" "la" "mas"
                 "por favor" "?")
(your question is too broad, could you please be more
specific?)

Input sentence to nlip: ("bueno")
(fine)
Resulting parse:    ("averiguaste" "lo que"
                    "necesitabas" "?")
(Did you find what you were looking for?)

...
Input sentence to nlip ("no" ".")
Resulting parse:    ("necesitas" "otra" "informacion"
                    "sobre" "este" "tema" "?")
(Do you need further information about this topic?)

...
Input sentence to nlip: ("bueno")
(yes)
Resulting parse:    ("hallaste" "lo que"
                    "necesitabas" "?")
(Did you find what you were looking for?)

Input sentence to nlip ("no" "realmente" ".")
(not really)
Resulting parse:    ("lo siento" "para" "otra" "vez"
                    "sera" "hasta" "pronto!!")
(I'm sorry, see you next time)
```

Conversation turns selected for the NLG task use the linguistic functions previously defined for each component of the dialogue structure as follows:

1. *Delimiting the question* concerns the dialogue initiation linguistic function (*FID*). Generation of utterance-question is performed on the theme the user requires to search for:

   ```
   Input sentence to nlip: ("hola" ".")
   Resulting parse: ("sobre" "que" "topico"
   ```

```
                    "deseas" "revisar" "?")
Input sentence to nlip: ("sobre" "linguistica" ".")
```

2. *Specifying the results of the web search* concerns generating the user's utterance as answer to the system's request so as to further specify the topic:

```
Resulting parse: ("tu" "consulta" "es" "muy"
                  "amplia" "puedes" "delimitar" "la"
                  "mas" "por favor" "?")
Input sentence to nlip: ("bueno" ".")
```

3. *Specifying the achievement of the discourse goal* in which the user's utterance is produced as an answer to the system's question regarding the accomplishment of the searched topic:

```
Resulting parse: ("averiguaste" "lo que"
                  "necesitabas" "?")
Input sentence to nlip: ("no" ".")
```

4. User's answer is generated from a system's generated question about the request for additional information on the topic:

```
Resulting parse: ("necesitas" "otra" "informacion"
                   "sobre" "este" "tema" "?")
Input sentence to nlip: ("bueno" ".")
```

5. *Finishing the dialogue* is carried out from a question on the discourse goal's accomplishment:

```
Resulting parse: ("hallaste" "lo que"
                  "necesitabas" "?")
Input sentence to nlip: ("no" "realmente" ".")

Resulting parse:("lo siento" "para" "otra" "vez"
                  "sera" "hasta pronto!!")
```

In order to generate different explanatory and descriptive discourse utterances aiming to refine the results of the topic search, a prototype search agent was designed for simulation and testing purposes. Generating this kind of discourse is then carried out using information form the user's feedback and the search's previous results. Triggering the different implemented rules and associated actions were simulated from five types of user's discourse inputs as follows:

- *Case* 1:

  Generating explanatory answers for which the system requires the user to provide further specification on the requested theme. This involved dealing

with the user's answer "*on linguistics*". Next, the search agent finds too much information about the topic which may be due to a too broad question (i.e., that containing a one-word topic). This behavior is exhibited in some of the processed samples as follows:

Sample No. 26:

```
Input sentence to nlip: ("hola" ".")
Resulting parse: ("sobre" "que" "asunto" "quieres"
                 "consultar" "?")
(What theme do you want to find?)

Input sentence to nlip: ("sobre" "linguistica" ".")
Resulting parse: ("tu" "consulta" "es" "muy"
                 "amplia" "debes" "delimitar" "la"
                 "mas""por favor")
(your question is too broad, could you please be
more specific?)

Input sentence to nlip: ("bueno" ".")
Resulting parse: ("hallaste" "lo que" "necesitabas"
                  "?")

Input sentence to nlip: ("no" ".")
Resulting parse: ("necesitas" "otra" "informacion"
                 "en relacion con" "este" "topico"
                 "?")

Input sentence to nlip: ("bueno" ".")
Resulting parse: ("encontraste" "lo que" "buscabas"
                 "?")
Input sentence to nlip: ("si" "gracias" ".")
Resulting parse: ("gracias por consultarme!!"
                 "nos vemos!!")
```

- *Case* 2:

  Generating explanatory utterances providing that too much information is found, requires delimiting the user's general question. For example, by processing answers such as "*on Latin*", "*on English*" in the following sample dialogues:

  Sample No. 8: (*question on Latin*)

```
Input sentence to nlip: ("hola" ".")
```

```
Resulting parse: ("en relacion con" "que" "tema"
                 "deseas" "revisar" "?")

Input sentence to nlip: ("sobre" "latin" ".")
Resulting parse: ("halle" "bastante" "informacion"
                 "acerca de" "este" "asunto" "que"
                 "rasgo" "mas" "preciso" "prefieres"
                 "revisar" "?")
(I found too much information about it,
what specific issue do you want to check?)

Input sentence to nlip: ("bueno" ".")
Resulting parse: ("encontraste" "lo que" "buscabas"
                 "?")

Input sentence to nlip: ("si" ".")
Resulting parse: ("necesitas" "revisar" "acerca de"
                  "otro" "asunto" "?")

Input sentence to nlip: ("bueno" ".")
Resulting parse: ("encontraste" "lo que"
                 "necesitabas" "?")

Input sentence to nlip: ("si" "gracias" ".")
Resulting parse: ("gracias por consultarme!!"
                 "nos vemos!!")
```

Sample No. 11: (*question on English*)

```
Input sentence to nlip: ("hola" ".")
Resulting parse: ("acerca de" "que" "topico"
                  "quieres" "consultar" "?")

Input sentence to nlip: ("sobre" "ingles" ".")
Resulting parse: ("hay" "mucha" "informacion"
                 "en relacion con" "este" "tema"
                 "puedes" "acotar""mas" "tu"
                 "consulta" "para" "ayudar" "te"
                 "bien"  "?")

Input sentence to nlip: ("bueno" ".")
```

```
Resulting parse: ("hallaste" "lo que"
                 "necesitabas" "?")

Input sentence to nlip: ("no" ".")
Resulting parse: ("necesitas" "otra" "informacion"
                 "acerca de" "este" "topico" "?")

Input sentence to nlip: ("bueno" ".")
Resulting parse: ("averiguaste" "lo que"
                 "necesitabas")

Input sentence to nlip: ("si" "gracias" ".")
Resulting parse: ("espero" "haber sido" "util"
                 "chao!!")
```

- *Case* 3:

  Generating descriptive discourse to express the different languages of the retrieved documents. This allows the discourse generator to produce suggestions on the desired features of the retrieved documents:

  Sample No. 9:

```
Input sentence to nlip: ("hola" ".")
Resulting parse: ("sobre" "que" "topico" "quieres"
                 "consultar" "?")
(what topic do you want to search for?)

Input sentence to nlip: ("sobre" "aleman" ".")
Resulting parse: ("la" "informacion" "esta" "en"
                 "distintas" "lenguas" "la"
                 "prefieres" "en"  "espanol" "?")
(I found information in different language, do you
prefer them in Spanish?)
...
Input sentence to nlip: ("bueno" ".")
Resulting parse: ("averiguaste" "lo que"
                 "necesitabas")

Input sentence to nlip: ("no" ".")
Resulting parse: ("necesitas" "otra" "informacion"
                 "en relacion con" "este"
                 "topico" "?")
```

```
Input sentence to nlip: ("bueno" ".")
Resulting parse: ("hallaste" "lo que" "buscabas" "?")

Input sentence to nlip: ("si" "gracias" ".")
Resulting parse: ("espero" "haber sido" "util"
                  "hasta la proxima!!")
```

- *Case* 4:
  Simulating the user's answer for "*on Spanish*" requires the system to generate a descriptive sentence involving an exact number of references to documents being found (i.e., expressing that less than 50 references were found) by the search agent:

```
Input sentence to nlip: ("hola" ".")
Resulting parse: ("acerca de" "que" "tema" "deseas"
                  "revisar" "?")

Input sentence to nlip: ("sobre" "castellano" ".")
Resulting parse: ("hay" "veinte" "referencias"
                  "sobre" "este" "asunto" "las"
                  "quieres" "revisar" "todas" "?")

Input sentence to nlip: ("bueno" ".")
Resulting parse: ("averiguaste" "lo que"
                  "necesitabas","?")

Input sentence to nlip: ("si" ".")
Resulting parse: ("necesitas" "otra" informacion"
                  "en relacion con" "este"
                  "asunto" "?")

Input sentence to nlip: ("bueno" ".")
Resulting parse: ("averiguaste" "lo que" "buscabas"
                  "?")

Input sentence to nlip: ("no" "realmente" ".")
Resulting parse: ("lo siento" "para" "otra" "vez"
                  "sera" "chao!!")
```

- *Case* 5:
  Generating descriptive discourse to express the three most frequent features found by the agent in the documents. This requires for the user to further specify

an specific criterion (i.e., "*on French*"). By producing this utterance, the user is informed the condition that "*more than* 50 *references and less than* 100 *were found*":

Sample no. 3:

```
Input sentence to nlip: ("hola" ".")
Resulting parse: ("acerca de" "que" "tema"
                 "prefieres" "consultar" "?")

Input sentence to nlip: ("sobre" "frances" ".")
Resulting parse : ("halle" "informacion" "sobre"
                   "grupos de investigacion"
                   "cursos y departamentos" "que"
                   "aspecto" "te interesa" "?")

Input sentence to nlip: ("bueno" ".")
...
Resulting parse: ("encontraste" "lo que" "buscabas"
                 "?")

Input sentence to nlip: ("no" ".")
Resulting parse: ("necesitas" "otra" "informacion"
                 "en relacion con" "este"
                 "topico" "?")

Input sentence to nlip: ("bueno" ".")
Resulting parse: ("averiguaste" "lo que"
                 "necesitabas" "?")

Input sentence to nlip: ("no" "realmente" ".")
Resulting parse: ("lo siento" "para otra" "vez"
                 "sera" "hasta la proxima!!")
```

As for the filtering issue, the model performance was analyzed regarding the experiments evaluating the number of conversational turns in the dialogue necessary to get a more precise requirement and filtered information against the number of references/documents which matched these requirements. Initially, the set of possible candidate documents became more than 30 000 documents references but for simplicity's sake, the scope has been reduced to no more than 100 references.

Three experiments were carried out and can be seen in Figs. 8, 9 and 10. These involved dialogue interaction on the focus topic **Java**, **Computer Networks** and

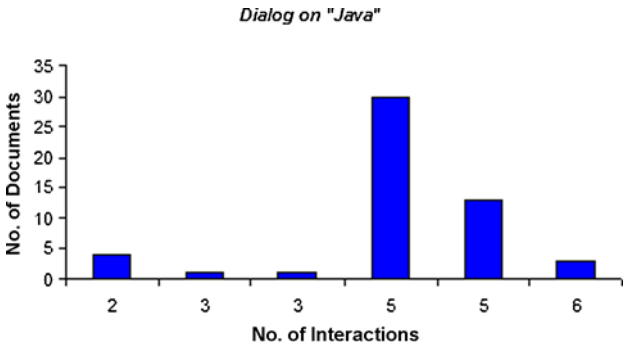


FIG. 8. Interactive dialogue-based search for `Java`.

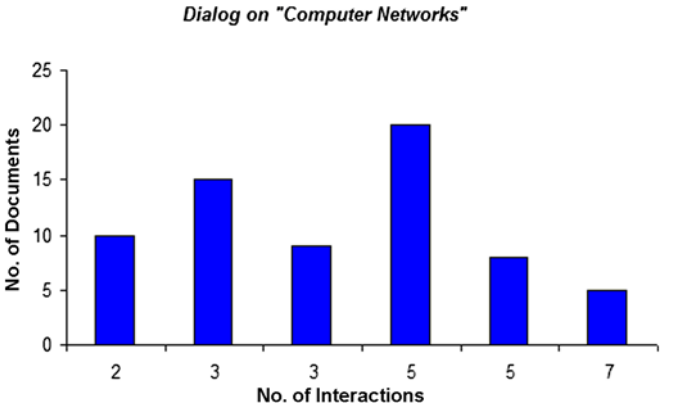


FIG. 9. Interactive dialogue-based search for `Computer Networks`.

Cartoons. To better understand the analysis, each interaction is defined by one or more dialogue interactions (also known as exchanges or turns) between a web user and the dialogue-based search system.

Dialogue interactions for experiment on `Java` (Fig. 8) showed an increase in the number of documents matched as more than three turns are exchanged. It does not come up by a chance: for the same number of interactions (five), different results are showed mainly due to the adaptive way how the dialog goes on. This is, the context and kind questions made by the agent are changing depending on the situation and the document's contents. Different results were obtained for the same number of interactions because the type of document searched for was changed as other features were restricted on the dialogue. A similar situation occurs as the dialog specifies a

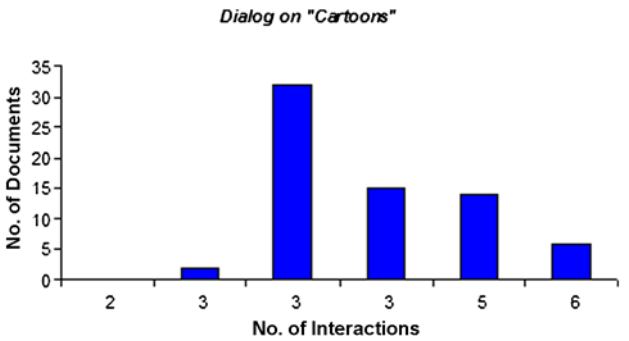


FIG. 10. Interactive dialogue-based search for `Cartoons`.

TABLE I
OVERALL RESULTS FOR FILTERING

Theme	Average		
	Interactions	Documents	Reduction (%)
`Java`	3.6	9.8	72.0
`Cartoons`	3.7	11.5	67.2
`Computer Networks`	3.5	7.5	78.5
Average	3.6	9.6	72.6

constraint on the language, in which case, most of the original document references were not matched.

For the interactions on `Computer Networks`, Fig. 9 shows a higher number of retrieved relevant documents than for the experiments on `Java`. Even so, the filtering at the end of this dialogue is better than for the previous experiments: the maximum number of obtained documents is less than for interactions on `Java`, and the drop on the filtered documents is better for 3-interaction dialogues.

In the experiments on `Cartoons`, a similar behavior to interactions on `Java` is observed. Even in dialogues with three exchanges, sudden increments were observed, going up from 1 to nearly 35 resulting references. This change is due to an inference drawn by the agent and a user's restriction related to the document type.

From all the experiments, it can be seen that there is an important drop in the results obtained with a minimum of conversation turns due to constraints on the nature of the information finally delivered. Overall, the average interactions and filtering results required to meet the goal for each experiment are shown in Table I. A filtering of almost 73% is achieved with an average number of 3.6 dialogue interactions, this is, in average, less that 27% of the initially-retrieved documents are actually showed

to the user. It is also important to highlight that all the experiments required no more than four conversation turns to filter at least 67.2% of the documents.

5. Conclusions

In this work, a new approach to collaborative web-driven search strategy is proposed in order to deal with the problem of information overloading and filtering. The model can recognize and exploit context and accommodate user feedback through a minimum number of dialogue interactions.

Experiments, conducted in the context of a filtering system for Web documents, highlight the promise of combining NLP methods and intelligent agents technology for an information access problem.

Experiments suggest that a lot of time may be saved if we are provided with the weighted features usually presented on the retrieved information depending on its importance degree or usage. In any case, interactions (form and content) will strongly rely on these factors. However, it should not leave user's contributions apart from the decisions being made by the system.

Despite the moderated complexity of the experiments and the design constraints, the issues which have been identified should not drastically change through more advanced requirements and implementations (i.e., different languages, different search capabilities, etc.).

From a language-centered viewpoint, the current model based on dialogue interactions shows promise to be a novel and interesting work strategy to deal with more specific information searching requirements. Here both designing and implementing a NLG system can easily be adapted to tailored communicating situations. Even although there is a lot of NLG systems, as far as we know, this is the first attempt to integrate these technologies to address the problem of searching and filtering on the Web effectively.

Compared to other approaches using NLP to tackle similar problems, in the proposed model the web user's interests and goals are obtained as the dialogue proceeds whereas in other approaches this kind of information (i.e., interestingness and relevance) is established in advance by providing different parameter values. On the other hand, the underlying working assumption in some search models using NLP is that a user's profile must be analyzed in order so to prune part of some "story" and so to reach a final answer. Since our "stories" are real documents extracted from the Web, we can not afford understanding the complete set of texts. Instead, the model deals with the dialogue as a way to extract important knowledge from the user and then properly filtering the obtained documents.

ACKNOWLEDGEMENTS

This research is sponsored by the National Council for Scientific and Technological Research (FONDECYT, Chile) under grant number 1070714 "An Interactive Natural-Language Dialogue Model for Intelligent Filtering based on Patterns Discovered from Text Documents."

REFERENCES

[1] Ardissono L., Boella G., Lesmo L., "Plan based agent architecture for interpreting natural language dialogue", *Internat. J. Human–Computer Stud.* **52** (2000) 583–636.
[2] Austin J., *How to do Things with Words*, Oxford University Press, A Galaxy Book, New York, 1970.
[3] Benamara F., Dizier P., "Dynamic generation of cooperative natural language responses in webcoop", in: *Ninth European Workshop on Natural Language Generation, EACL, Budapest, Hungry*, 2003.
[4] Berry M., Browne M., *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, SIAM, 1999.
[5] Bloedorn E., Mani I., "Using NLP for machine learning of user profiles", *Intell. Data Anal.* **2** (1–4) (1998) 3–18.
[6] Burstein M., Ferguson G., Allen J., "Integrating agent-based mixed-initiative control with an existing multi-agent planning system", Technical Report 729, Computer Science Department, University of Rochester, May, 2000.
[7] Callan J., *Artificial Intelligence*, Palgrave–MacMillan, 2003.
[8] Cohen P., Levesque H., "Performatives in a rationally based speech act theory", Technical Note 486, SRI International, 1990.
[9] Cohen P., McGee D., Clow J., "The efficiency of multimodal interaction for a map-based task", in: *Proc. of the 6th Conference on Applied Natural Language Processing, Seattle, USA*, 2000.
[10] Hermjakob E., Hovy U., Gerber L., Junk M., "Question answering in webclopedia", in: *Proc. of the TREC-9 Conference, NIST, Gaithersburg, MD*, 2000.
[11] Di Eugenio B., Fossati D., Yu D., Haller S., "Aggregation improves learning: experiments in natural language generation for intelligent tutoring systems", in: *ACL05, 43rd Meeting of the Association for Computational Linguistics, Ann Arbor, MI, USA*, June 2005.
[12] Di Eugenio B., Glass M., Haller S., "Simple natural language generation and intelligent tutoring systems", in: *The AIED 2001 Workshop on Tutorial Dialog Systems, San Antonio, TX*, May 2002.
[13] Ferreira A., "Generating descriptive and explanatory discourse based on a computational linguistics model", PhD thesis, Catholic University of Valparaiso, Chile, 1998.
[14] Ferreira A., "Feedback strategies for second language teaching with implications for intelligent tutorial systems", PhD thesis, Division of Informatics, University of Edinburgh, Edinburgh, Scotland, 2003.

[15] Ferreira A., Atkinson J., "Intelligent search agents using web-driven natural-language explanatory dialogs", *IEEE Computer* **38 (10)** (2005) 44–52.
[16] Grice H., "Logic and conversation", in: Cole R., Morgan (Eds.), *Syntax and Semantic*, Academic Press, 1990.
[17] Grosz B., Hunsberger L., Kraus S., "Planning and acting together", *AI Magazine* **20** (4) (1999) 23–34.
[18] Haller S., Di Eugenio B., "Minimal text structuring to improve the generation of feedback in intelligent tutoring systems", in: *Proc. of the Sixteenth Florida Artificial Intelligence Research Conference, St. Augustine, FL*, May 2003.
[19] Haller S., Di Eugenio B., Trolio M., "Generating natural language aggregations using a propositional representation of sets", in: *Proc. of the Fifteenth Florida Artificial Intelligence Research Conference, Pensacola, FL*, May 2002, pp. 365–369.
[20] Haller S., "An introduction to interactive discourse processing from the perspective of plan recognition and text planning", *Artif. Intell. Rev.* **13** (4) (1999) 259–311.
[21] Harabagiu S., Pasca M., Maioramo S., "Experiments with open-domain textual question-answering", in: *Proc. of COLING 2000*, 2000, pp. 292–298.
[22] Hofmann T., "Latent semantic models for collaborative filtering", *ACM Trans. Inform. Systems* **22** (1) (2004) 89–115.
[23] Holscher C., Strube G., "Web search behavior of internet experts and newbies", in: *9th International World Wide Web Conference, Amsterdam*, May 2000.
[24] Jansen B., Spink A., "Real life, real users, and real needs: A study and analysis of user queries on the web", *Inform. Process. Manag.* **36** (2) (2000) 207–227.
[25] Jurafsky D., Martin J., *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, 2000.
[26] Kraus S., *Strategic Negotiation in Multiagent Environments*, MIT Press, 2001.
[27] Landauer T., Foltz P., Laham D., "An introduction to latent semantic analysis", *Discourse Processes* **10** (25) (1998) 259–284.
[28] Lau T., Horvitz E., "Patterns of search: Analyzing and modeling web query refinement", in: *Proc. of the Seventh International Conference on User Modeling*, ACM Press, 1999.
[29] Levy A., Weld D., "Intelligent internet systems", *Artificial Intelligence* **11** (8) (2000) 1–14.
[30] Lochbaum K., Grosz B., Sidner C., "Models of plans to support communication: An initial report", in: *Proc. of AAAI-90, Boston*, 1990.
[31] Maes P., "Evolving agents for personalized information filtering", in: *Proc. of the Ninth Conference on Artificial Intelligence for Applications '93, Orlando, FL, USA*, March 1993.
[32] Maes P., "Agents that reduce work and information overload", *Commun. ACM* (1994) 31–40.
[33] Mann W., Thompson S., "Rhetorical structure theory: A theory of text organisation", in: L. Polanyi (Ed.), *The Structure of Discourse*, Ablex, Norwood, NJ, June 1987.
[34] Marcu D., "The rhetorical parsing, summarization, and generation of natural language texts", PhD thesis, Department of Computer Science, University of Toronto, June 1997.
[35] Marcu D., "A formal and computational synthesis of Grosz–Sidner and Mann–Thompson theories", in: *Workshop on Levels of Representation in Discourse, Edinburgh, Scotland*, June 1999.

[36] Moore J., *Participating in Explanatory Dialogues: Interpreting and Responding to Questions in Context*, MIT Press, Cambridge, MA, 1995.
[37] Moore J., Foster M., Lemon O., White M., "Generating tailored comparative descriptions in spoken dialogue", in: *FLAIRS Conference, Florida, USA*, 2004.
[38] Ram A., "Interest-based information filtering and extraction in natural language understanding systems", in: *Bellcore Workshop on High-performance Information Filtering*, November 1991.
[39] Ram A., "Natural language understanding for information filtering systems", *Commun. ACM* **35** (12) (1992) 80–82.
[40] Reiter E., Dale R., *Building Natural Language Generation Systems*, Cambridge University Press, 2000.
[41] Salter J., Antonopoulos N., "Cinemascreen recommender agent: Combining collaborative and content-based filtering", *IEEE Intelligent Systems Appl.* **21** (1) (2006) 35–41.
[42] Schiffrin D., *Discourse Analysis*, Cambridge University Press, England, 1987.
[43] Shapiro S., "Sneps 2.3. user's manual", Technical report, Department of Computer Science, SUNY at Buffalo, NY, USA, 1995.
[44] Shapiro S., "Conditional snere policies, snerg technical note 39", Technical report, Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, December 2005.
[45] Stent A., "Rhetorical structure in dialog", in: *Proc. of the 2nd International Natural Language Generation Conference (INLG'2000)*, June 2000.
[46] Stone M., Webber B., "Textual economy through close coupling of syntax and semantics", in: *Proc. 1998 Internat. Workshop on Natural Language Generation, Niagara-on-the-Lake, Canada*, August 1998.
[47] Tong L., Changjie T., Jie Z., "Web document filtering technique based on natural language understanding", *Internat. J. Comput. Processing of Oriental Languages* **14** (3) (2001) 279–291.
[48] Ungar L., Foster D., "A formal statistical approach to collaborative filtering", in: *Conference on Automated Learning and Discovery (CONALD)*, 1998.
[49] Van Dijk T., "Principles of critical discourse analysis", in: Wetherell E. (Ed.), *Discourse Theory and Practice*, Sage, London, 2001.
[50] Voorhees E., "Natural language processing and information retrieval", *SCIE* (2001) 32–48.
[51] Wooldridge M., *Reasoning about Rational Agents*, MIT Press, 2000.
[52] Wright H., Poesio M., Isard S., "Using high level dialogue information for dialogue act recognition using prosodic features", in: *Proc. of the ESCA Workshop on Prosody and Dialogue, Eindhoven*, 1999.
[53] Zukerman I., Albrecht D., Nicholson A., "Predicting users' requests on the www", in: *Proc. of the Seventh International Conference on User Modeling*, 1999.

This page intentionally left blank

A Tour of Language Customization Concepts

COLIN ATKINSON

University of Mannheim
68161 Mannheim
Germany

THOMAS KÜHNE

Darmstadt University of Technology
64289 Darmstadt
Germany

Abstract

Although the UML is often perceived as a *universal* modeling language it was not designed to fulfill this goal. Rather, the primary goal of the designers was to provide a *unified* modeling language. Because the applicability of the core language is limited, the UML has always offered mechanisms to enable it to be adapted to the needs of different users and applications. Over the years, a wide variety of different language customization mechanisms have been proposed, ranging from stereotypes, profiles, and metamodeling to domain-specific languages. However, the relationships between these different approaches and the different capabilities and options that they provide to users have never been clearly elaborated. In this chapter we provide a tour of the most important language customization mechanisms and by means of a unified case study we compare and contrast their pros and cons. We also review the current "state of the art" and present our view of how the "domain-customized language" approach to software engineering can best be supported in the future.

ADVANCES IN COMPUTERS, VOL. 70
ISSN: 0065-2458/DOI: 10.1016/S0065-2458(06)70003-1

1. Introduction

Software engineering involves the creation of many artifacts which may be described using a variety of languages. However, choosing the optimal language for each artifact is not easy. One of the most important issues is the tension between the goal of widespread communication and the goal of using the most appropriate language for each specific task. With respect to modeling, a standard like the UML (Unified Modeling Language) [16,17] which has unified and replaced a large number of similar object-oriented modeling notations, is an ideal facilitator of world-wide

communication. It has, however, been criticized as promoting a "one size fits all" approach that provides suboptimal abstractions for the majority of applications. This is not surprising since the "U" in "UML" was never meant to mean "universal", i.e., the UML is useful in a broad range of applications, but certainly not all. Domain-Specific Languages (DSLs), on the other hand, provide the best possible fit for a particular task but create communication chasms since their concepts and notations are often only known to a small team of language designers and users. In the worst case, domain-specific languages can be conceptually very similar, but vary considerably in terms of their notation, thus creating an unnecessary Tower of Babel and undoing the consolidation achievement of the UML.

Fortunately, there is some useful middle-ground between a fixed language standard and an uncompromisingly adapted, one-off language design which we refer to as a *domain-customized language* (DCL). A domain-customized language is a *derived language* whose definition is based on some existing base language. It customizes that language for some specific purpose using such operations as reduction, modification, and extension. Domain-customized languages are therefore a subclass of domain-specific languages whose members have carefully defined relationships to their respective base languages. Typically, the base language will be a widely-known standard and the amount of customization will be minimized in the sense that as much as possible of the base language will be left unaltered. The use of a DCL thus allows maximum communication while still being tailored to a specific task. The OMG's (Object Management Group) approach of regarding the "UML as a family of languages" [5] and supporting the customization of the UML with so-called profiles may be regarded as promoting the DCL concept.

However, a DCL approach will only be economically viable if the cost of creating a DCL—which involves the adaptation of the syntax and semantics of a language—does not exceed the return on investment, i.e., the productivity gains. A DCL approach therefore needs to make the definition and derivation of languages as simple and intuitive as possible. As well as providing a chronological overview of the main concepts and technologies involved in defining and using DCLs, this chapter will therefore have a particular focus on the relative ease with which DCLs may be derived from base languages. Only language customization approaches and technologies that manage to minimize the cost of creating a DCL will have a long term future in software engineering.

2. Languages, Abstraction and Domain-Specificity

To set the scene for the tour of language customization technologies that follows, we use this section to explain some basic concepts and to introduce the case study that we use in the rest of the chapter.

2.1 Languages

In the following we do not distinguish between the traditional categories of modeling languages and programming languages, whose distinction has been considerably blurred by model-driven development approaches. Even though our examples are exclusively from the area of modeling, all observations and conclusions also apply to programming languages.

The task of a modeling language is to provide a set of concepts and an associated notation that allows the description of subjects of interest. In the case of software engineering, models are frequently used as construction plans, i.e., as descriptive models. In this case the subject is typically a software system to be built and sometimes includes the environment with which it interacts. The OMG's MDA Guide consequently defines the term "model" as follows:

> "A model of a system is a description or specification of that system and its environment for some certain purpose" [11].

It is, however, important to realize that "description" may refer to either of two modes of description: either one describes singular aspects of one particular incarnation of a system kind, i.e., one creates system snapshots, or one describes the universal aspects of all systems of a particular kind. Models using the singular description mode are *token models* [9] and *represent* the original subject. They are often expressed using UML object diagrams when referring to structural aspects of a system. Models using the universal description mode are *type models* and *classify* the original subject. They are often expressed using UML class diagrams when referring to structural aspects of the system. Although the description mode of a model is subject to interpretation—i.e., a class diagram may play the role of a type model (with respect to the described domain) but may at the same time also play the role of a token model (with respect to the implementation classes to be generated from it)—it can be useful to have a dedicated notation for each purpose. Some modeling languages, in particular domain-specific ones, only support one description mode explicitly. The UML supports dedicated notations for both token-level modeling (→ user instances) and type-level modeling (→ user types).

In order to be able to discuss the customization of modeling languages, we need to briefly introduce the fundamental ingredients of a language.

2.1.1 Abstract Syntax

The abstract syntax defines the basic set of concepts that can be used to make statements in the language together with rules for using them correctly. One particular useful way to specify the abstract syntax of a modeling language is to create a

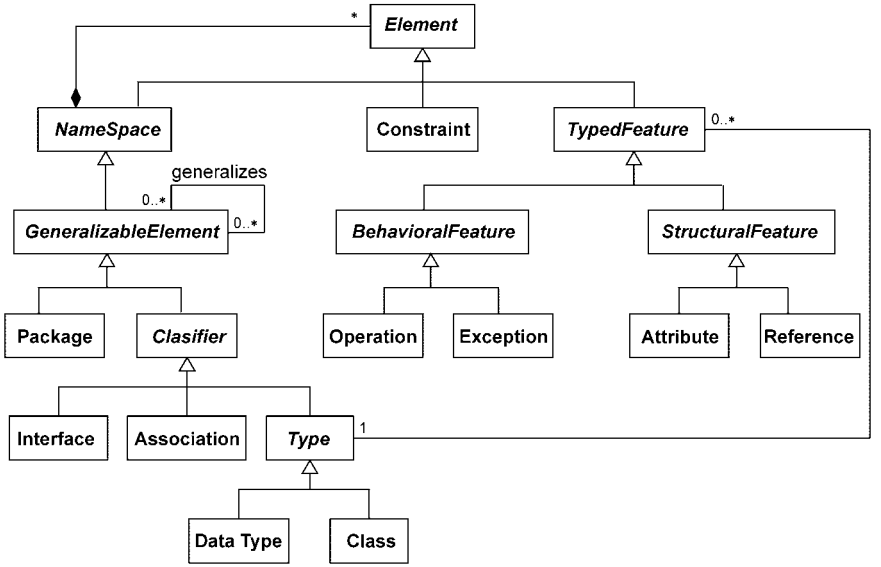


FIG. 1. Simplified UML metamodel.

model of the abstract syntax using the notation of the modeling language to be defined. This approach has been coined *metamodeling* since an (abstract syntax-) model is used to define the shape of other (user-) models. Metamodeling in this sense[1] is useful to minimize the intellectual burden on language designers, since it does not require knowledge of a dedicated formalism such as (E)BNF, but lets language designers work with a familiar notation.

Figure 1 shows a simple model of the abstract syntax of the core parts of a MOF-like [13] modeling language expressed in the notation of that same language. The UML and the MOF share a common core which can be used to describe the abstract syntax of other languages, including UML and MOF themselves.

2.1.2 Well-formedness Rules

By analogy to programming languages, modeling languages typically involve a number of rules which cannot be enforced by the abstract syntax definition. Also known as *static semantics* in the area of programming language definition, well-formedness rules thus complement the restrictions introduced by the abstract syntax

[1] This is, however, just one way to understand the term "metamodeling" [9]. We will discuss another interpretation in Section 6.

with further constraints which cannot be expressed using the metamodeling notation. For instance, the fact that identifiers are required to be unique within a namespace is a constraint that needs to be defined on top of the abstract syntax. Constraint languages such as OCL [18] can be used to formulate such constraints and attach them to the abstract syntax definition. Note that in contrast to grammars, metamodels may capture some well-formedness rules even without resorting to constraints, e.g., by using multiplicities.

2.1.3 Concrete Syntax

The concrete syntax defines the notation to be used to *present* structures which are *represented* using the abstract syntax. Textual languages also use the concrete syntax as a basis to generate import tools (i.e., parsers), but most modeling languages have a visual notation and only use the concrete syntax for displaying models.

2.1.4 Semantics

The final element of a language is its semantics which defines the interpretation given to sentences in that language, i.e., to expressions represented with the abstract syntax. Well-known ways of defining this interpretation include:

informal semantics: in terms of natural language descriptions;
operational semantics: in terms of relating abstract syntax concepts to execution machinery;
denotational semantics: in terms of mathematical mappings to known semantic domains;
translational semantics: in terms of mapping the language to a target language.

In all cases, the semantics are defined in terms of relationships to existing, accepted targets. In the context of tool-supported semantic definitions the *translational semantics* approach is the most common as it, in essence, corresponds to providing a compiler for a language. An *operational semantics* may be used to define an interpreter for a modeling language but then requires the additional specification of the execution machinery.

The economics of defining a DCL therefore depends on the ease with which the abstract syntax of a language, its well-formedness rules, concrete syntax, and its semantics can be defined and/or derived. Due to space constraints we will focus on the abstract syntax (see Sections 3–8) and only briefly touch upon the semantics (see Section 9). Before doing this, however, we need to clarify exactly what the term "domain-specific" entails.

2.2 Abstraction and Domain-specificity

Domain-specific approaches are often advertised as "raising the level of abstraction" and indeed many of their typical usage scenarios enable the specification of software at higher levels of abstraction. However, on closer inspection one observes that domain-specificity and abstraction level are in principle uncoupled and may be freely combined. We will briefly investigate the relationship between these two dimensions since it is instructive to see what choices are available in the two-dimensional design space and what role they play in typical development scenarios.

2.2.1 Abstraction

In the context of this chapter we use the term "abstraction" to be a measure of the extent to which a model or modeling language is geared toward *solution* technology. Thus, a model or language with a low degree of abstraction can be characterized as *solution-oriented* while a model or language with a high degree of abstraction can be regarded as *problem-oriented*. Notice that we are assuming that a model is a specification of a software system to be built and that the software is going to be executable. The ultimate "solution" technology from a software engineer's point of view is therefore binary machine code. However, several levels of abstraction exist on top of this, such as assembly language, C, operating systems, libraries, virtual machines, object-oriented languages, middleware platforms, etc. The more "problem-oriented" a model or modeling language is, the more it will address the question of "*what*" is done by the system and the more it will abstract away from the properties of the ultimate solution technology. The more solution-oriented it is, the more it will address the question of "*how*" it is done in terms of the ultimate solution technology. A solution-oriented language thus requires or allows—depending on whether one regards that as a necessity or a feature—a high degree of control over the realization aspects of a system. In contrast, a problem-oriented language focuses on the pure functional specification of a system without concern for realization aspects. The ambitious goal of domain-specific modeling—and of course model-driven development in general—is to allow the creation of models whose degree of abstraction is as close to the problem as possible, while still having a defined and automatic translation chain to the ultimate solution technology.

Of course, even without an automatic transformation chain, software engineering has been making use of different abstraction levels in the development process for a long time. A traditional waterfall process starts with highly abstract models in the analysis phase, progressing to a design phase with more solution-oriented models, and eventually ending up with very concrete and solution-specific descriptions of the system.

2.2.2 Domain-specificity

In the context of this chapter we use the term *domain-specificity* as a measure of the extent to which a modeling language is tailored to a certain application domain. A modeling language tailored to an application domain will directly support concepts which are naturally found in this application domain. A language with a low degree of domain-specificity contains general concepts which span multiple domains and is widely applicable (i.e., general-purpose), while a language with a high degree of domain-specificity contains concepts that are natural and well-suited for a particular domain and has a relatively narrow application domain in comparison. Domain-specific modeling languages therefore trade wide applicability for a minimum impedance mismatch which results in very intuitive and concise models.

Obviously, domain-specific approaches—such as Csound [4], a language dedicated to the creation of sounds and music, or Simulink [1], a language dedicated to simulation—have existed for a long time. However, the supporting tools and environment for such language were essentially developed "by hand" by the vendors that sell these systems, and users had to use the languages as they were delivered "out of the box". The new interest in DSL's stems from the fact that the definition of DSLs and their supporting tools is now a much less laborious task. As a result, they are now more frequently considered as a serious alternative or complement to regular software engineering practices.

2.2.3 Relationship between Abstraction and Domain-specificity

As illustrated in Fig. 2, abstraction and domain-specificity can be regarded as two orthogonal dimensions. The many ellipses in Fig. 2 indicate choices within the two-dimensional spectrum and therefore define the relative positions of the four displayed samples.

Figure 2 illustrates the fact that a modeling language has two properties—abstraction level and domain-specificity—which may occur in arbitrary combinations. For instance, the language in the bottom left-hand side corner of Fig. 2 is a highly general language, applicable to a wide range of application domains and a highly problem-oriented language, having very little concern for the realization of a solution. Such a language would typically be used in the "analysis" phase of a traditional software development project. Figure 3 shows GRL (Goal-Oriented Requirement Language) as a concrete example.

The bottom right-hand corner of our two-dimensional space features languages which are also general-purpose with respect to the application domain, but have a high-degree of solution-orientation. Java is an example of a widely-applicable language that requires/allows the specification of many implementation details.

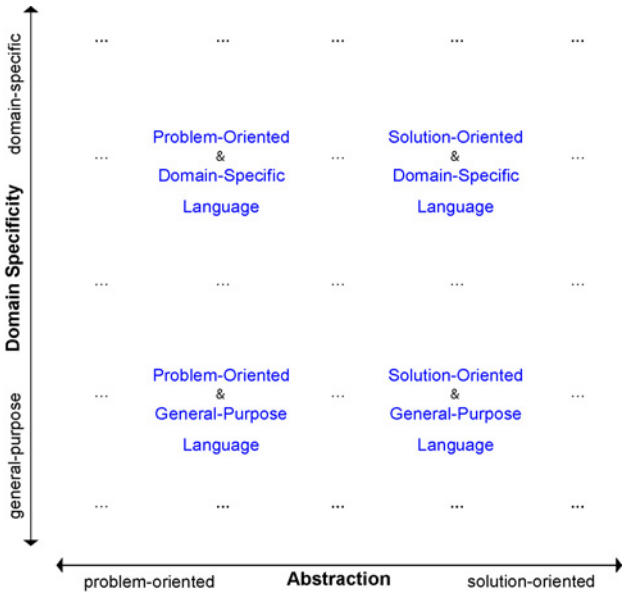


FIG. 2. Abstraction and customization dimensions.

The top right-hand corner features languages which are both highly customized to a certain application domain, such as e-commerce or gaming, and are designed to require/allow the specification of many realization details. It is fair to call these DSLs, however, the term DSM (Domain-Specific Modeling) is typically reserved for modeling languages in the top left hand corner of our two-dimensional spectrum. Such languages also focus on specific application domains, but they try to abstract away as much as possible from any realization concerns. An example might be a language for the specification of traffic light control systems offering concepts from that domain, such as cars, traffic lights, crossing etc. In contrast to a language with the same level of domain-specificity but a higher degree of solution-orientation, many realization aspects will be implicit and be either fixed or deferred to interpretation choices, such as various transformations employing different strategies for realizing control algorithms.

Figure 3, showing samples in the two-dimensional design space, points out that becoming more application domain-specific does not necessarily imply a higher level of abstraction. Likewise, a level of abstraction similar to that addressed by a dedicated “Pet Store” language may also be achieved with a general-purpose requirements language. The virtue of domain-specificity is the focus on a partic-

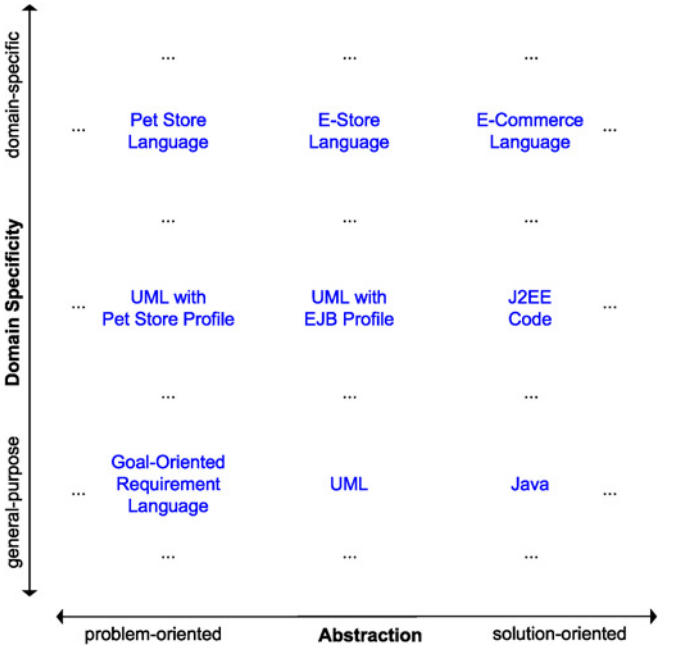


FIG. 3. Some samples in the language design space.

ular application area, allowing the models to be expressed using natural domain terms.

Although the two dimensions are theoretically independent, in practice there are correlations between them. Since typically the underlying goal of domain-specific languages is to provide a way for domain experts (but not computer scientists) to describe *what* they want, as opposed to *how* a system is going to be realized, most DSLs tend to be problem-oriented as well as domain-specific. In other words, most practical DSLs occupy the top left hand corner of the customization space. This frees developers from the need to work with solution-oriented concepts and allows them to write their high-level solutions using concepts natural to the domain in hand.

A common misconception with DSLs is that they always raise the level of abstraction at which applications are written beyond that of general purpose language such as the UML. This is because, like general purpose languages, they reduce the need for the human engineer to worry about the idiosyncrasies and properties of the underlying execution technology. However, this should not be confused with "raising the level of abstraction" relating to language concepts. What DSLs in fact do is to

provide the "right", or the "best" abstractions for the problem in hand rather than the most abstract. Using concepts that are too abstract for a particular topic can be just as problematic as using concepts that are too concrete. In both cases the existing abstractions have to be constrained or generalized to the required concepts which can involve just as much effort. DSLs avoid this problem by providing exactly the right abstractions for the problem in hand.

2.2.4 *Abstraction and Domain-Specificity in the Development Process*

Even though Figs. 2 and 3 suggest an orthogonal and free combination of *abstraction* and domain-specificity there are of course combinations which are more useful and popular. Figure 4 shows a number of models expressed using different modeling languages and highlights the typical location of primary artifacts within a given development style.

In Fig. 4 the bottom row corresponds to typical mainstream development practices, using general-purpose languages. The shaded area in the bottom row indicates how the effort of human engineers is typically distributed in most mainstream

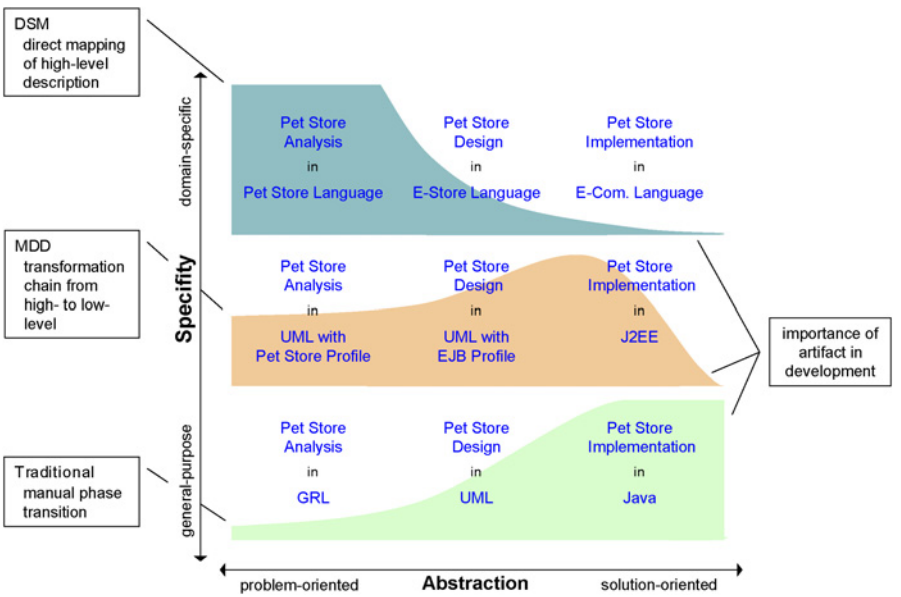


FIG. 4. Typical location of primary artifacts.

projects today. Generally speaking, the amount of effort spent on developing artifacts in traditional development is much greater at the solution-oriented end of the abstraction spectrum. This reflects the fact that the primary artifact is executable code (or something very close to it), with more high-level models playing only a supporting, but definitely secondary, role.

In contrast, the top row of the diagram represents the opposite end of the domain-specificity spectrum in which highly application domain-specific languages are used to develop software for very narrow domains. As shown by the shaded area in the top row, this has the very distinct effect of shifting the position of the primary artifacts. Two observations explain this phenomenon: First, truly domain-specific and solution-oriented languages—of the kind needed to operate in the top right hand side of the diagram—are today few and far between. Thus, in most domains it is simply not economically viable to develop software in a highly domain-specific but solution-oriented way. Second, even if such languages were more widely available there would often be little incentive to use them. That is to say, it is much easier to define automated transformations from problem-oriented to solution-oriented models in very specialized domains than it is in general purpose domains. For instance, it is much easier to define a transformation for a tailored version of "Class" (e.g., "ShoppingCart") compared to the general, underconstrained case where all is known about the source element that it is a "Class". Therefore, there is rarely a need to use a solution-oriented, domain-specific language since the desired executable system may be generated from a problem-oriented model just as well, with any realization decisions deferred to the transformation.

Finally, the middle row shown in Fig. 4 is characteristic of "model-driven development". The focus on solution-oriented modeling (i.e., coding) is much less than in the general purpose case, but higher than in the highly application specific case. The middle row also shows that the UML is not only "general-purpose" with respect to the application domain, it is also widely applicable with respect to the abstraction level targeted by corresponding models. Hence, in a transformation chain from problem-oriented to solution-oriented models using relatively small transformation steps between models, UML (often customized using profiles) may be used multiple times.

2.2.5 Orphan DSLs versus DCLs

As illustrated in Fig. 2, domain-specificity is a relative concept rather than an absolute concept. Certainly, there is no absolute threshold beyond which a language deserves the label "domain-specific" and below which it does not. However, although we do not provide any objective criteria here—and these may be hard to define—judging whether one language is more or less domain-specific than another is fairly intuitive.

In practice the label "domain-specific language" is used in a loose way to refer to any language which was designed with the deliberate intention of being more domain-specific than general, mainstream languages widely used for software development. The term "domain-specific language" is thus in practice often used to designate a language as being much more domain-specific than UML or Java, which are thought of as general-purpose languages.

We retain and use this loose meaning for the notion of a DSL. However, there is an important dichotomy which has a fundamental bearing on the topics we will discuss in the rest of the chapter: It is the difference between *orphan DSLs* and *customized DSLs*:

> An **orphan language** is a language that is defined without any formal or explicit reference to an existing language. The language therefore exists independently in the customization space (as illustrated in Fig. 2) and is not regarded as being part of a family of languages. An orphan language might be the result of reusing an existing language in its development, but as long as the reuse was done in an ad-hoc way, the result remains an orphan language. An orphan language which is domain-specific is known as an *orphan DSL*.
>
> A **customized language** is a language that is explicitly derived from and related to another language with the goal of supporting a different level of domain-specificity and/or abstraction. Such a language always exists in relation to another language. A customized language which is also a domain-specific language is referred to as a "*domain-customized language*" or DCL.

The notions of "orphan DSL" and "DCL" are thus disjoint and partition the set of DSLs. All DCLs are DSLs, but not all DSLs are DCLs. Formally, $S_{\mathrm{DCL}} \subset S_{\mathrm{DSL}}$ and $S_{\mathrm{ODSL}} = S_{\mathrm{DSL}} \backslash S_{\mathrm{DCL}}$. The basic premise of DCL-based development is that the effort involved in creating a DCL (and supporting environment) from a general purpose language like UML is lower than that involved in creating an orphan DSL. Furthermore, DCL-based development aims to draw on the familiarity of developers with the base language, requiring them to learn only the new, customized concepts.

2.3 Case Study

An important goal of this chapter is to present the ideas and technologies of DCL-based software engineering in terms of a single unified case study. We chose to use an electronic Pet Store system because even though it relatively simple, it is rich enough to exercise all the key issues in DCL-based development.

To help illustrate how DCL based development works and to provide various reference points along the customization and abstraction axes we will start by showing

three different kinds of models which occupy three different locations in the 2D customization space.

2.3.1 UML Analysis Model

Figure 5 shows a model that corresponds to the bottom-middle entry in Fig. 4. Due to space constraints we only show a rather reduced model, focusing on a few of the key abstractions in the Pet Store problem space. A comprehensive model for the Pet Store would be much larger, but including all this additional information would serve no didactic purpose. On the contrary, important phenomena would be more difficult to spot.

Note that all the elements in Fig. 5 are expressed using standard UML modeling element types, such as classes, attributes, associations etc. As long as the semantics of these match the modeled domain, the use of UML works well, without any impedance mismatch. Since the UML is not a universal modeling language this will not always be the case. Even when it is a good fit, though, it is important to note that there is more freedom in editing the UML diagram of Fig. 5 then one may desire. Let us assume the job of an application developer is to adapt the design to include two kinds of shopping carts, e.g., using different payment methods: He may choose to include a "contains" aggregation relationship between the new "ShoppingCart" element and "User", even though this does not make sense in the Pet Store appli-

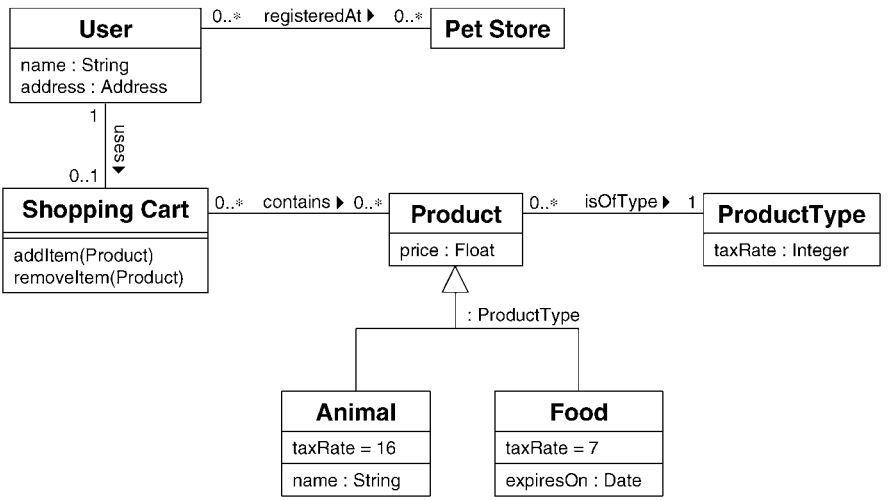


FIG. 5. Pet store analysis model.

cation domain. Generally speaking, whatever domain-specific rules may exist, they cannot be enforced by a plain UML diagram.

Figure 5 makes use of a so-called *powertype* in order to support the assignment of an individual "taxRate" to subclasses of "Product", and to make sure the latter have a "price" attribute, which they must inherit from "Product" by virtue of being an instance of powertype "ProductType" [14]. We will see how this design detail is treated differently in each of the following variants.

2.3.2 Domain-Specific Model

Figure 6 shows the model that corresponds to the top left-hand entry of Fig. 4, using a language specialized for the design of Pet Stores, a subclass of electronic retailing applications. As is typical for such a DSL, a dedicated, intuitive notation is used to depict the concepts from the application domain.

Note that the model using domain-specific notation needs less labels, since each of the connectors has a dedicated meaning and can be distinguished from others by shape, color, or what element types it connects. Also note that there is no explicit mentioning of a "ProductType". The Pet Store DSL may simply specify that all instances of product types (such as "Animal" or "Food") will automatically have a "taxRate" slot and a "price" attribute.

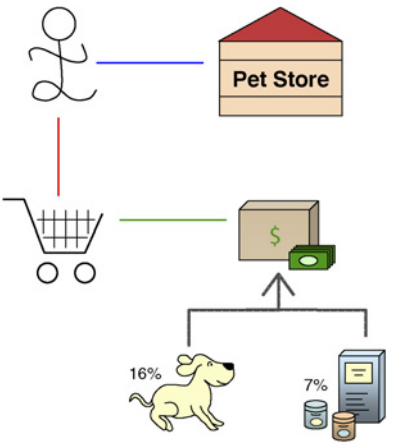


FIG. 6. Domain-specific pet store model.

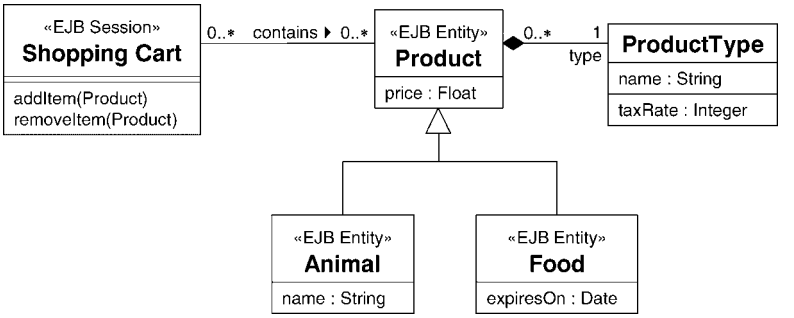


FIG. 7. Solution-oriented pet store model for the EJB solution domain.

2.3.3 UML Solution-Oriented Model

Figure 7 shows the model that corresponds to the entry at the center of Fig. 4. It features concepts similar to those of Fig. 5, but these are now more solution-oriented because they refer to J2EE and EJB (Entity Java Bean) middleware technology. The model of Fig. 7 actually uses UML's profiling mechanism (see also Section 3) to label the model elements in order to convey their mapping to different kinds of implementation concepts. The combination of UML + EJB profile can be considered to be a DCL.

In comparison to the model of Fig. 5 one may observe that the model of Fig. 7 not only adds mapping information, but also documents further realization choices. The "ProductType" concept has been realized as a reference from products to an object that both indicates the type of a product and contains a respective tax rate value. This reflects the fact that the intended solution technology for the EJB design only assumes an object level at runtime without the ability to attach values to classes, inquire about the type of objects and/or add new product types at runtime.

3. Derivation Types

In this section we consider the different kinds of derivations of a general-purpose modeling language that might be sensible from a language user's point of view. In other words, we consider what is the available range of options for creating one language by derivation from another *base language* and what their associated properties are. We are not so much interested in *how* the base languages or derivations are defined, but in the logical effect of the derivation in terms of the modeling capabilities it

offers to users and the extent to which it enables backward and upward-compatibility of models.

In general, all derived languages are created from a base language by applying some derivation specification to it. We can represent this formally as follows:

$$L_{\mathrm{d}} = L_{\mathrm{b}} \Phi D,$$

where L_{d} is the new language, L_{b} the base language, Φ the derivation operator, and D the derivation description. A derivation description, D, is composed of one or more primitive transformation operations, such as removing a language element, adding a language element, etc.

If a language is derived from a base language with some customization goal, i.e., the intention to place it at a different location in the customization space (see Fig. 3) we refer to it as a **customized language**. Customization therefore may involve a change to the domain-specificity and/or the targeted abstraction level of the language. There are a number of derivation descriptions which do not change a language's position in the customization space, e.g., renaming of concepts or replacing one paradigm with another without changing the domain-specificity or abstraction level. Such derivation descriptions then lead to purely derived languages as they have no customization goal in the sense of our coordinate system.

Obviously, one point in our customization coordinate system can be occupied by many languages, i.e., by languages addressing the same customization goal but using different means, such as different terminology, basic paradigm, etc. While they are equivalent in terms of reaching a customization goal, they will differ in terms of compatibility to the base language. A language that manages to attain a certain customization goal while maintaining a maximum level of compatibility (see below) to the base language we refer to as an *essentially-customized language* (ECL).

Essentially-customized languages are very useful because they define the optimal compromise between being widely understood as represented by modeling standards (by maintaining maximum base language compatibility) and being the best possible fit for an application domain as represented by DSLs (by being customized to the desired level).

When a new language has been derived from a base language it is important to determine which of the following relationships hold between the two languages, if $L_{\mathrm{d}} = L_{\mathrm{b}} \Phi D$:

$$\textit{non-conformant derivation} \quad \neg\exists m_{\mathrm{b}} \in L_{\mathrm{b}}: \quad m_{\mathrm{b}} \in L_{\mathrm{d}}, \tag{1}$$

$$\textit{partially-conformant derivation} \quad \exists m_{\mathrm{b}} \in L_{\mathrm{b}}: \quad m_{\mathrm{b}} \in L_{\mathrm{d}}, \tag{2}$$

$$\textit{fully-conformant derivation} \quad \forall m_{\mathrm{b}} \in L_{\mathrm{b}}: \quad m_{\mathrm{b}} \in L_{\mathrm{d}} \quad (\equiv L_{\mathrm{b}} \subseteq L_{\mathrm{d}}). \tag{3}$$

If a derivation yields property (1) then no model that can be created with the base language is a valid model of the new language. This implies rather radical reductions and modifications to the base language. In practice, this will rarely occur, as it strongly questions the choice of the base language to start with. If property (2) applies, at least one or more of the models created with the base language are also valid instances of the new language. However, unless the property (3) also applies, it means some models are not. This has important consequences for the upward-compatibility of an organization's model base. If property (3) does not apply, then steps may have to be taken to ensure that an organization's existing model base does not fall out of date. Note that a derived language with property (3) is known to be an essentially-customized language, since it achieves an optimal result regarding the upward-compatibility of models in the base language.

In the following subsections we discuss three different kinds of language derivations that may occur in practice. Note that we will often use the term "language" to refer to the set of all instances that may be created with a given language definition, i.e., to the set of all conformant models of a language definition.

3.1 Reduction

Reduction is the simplest kind of derivation. A reduction does not make any new modeling constructs available to modelers—on the contrary, it takes them away. The reduced language thus offers a subset of the features of the base language. Two ways of creating a language subset in this way may be distinguished.

3.1.1 Destructive

A destructive reduction may make any change to the base language other than add new features. The result will be a language that is less expressive than the base language. However, because it is possible to remove features which were regarded as mandatory in the base language there can be cases in which a model of the reduced language will not be an instance of the base language. The right-hand side of Fig. 8

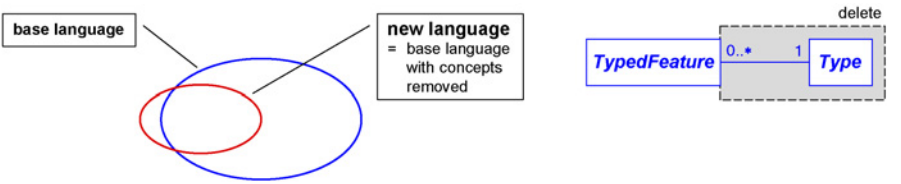


FIG. 8. Destructive reduction.

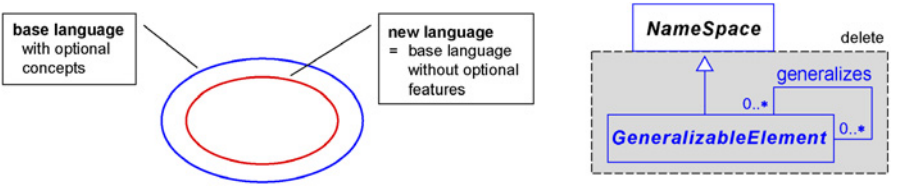


FIG. 9. Conformant reduction.

shows a change to the metamodel of Fig. 1 that illustrates this situation. If one removes concepts from a language that other concepts refer to in a mandatory manner, then models of the new language will not be conformant to the base language.

One strategy to gradually remove features without immediately invalidating models that use them is "deprecation" as known from Java. At first sight there might seem little use for reducing a language, but it can be quite useful to enforce company wide standards and modeling/programming practices. For example, a company might want to make sure that its modelers do not use "goto"-statements when modeling C programs or refrain from using powertypes in modeling.

3.1.2 *Conformant*

As illustrated in Fig. 9, a conformant reduction creates a new language ("language" here in the sense of the set of all conformant models) which is a proper subset of the base language, implying that only optional parts of the base language are removed. The right-hand side of Fig. 9 shows how a conformant reduction of the simplified metamodel in Fig. 1 could be achieved by removing the element "GeneralizableElement" from the metamodel.

On a technical level, the constraints of the new language must be within (i.e., stronger than) the constraints of the base language. This does not ensure that property (3) will apply after a conformant reduction, since a base language model may use optional features, but it ensures that every model of the derived language is a model of the base language.

3.2 Extension

Extension is the opposite of reduction. When a base language is extended, new modeling features are added and none are removed. However, there are two basic ways in which this can be done, resulting in two basic subcategories of extension.

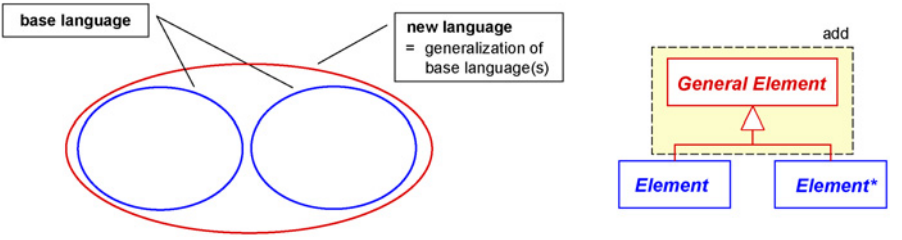


FIG. 10. Generalization extension.

3.2.1 Generalization

In generalization extensions, the new language is a superset of a number of base languages (see Fig. 10). Generalization extensions are a popular means to join hitherto independent base languages together in a unified extended language. As a consequence, a generalization extension satisfies the property (3) above—every model expressed in one of the base languages is also an instance of the new language. Apart from extending the notion of derivation to more than just a single base language, though, generalization extensions have the same subcategories as specialization extensions (see below). We will therefore treat them as a special case of disjoint specialization extensions that just feature more than one base language.

3.2.2 Specialization

In specialization extensions, the new language is a superset of the base language. As a result, property (3) applies and the new language is known to be a customized language. However, three different subcategories of specialization can be identified depending on whether or not the new language constructs can be understood in terms of the base language. Intuitively, we introduce a *base language view* on models created in the new language. If a new modeling concept can be seen using such a *base language view* then a new language construct is either a copy or a refinement of a base language construct. Technically, a parser for the base language will be able to read the new language construct, but will lose any additional information carried by the new construct. Intuitively, this is the case when a model expressed using the derived language can be regarded as being a *direct* instance of the derived language and an *indirect* instance of the base language. This is analogous to looking at an object through the interface of one of its class's superclasses.

3.2.2.1 Disjoint. A disjoint extension creates new modeling constructs which are entirely unrelated to the modeling constructs of the base language (Fig. 11). Hence the only modeling constructs which can be understood in terms

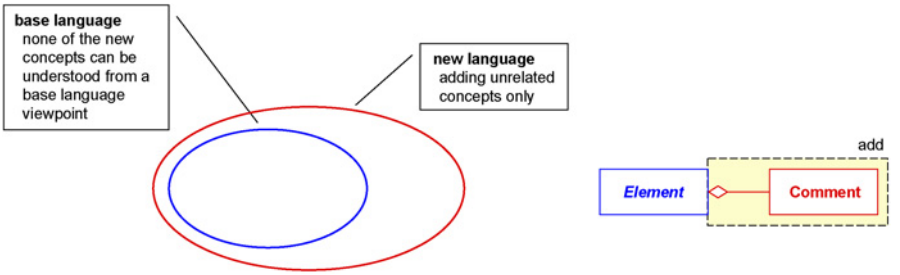


FIG. 11. Disjoint specialization extension.

of the base language are the base language constructs themselves:

$$\neg\exists m_d \in L_d \backslash L_b: \quad m_d \in_p {}^2 L_b.$$

Since such an extension results in a new language that is essentially the union of the base language plus an additional extension language, one may regard generalization extension as a special case of disjoint specialization, which happens to involve multiple base languages.

3.2.2.2 Conservative. A conservative extension does not introduce any entirely new modeling constructs but just refinements of existing ones (Fig. 12). Assuming that conservative extensions adhere to the Liskov Substitution Principle [10], all models expressed in the new language are therefore also indirect instances of the base language, i.e., the constructs of the new language can be understood in

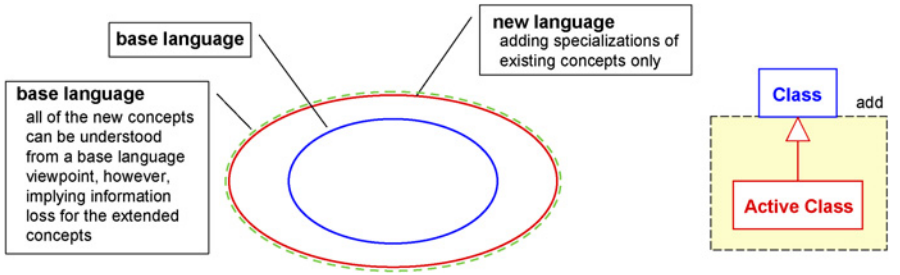


FIG. 12. Conservative specialization extension.

[2] "$\in_p$" means "element of, with respect to a base language view", i.e., a member test involving a projection of the element with respect to the base language.

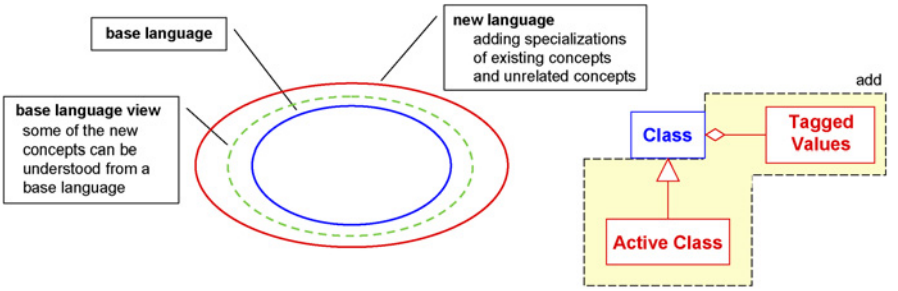


FIG. 13. Additive specialization extension.

terms of the base language,

$$\forall m_d \in L_d: \quad m_d \in_p L_b.$$

Consequently, there is a projection p which can turn every model of the new language into a model of the base language, albeit typically implying information loss.

3.2.2.3 Additive. An additive extension (Fig. 13) lies between the two extremes represented by the previous two alternatives. Some new language constructs from the extension are refinements of the base language; others are unrelated to the base language,

$$\exists m_d \in L_d \backslash L_b: \quad m_d \in_p L_b.$$

This results in a new language that may instantiate some models featuring new language constructs which will be indirect instances of the base language, i.e., can be entirely understood using a base language view, whereas the rest of the models featuring new language constructs cannot be understood using a base language view. Property (3) still applies, of course.

3.3 Modification

Modifications neither purely remove nor purely add modeling constructs, they may change existing ones and/or involve a combination of reduction and extension (Fig. 14). Since modifications involve reduction and extension (otherwise they would be plain extensions or reductions respectively), the corresponding number and kinds of subcategories exist (conformant/destructive × disjoint/conservative/additive), i.e., six in total. The resulting properties (1)–(3) can be calculated for modifications by simply combining the individual reduction and extension properties, with the weakest (strength increasing in the order of (1)–(3)) always taking precedence.

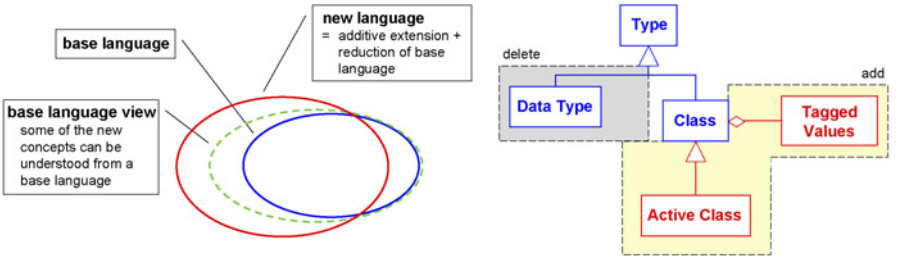


FIG. 14. Modification variant.

Note that a destructive reduction may be complete, so if all of the base language is removed prior to adding different, unrelated language concepts (disjoint extension) then property (1) applies, strongly questioning the choice of the base language which is not reused at all.

4. Lightweight Language Customization

Of all the customization types discussed in the previous section, "Conservative Specialization Extension" has the nicest properties. Not only does it ensure that all models expressed in the base language are expressible with the customized language (→ upward-compatibility of models) but also that all models expressed using the customized language are understandable (albeit with some information loss) through a base language view (→ backward-compatibility of models). Obviously, restricting oneself to just "Conservative Specialization Extension" is quite limiting, since then it is neither possible to remove concepts from the base language nor to introduce completely new ones that go beyond specializations of existing ones. Nevertheless, as it is an extension type that ensures maximum compatibility in both upward- and backward directions, it is an attractive option when customizing a language.

The so-called "Lightweight Extension" mechanism associated with the UML directly supports the "Conservative Specialization Extension" form of customization provided that one refrains from using constraints to remove features from the base language. The primary reason why this form was chosen for the UML, however, was the fact that the first generation of modeling tools that existed at the time had the UML language definition hardwired into their code. In other words, it was not possible to change the language supported by a tool without changing its source code. To get around this problem a so-called "Lightweight Extension" mechanism was introduced which allowed changes to the UML metamodel to be simulated within what

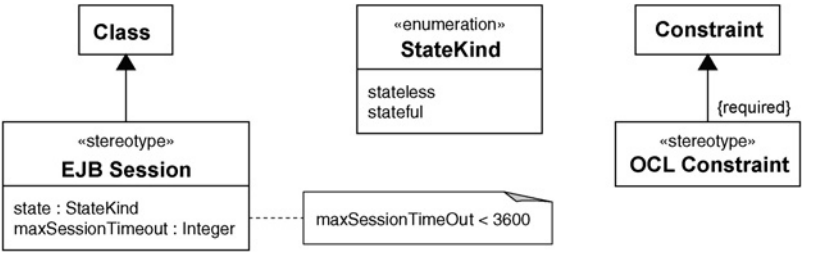


FIG. 15. Lightweight extension example.

is otherwise a normal model. In this section we give an overview of the UML 2.1 lightweight extension mechanism and illustrate how it can be used.

4.1 Extension Concepts

Lightweight extensions are defined using a combination of three distinct concepts: stereotypes, stereotype properties (also known as tag definitions) and constraints. These are illustrated in Fig. 15.

4.1.1 Stereotypes

Stereotypes are the core of the extension mechanism. A stereotype essentially has the effect of defining a specialization of an element in the metamodel of the base language. Every stereotype must therefore be defined with respect to at least one base element. To reflect the conceptual similarity to the specialization relationship, the extension relationship between a stereotypes and its base class has a very similar form. The only difference is that the triangle at the head of the extension relationship is solid (black) while the triangle at the head of the specialization relationship is empty (white). The left-hand side of Fig. 15 shows an example of a stereotype definition. Here "EJB session" is defined as an extension to the base abstraction "class" from the core UML metamodel in Fig. 1. This has the logical effect of defining a new virtual metaclass which represents a concept in a specific domain. The right-hand side of Fig. 15 is similar but illustrates the use of a required stereotype. It introduces the abstraction "OCL Constraint" as a mandatory specialization of the class "Constraint" from the core UML metamodel, that is, models using a profile (see below) containing this stereotype definition must only use "OCL Constraints".

From the language user's point of view the effect of attaching a stereotype to a base class in the metamodel is to allow instances of that class to be "branded"

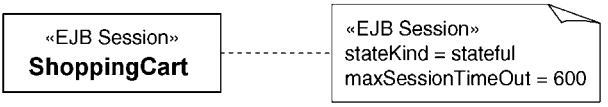


FIG. 16. Stereotype application example.

(i.e., marked) with the name of the stereotype. This marking is performed using the guillemot notation as shown in Fig. 16.

In this figure, "ShoppingCart" is a class that has been branded with the stereotype "EJB Session". This means that it is no ordinary class, but a class representing the notion of "EJB Session" (-bean) which is of importance in the domain of building EJB applications. To a layman such a branding adds little value, but to an expert in the domain branding a class with a domain concept such as "EJB Session" adds a lot of extra information to the model. Model element branding is therefore a good example of "Customized Modeling"—the idea that base language concepts are adapted to make the language more specific and expressive for a certain purpose.

4.1.2 Stereotype Properties

Stereotypes can be used to effectively add specializations of the classes in the metamodel of the base language. However, on their own they cannot add new attributes to the specializations. To do this stereotype properties, also known as tag definitions, must be used. A stereotype property is essentially a new attribute attached to the new conceptual metamodel element represented by the stereotype. The definition of the stereotype "EJB Session" in Fig. 15 includes the definition of two properties "state" and "maxSessionTimeout". These can be defined using all the normal types for attributes, including the values of an enumeration type as illustrated in Fig. 15. "StateKind" is an enumeration class which defines the values that the property "state" can assume.

The values of stereotype properties at the stereotype usage level are also known as tagged values. They effectively correspond to slots in normal class instances. The difference, as shown in Fig. 16, is that tagged values are defined in a separate model element. Fig. 16 shows the stereotype class "ShoppingCart" with tagged values "stateful" and "600" for its "state" and "maxSessionTimeout" properties respectively. The new UML 2 terminology—"stereotype property" instead of "tag definition"—is a sign of the growing recognition that tag definitions and tagged values are effectively nothing more than attributes and slots, but at the level of the metamodel and model respectively. However, semantically they are still a special extension concept that cannot be fully explained using the attribute/slot analogy. For example, instances of stereotyped classes can not refer to stereotype properties.

4.1.3 Constraints

When used in a lightweight extension, constraints typically define well-formedness rules for model elements branded with stereotypes. The stereotype definition "EJB Session" on the LHS of Fig. 15 features such a constraint definition. It serves to indicate that a well-formed model element branded with this stereotype must not have a tagged value for its "maxSessionTimeOut" tag that exceeds "3599".

4.2 Profiles

Although all parts of the lightweight extension mechanism can be used individually the intention is that they be used collectively to define a coherent customization for a specific purpose. Such a coherent set of extensions is known as a *profile* and as with all logically related sets of model elements in the UML they are usually collected into a common package. Figure 17 shows an example of how some of the extensions from Fig. 15 can be combined together into a profile package.

As illustrated in Fig. 17, a profile is always related to a base metamodel which it gains access to by means of the "imports" dependency. As can been seen from Fig. 17, profiles possess both of the key elements of a customized language: they

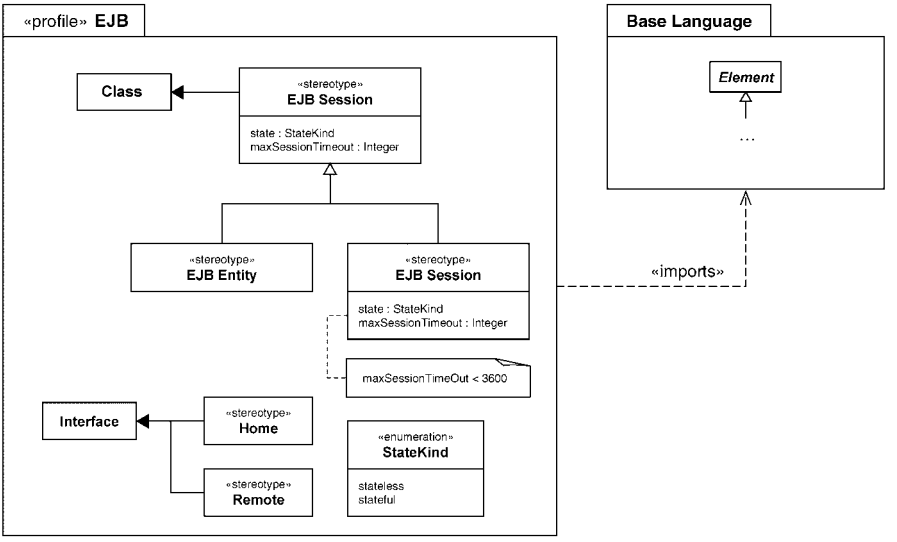


FIG. 17. J2EE profile.

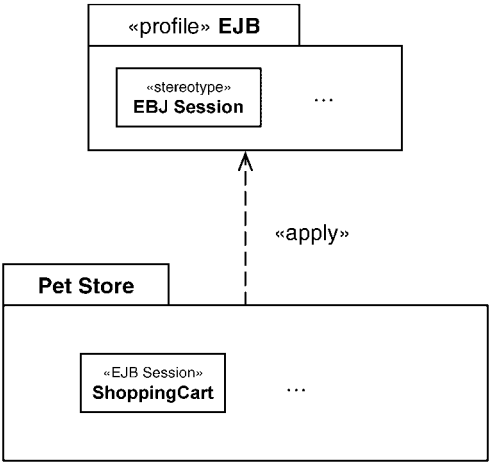


FIG. 18. Profile application.

define new abstractions which have a special meaning and thus can be used to create more expressive models, and they are explicitly customized from a base language.

In order to make the extension available for use in a new model it is necessary to "apply" that profile to that model. Figure 18 illustrates how the extensions in a profile are applied to another model using the "apply" dependency.

This effectively makes the customization available for use in the creation of a model, in this case the EJB-specific implementation of the Pet Store example illustrated in Fig. 7.

4.3 Lightweight Extension: Pros and Cons

The lightweight extension mechanism offered by the UML has some advantages and disadvantages compared to a heavyweight approach allowing liberal manipulation of the UML language definition.

4.3.1 Advantages

First, unless a profile selects a subset of the base language or constraints are used to effectively limit or remove the applicability of existing modeling elements, the defined customization will always be a "Conservative Specialization Extension" ensuring maximum upward- and backward-compatibility.

Second, stereotypes can be added and removed from model elements dynamically. Since stereotype applications are treated like annotations of regular model elements

it is possible to change the branding (i.e., stereotypes) of model elements just as easily as it is to change their links and their attribute values. The stereotyped element is unaware whether a particular stereotype is applied to it. In other words, stereotypes represent an implementation strategy for dynamically reclassifying model elements, including the addition/removal of properties.

Third, the profile mechanism allows a model element to be branded by multiple stereotypes, which is equivalent to it having multiple types. The corresponding heavyweight equivalent would be an explicitly defined metaclass which is a specialization of all of the base element classes which the branded model element is intended to be an instance of. While such an explicit combination could be used to resolve any conflicts arising from the combination of various stereotypes, one would have to create one explicit combination class for all possible stereotype combinations and for all base classes to which the stereotype is applicable. In this basic form, this would not be a viable approach because of the combinatorial explosion of cases.

4.3.2 Disadvantages

First, even if used to their maximum effect profiles can only support a combination of destructive reduction (through selecting a subset of the metamodel to modify and constrain) and conservative specialization extension (through stereotypes and stereotype properties). The important need to extend the base language with completely new concepts that cannot be explained as specializations of existing ones is not supported. This limits the range of customized languages which may be defined through profiles.

Second, stereotypes and stereotype properties are additional concepts which could often be replaced with metaclasses and their attributes with an additional degree of expressiveness. Metaclasses may inherit from each other and may have associations. Attributes of metaclasses could be specified just like slot values for objects. Apart from the above mentioned dynamic properties and multiple applicability of stereotypes they just represent a lightweight way of metamodeling. The latter could be offered in various forms, allowing various kinds of modifications that yield the same guarantees for extensions as featured by stereotypes.

Third, since stereotype-based branding offers such a handy means of assigning a special status to a model element there is a great deal of confusion in the general modeling community over how exactly they should be used. In fact, as reported in [2], three different usage patterns can be observed in practice. Figure 19(a) illustrates the "official" usage mode in which the stereotype is used to classify the class it stereotypes. Here the concept "taxed" is correctly being used to brand a model element "Animal" as being conceptually of the type "taxed", i.e., this particular pet

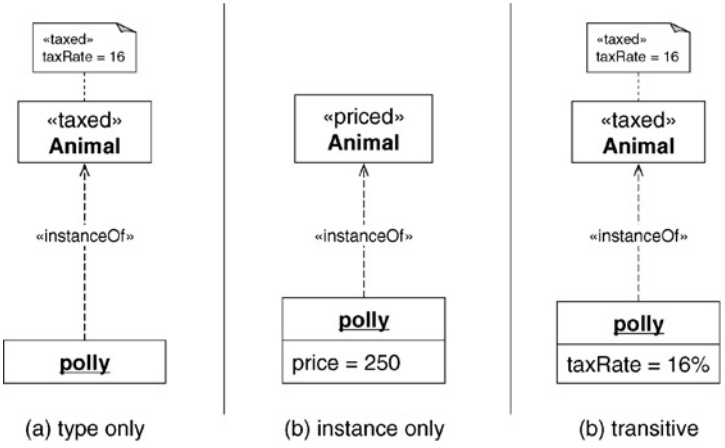


FIG. 19. Stereotype usage patterns.

store product kind is associated with a tax rate, not any individual product such as "polly".

However, an equally frequent usage in practice is to use the guillemot notation to name a generalization (supertype) of the branded class (see Fig. 19(b)). Here the intent of applying the stereotype "priced" to "Animal" is not to define additional properties for "Animal" but to define additional properties for instances of "Animal" such as "polly", e.g., that they have a "price" slot. Even though stereotypes are not designed for doing this, they are nevertheless often (mis-)used for exactly this purpose. A shorthand notation for specifying supertypes [3], e.g., "⟨priced⟩ Animal" would probably reduce the amount of improper stereotype uses of this form.

The third from, which is less frequent, is a mixture of the first two and uses stereotypes for both type and instance classification. Currently, the UML has just one official interpretation for stereotype applications and consequently offers only one notation. In practice, modelers find the need to cover all three cases shown in Fig. 19 and some notational means to distinguish them [2] would greatly reduce the confusion experienced by modelers.

5. Customization Support Environments

In Section 3 we described the different forms of language customizations that make sense from a theoretical point of view, and in Section 4 we discussed a way of supporting one of them—namely, the "lightweight extension" approach. How-

ever, we have not yet discussed the issues involved in supporting the others using a "heavyweight" approach. The goal of this section is to address the architecture of tools supporting "heavyweight" modeling and language customizations.

5.1 Multiple Modeling Levels

The first generation of modeling tools was only capable of supporting modeling in one fixed language. An environment intended to support modifications to the modeling language and/or support modeling in completely different modeling languages has to deal with at least three modeling levels, shown in Fig. 20.

The middle level shown in Fig. 20 contains the language definition which can be used to create user models at the bottom level. To support its modification or replacement this level must be user modifiable. As long as modeling environments treated the language as being fixed there was no need to represent the language definition in a modifiable way; the tool builders just chose on arbitrary approach and hardwired it into the tool. A flexible tool, however, has to support a further level (labeled "meta-language" in Fig. 20) to determine how language definitions may be specified. We have already mentioned that it is beneficial to define a language by using a metamodel. The level labeled "meta-language" contains a model that defines the concepts that can be used to define a language, the MOF being a concrete example. The UML superstructure [17] in turn is an example of content that may reside at the "language" level.

Note that there are two ways of looking at the middle "language" level, depending on whether one regards its contents as instances of the level above or as types for the level below. Figure 21 depicts both viewpoints within the middle level.

The left-hand side of Fig. 21 shows a language definition as an instance of the meta-language and the right-hand side shows the same language definition using the usual notation, i.e., as a number of types describing the contents of the level below. While both views are just different interpretations of the same set of concepts (the language definition) they flag an important issue: can the language definition data (as

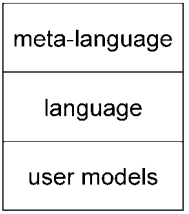


FIG. 20. Three-level language definition stack.

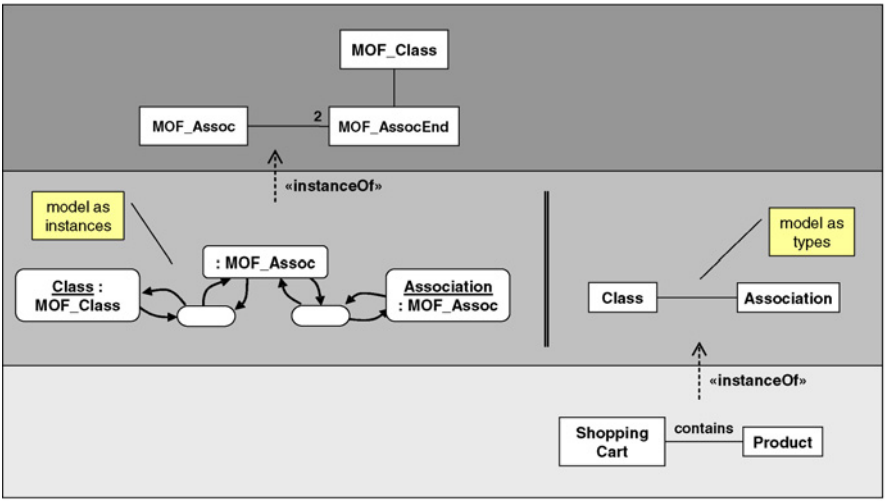


FIG. 21. Instance versus type view on middle model.

seen through the left-hand side view) be directly used to control (model-)data at the level below or is it necessary to promote the instances to types before they can fulfill this function?

5.2 Two-Level Architectures

The above mentioned "promotion step" is often used in today's tools since they are often implemented using only two levels (i.e., a type level and an instance level). Consider implementing the three levels shown in Fig. 21 using a typical object-oriented language like Java. The classic approach is to define Java classes for each of the elements in the top level, and to use Java objects to represent the language definition data in the middle level (see Fig. 22(a)).

However, these objects may not be directly used in the style just described to represent and control the data at the "user models" level. This is possible only if the language definition is available as types, i.e., classes (see Fig. 22(b)). To achieve this, the objects need to be "promoted" to types. Figure 23 explicitly illustrates the "promotion step" from objects to types.

We refer to an approach of the kind depicted in Fig. 23 as a "cascading architecture" since two-level implementation "windows" are used repeatedly to cover multiple modeling levels. Each of the tool's partial architectures covers only two levels, but in combination a hierarchy of multiple levels may be constructed (see

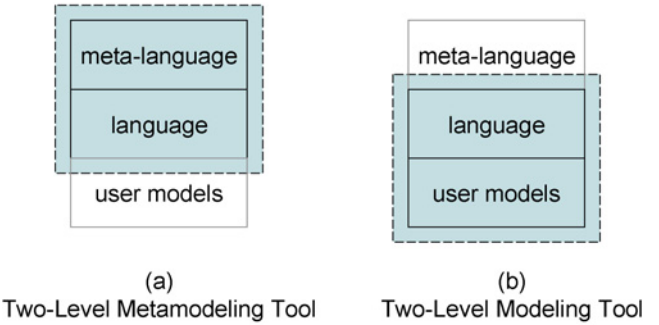


FIG. 22. Two-level tools.

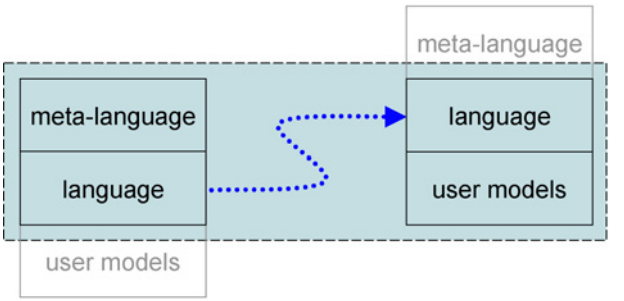


FIG. 23. Cascading architecture.

Fig. 22). The advantage of this cascading architecture is that one never needs to support more than two levels at the same time and access to or manipulation of data is always very efficient, since the access types (the top level of the two) are known at compile time. The disadvantage, however, is the need to perform the promotion step which represents a similar development overhead to compilation in the context of programming languages (in comparison to direct interpretation). This approach is sometimes referred to as a generative approach since each tool is usually generated automatically from the objects at the immediate level above. A significant number of today's metamodeling (language customization) tools are based on this architecture, relying on the *generation* of modeling tools.

5.3 Flattened Architectures

The alternative to the generative, cascading approach is to dispense with the idea of always associating metamodels with the type level (e.g., Java classes) of the sup-

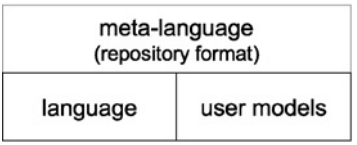


FIG. 24. Flattened architecture.

porting implementation technology. This requires that elements of user models will not be instances of metamodel elements anymore in the usual sense (e.g., using the Java object-class relationship). Instead, the "instance of" relationship is just modeled as data. As a result, in our example the user models, the language definition and the relationship between them are just data at the same level.

Figure 24 shows how what we call a "flattened architecture" accommodates two of the three levels of Fig. 20 in just one level. Figuratively speaking, it squeezes two of the levels into one. One of the advantages of this architecture is the ability to change content at the language level and immediately use the new definition for the creation of user models. Furthermore, even when user models may no longer conform to a changed language definition, they are still represented using the common (meta-language) format. In comparison, models represented using a two-level cascading style depend on their language definition to stay fixed. A change to the language implies the need to migrate models represented in the old language definition to the new language definition. Note that in Fig. 24 the role of the "meta-language" level has turned into providing a common repository format, rather than defining the primary concepts with which to build language definitions. It can, however, be used for both roles—the MOF being a prominent example.

A disadvantage of the flattened architecture is reduced performance related to the accessing and manipulation of model data since its definition has to be interpreted. Moreover, access to the data has to occur in a reflective style, using very common access methods, parameterized with (elements of) the language definition. This is less straightforward and cannot be type-checked using the implementation technology (e.g., Java type checking).

Note that the "user models" level may in general be internally structured. Users can model at the type level just as easily as at the instance level. Figure 25 shows how the "user models" level may be subdivided into two such levels.

We did not arrange the "language", "user types and "user instance" levels in a linear fashion because the "language" level describes the contents of both "user types" and "user instances" levels. Figure 25 hence describes the OMG's so-called four-layer architecture, with "meta-language" corresponding to the MOF (M_3), "language" to "UML metamodel" (M_2) and "user types"/"user instances" to "user mod-

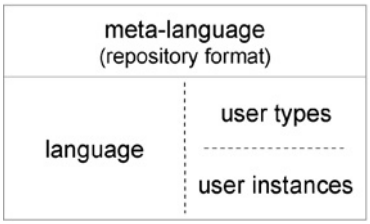


FIG. 25. Detailed flattened architecture.

els" (M_1). The fourth layer (M_0), the modeled world, is not shown in Fig. 25, since it is outside tool or language definition stacks.

In the next section, we will take a closer look at the level boundary between "user types" and "user instances", since it is different in nature to any of the other boundaries in Fig. 25.

6. Ontological Metalevels

In the previous section we focused on a relationship between models and their (meta-)models which we refer to as *linguistic* classification. This reflects the widely accepted approach of characterizing metamodels as being *language definitions* and "user models" as being statements expressed in that language. The "meta-" prefix is motivated by the location of language definitions in the original, stack-oriented visualization of description levels that was put forward by CDIF [15] and then became the foundation of the UML infrastructure [16]. This hierarchy (of which the three top levels are shown in Fig. 20) assumes that "user models" are type models (of user data) and uses the same type-classification to describe "user models" with a (language definition) model. Hence, the language definition apparently represents second order classification with respect to user data, and it is therefore natural to think of it as being "meta" with respect to "user data". Note that this line of reasoning works well in the CDIF context, since user models are exclusively type models. However, the case of the UML is more involved as already pointed out in the previous section. In the next section we investigate why the term "metamodel" is not as appropriate in the case of the UML [9].

One should always qualify the term "metamodel" since although linguistic metamodeling is important there is an important alternative: ontological metamodeling. Consider Fig. 25, where we have drawn a level boundary between "user types" and "user instances". Although we have drawn this boundary one would normally not think of a user type model as defining a language of which user instance models are

statements expressed in that language. The underlying meaning of user types is of course to create domain abstractions in terms of classification. We therefore refer to the "instance of" relationship between a user instance and a user type as *ontological classification* (as opposed to *linguistic classification*).

Note there is no specific reason why the number of user modeling levels should be limited to two. It is perfectly possible to envisage three or more ontological classification levels. Elements at the third level (i.e., the types of the types of the instances) would warrant the label of metatypes, due to their second order classification property. To distinguish such types from "language definition"-types we refer to the former as *ontological metatypes* or *domain metatypes* and the latter as *linguistic* (*meta-*)*types*, since the former are concerned with describing domain knowledge rather than language features.

Discussing the nature of, and potential support for, ontological metatypes is important in the context of this chapter, since language customization is often performed not to enrich one's vocabulary of modeling primitives, but to capture ontological information due to a lack of a better alternative. After the following subsections which clarify the notion of ontological metatypes and their support we will further discuss the relationship between ontological (meta-) modeling and language customization.

6.1 Ontological Metatypes

A good example of a domain concept which naturally corresponds to the notion of a metatype is the concept of "ProductType". This is an important concept in the Pet Store example because it classifies the different *kinds* of products that are sold in the store. Different kinds of products, such as "Animal", are naturally thought of as instances of "ProductType". However, specific product objects, such as individual animals or concrete food items, could not naturally be thought of as instances of "ProductType". For example, it does not make sense to say that "parrotInstance2947" (alias "Polly") is a "ProductType". The most natural relationship between these concepts is illustrated in Fig. 26. Object "parrotInstance2947" is an instance of "Animal" and "Animal" is an instance of ProductType. Since Fig. 26 shows concepts occupying three distinct ontological levels, "ProductType" is naturally thought of as an (ontological) metatype of "parrotInstance2947".

Even though ontological classification hierarchies will naturally be much less deep than generalization hierarchies, it is easily possible to think of requiring four or more classification levels. For example, the Pet Store owner may receive frequent requests for a "parrot just like Polly" or children saying "I want a Lassie", intending to refer to a Collie that looks like the famous movie character. The Pet Store design may therefore be changed to no longer interpret "Polly" and "Lassie" as referring to individual

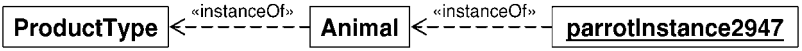


FIG. 26. Ontological meta levels.

animals but to a class of animals that looks like the prototypical instance. In this case, element "Polly" becomes a type for all parrots looking like "Polly", creating a four-level classification hierarchy.

At this point it is important to address a potential source of confusion—the mixing up of ontological metatypes and supertypes. They are often mixed up because they are both more abstract than the notion that they classify or generalize and in natural language they can both be related to the more concrete notion using the phrase "is-a". In other words, if in a particular domain it is natural to say "X is a Y", it is possible that Y may be a metatype or a supertype. However, the right choice can easily be determined with a simple test: If instances of "X" can be considered to be instances of "Y" as well then "Y" is a supertype not a metatype. Consider the situation in Fig. 27.

In this example, the class "Animal" has both a superclass "Product" and a metaclass "ProductType". As already mentioned above, when discussing the relationships between the concepts in this model in natural language it would be natural to state that "Animal is-a ProductType". However, it would also be natural to state that "an Animal is-a Product". So why is one cast as a classifier of "Animal" and the other as a supertype? The answer is hinted at by the slightly different forms of the above statements. In the second we prefixed the statement with the article "an", while in the first there was no such article. This reflects the fact that in the first case, as already mentioned above, it would not be natural to state that "parrotInstance2947 is-a ProductType" whereas it would be natural to state that "parrotInstance2947 is-a Product". This indicates that "parrotInstance2947" can be regarded an *indirect* instance of "Product" as well, as shown in Fig. 27. This therefore makes the set of animals a subset of the set of products and naturally places them in a specialization relationship.

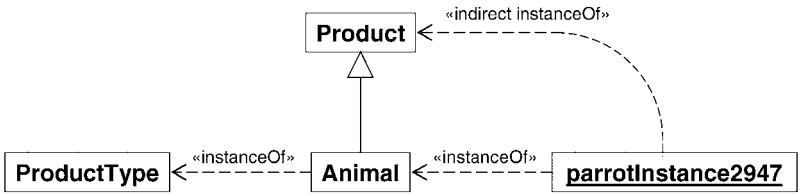


FIG. 27. Supertypes versus ontological meta types.

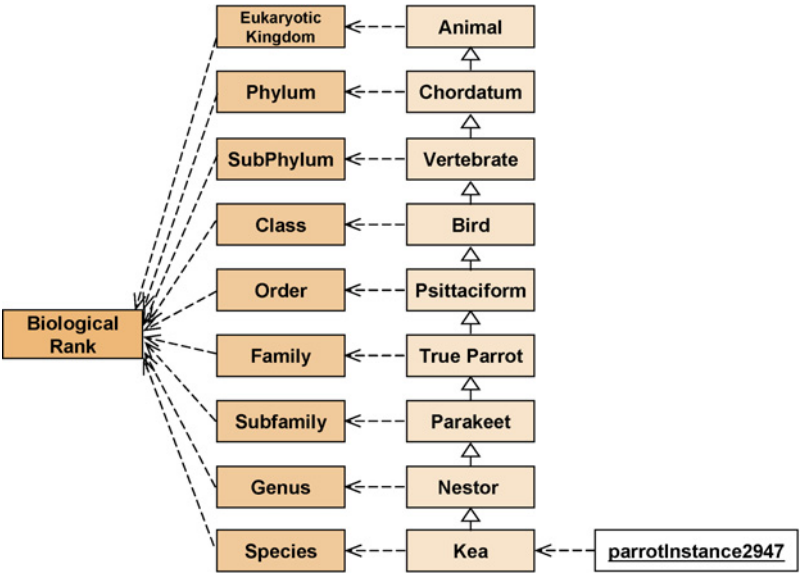


FIG. 28. Biological classification.

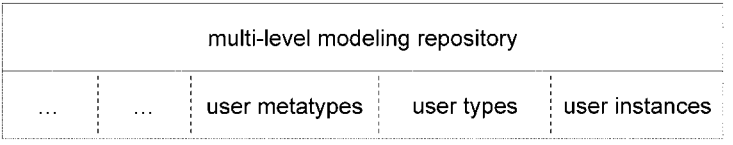


FIG. 29. Multi-level modeling support.

A natural example of a domain with multiple ontological levels and multiple specialization levels is the Linnaean classification system used in Biology. Figure 28 shows how generalization and ontological classification can be used together to place our sample parrot within the kingdom of animals. Of course, this is overkill for the Pet Store application. A cut down version of this scheme, however, could in principle be used to organize the products sold by the store.

Consequently, the architecture shown in Fig. 25 should be extended to look like the architecture in Fig. 29, i.e., it should feature an unbounded number of ontological modeling levels.

6.2 The Dual Facet Property

As soon as more than two ontological levels are used, e.g., as in Fig. 26, an interesting question arises: what exactly is an element (e.g., "Animal") that is both the instance of a type ("ProductType") *and* a type for its instances ("parrotInstance2947")? The top part of Fig. 30 highlights the two ways one can look at "Animal".

In comparison to Fig. 26 we have added some sample properties to show that in general an element (here "Animal") has both slots ("taxRate") and attributes ("price" and "name"). The topmost occurrence of "Animal" depicts the "instance" role of "Animal" in which it is seen as an instance of its classifier "ProductType". We have therefore used the UML notation for objects to reinforce this perspective in which "Animal" has a "taxRate" slot with a value of "16" by virtue of the fact that its classifier, "ProductType" has an attribute "taxRate". This is a normal UML class/object relationship in which "Animal" plays the role of an object.

Just below in Fig. 30 we show the "class" perspective on "Animal" in which it is seen as a classifier for its instances at the level below. We therefore use the UML notion for classes, declaring two attributes for "Animal", one of type Float and the other of type String. Instances of "Animal" such as "parrotInstance2947" thus have corresponding slots with appropriate values for these types, "2502" and "Polly", re-

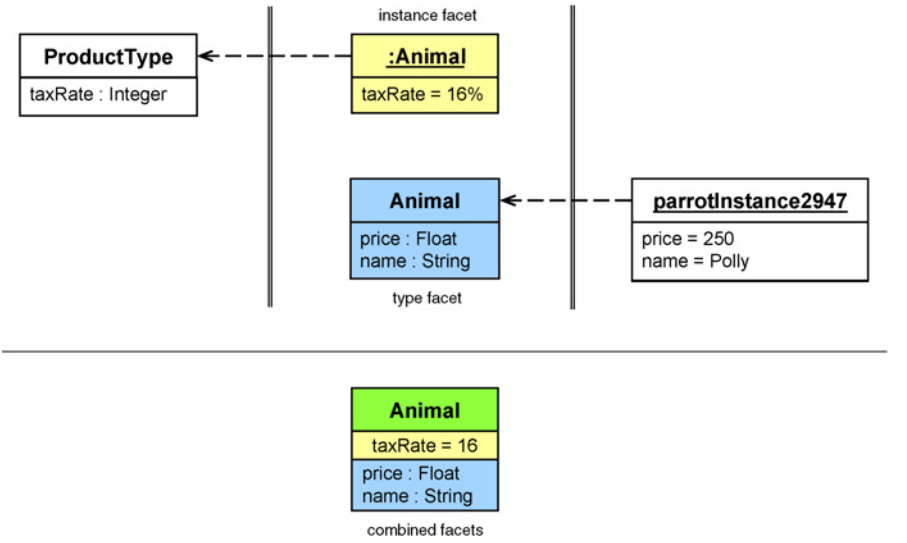


FIG. 30. Two views of intermediate-level model elements.

spectively. Again this is a normal UML class/object relationship but in this case "Animal" plays the role of a class.

The bottom part of Fig. 30 shows how both "class" and "object" perspectives on "Animal" can be integrated into a single model element. Note that this is not official UML syntax. The UML's support for more than two ontological levels is rather limited as described in the following subsections.

Summarizing, when there are more than two ontological classification levels, it is necessary to explicitly recognize that model elements—with the exception of those at the top and bottom levels—have two distinct facets, both of which are a real and meaningful part of the element. This is highlighted in Fig. 31 which shows a *clabject*, i.e., a combined "class" and "object". The instance facet of a clabject represents its object-like properties and the type facet its class-like properties.

Note that although we have used the term "perspective" before, the facets of a clabjects are not to be confused with the "instance" and "type" viewpoints shown in Fig. 21. These viewpoints provided views of the same set of data either as instances or as types. The instance and type facet of a clabject are two different entities and a perspective, as mentioned above, may only mask one of them, but does not interpret the same facet either as an instance or a type property. Elements that populate a multi-level modeling environment are, hence, neither classes nor objects as understood in classic two-level modeling environments.

The dual facet property of clabjects is not an issue that occurs in isolation with respect to architectures, such as the one shown in Fig. 29. The dual facet phenomenon arises whenever multiple ontological levels are modeled. The next subsections discuss ways of providing at least three ontological levels in the absence of a supporting architecture such as the one in Fig. 29.

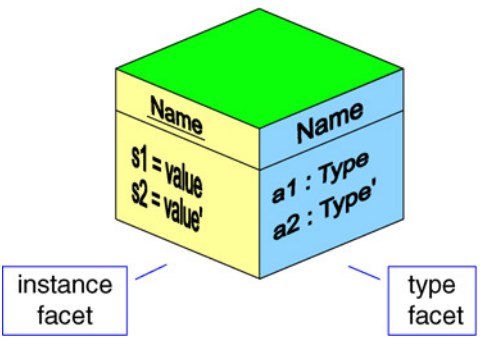


FIG. 31. Clabjects.

6.3 The Item-Descriptor Pattern

Since ontological metatypes such as "ProductType" in Fig. 26 frequently occur in practice modellers have always found ad-hoc ways of modeling them. A well-known workaround that is used in two-level environments is the "Item Descriptor" pattern [6] which is often employed when domain types need to be available as runtime data [7]—that is, when domain types are required to be treated as instances. Sometimes the motivation is to simply have a way to store and change type related data (such as tax rates) [6], sometimes one actually needs to support the creation of new types at runtime [8].

The basic idea of the Item-Descriptor pattern is to represent both application instances and types as objects. Figure 32 shows this idea in the context of our case study.

The "instance of" relationship between "Animal" and "parrotInstance2947" is part of the runtime data and is described by the "instance of" association between "ProductType" and "Product". Ignoring the type "Product" in Fig. 32 —which is just required in order to have a way to produce instances like "parrotInstance2947", since the *object* "Animal" cannot do it—the hierarchy between "ProductType", "Animal", and "parrotInstance2947" is exactly the same as in Fig. 26, except that the "instance of" relationship between "Animal" and "parrotInstance2947" is user modeled instead of being supported by a multi-level modeling environment. This observation highlights the "workaround" character of the Item-Descriptor pattern which represents a poor man's way of modeling at three ontological levels. Note how the instance and type facets of "Animal" are split between "Animal" and "Product". The disadvantages of applying this pattern include the lack of any supported type checking between "Animal" and "parrotInstance2947", the lack of support for inheritance between modeled types, and the addition of accidental complexity for scenarios that

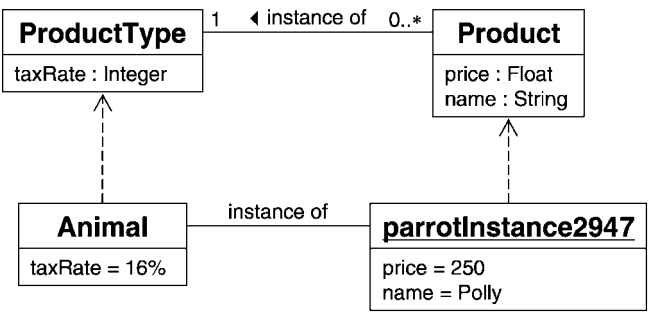


FIG. 32. Item descriptor pattern.

could be more concisely described using proper support for ontological metamodeling.

6.4 Powertypes

The above subsection described a way to introduce more ontological levels by using a two-level technology. The UML, in fact, has some limited support for dealing with more than two-levels. The best approximation of ontological metamodeling support in the UML is the powertype concept. An example use of a powertype in our Pet Store example is shown in Fig. 33 (which is an excerpt of Fig. 5).

The basic idea behind the powertype concept is to represent the separate facets of a clabject as separate model elements and to establish a constraint between the separate model elements. In Fig. 33, the abstractions that we wish to think of as clabjects (with two facets) are "Animal" and "Food". The type facet of these abstractions is defined in their common supertype, "Product", since any attributes and associations defined on "Product" influence the form of their instances. The instance facet of these abstractions, in contrast, is defined in their common classifier, "ProductType", since any attributes and associations defined in "ProductType" will be possessed by "Animal" and "Food" in the form of slots and links respectively. The classifier of the clabject abstraction(s), in this case "ProductType", is referred to as the powertype of the supertype ("Product").

The aforementioned constraint between the elements defining the instance facet ("ProductType") and the type facet ("Product") for clabjects is the requirement that every instance of a class designated to be a powertype ("ProductType") must also be a subclass of another designated class ("ProductType"). This ensures that all instances

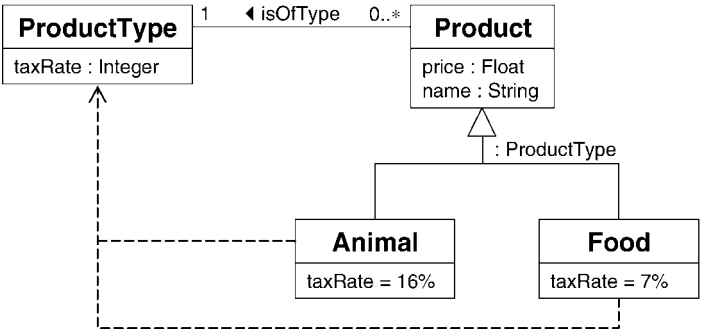


FIG. 33. Powertypes.

of a powertype also obtain the desired type facet, which cannot otherwise be achieved in a direct manner.

Note that the desire to influence the type facet of an element arises from the fact that a particular concept (here "ProductType") not only wants to specify properties of its instances (e.g., "Animal") but also properties of the instances of its instances, e.g., the fact that "parrotInstance2947" has a "price" slot. One would like to associate this property with the "ProductType" abstraction, since whenever a new instance of "ProductType" is created, this instance needs to be guaranteed to have a "price" attribute. We refer to such requirements as the need for *deep characterization*.

Obviously, when dealing with only two-level scenarios such a need does not arise as the one type level can always control the one instance level. In a multi-level classification hierarchy, however, the limitation of a type to only influence its instances at the intermediate level below, without influence on its instance's instances, may

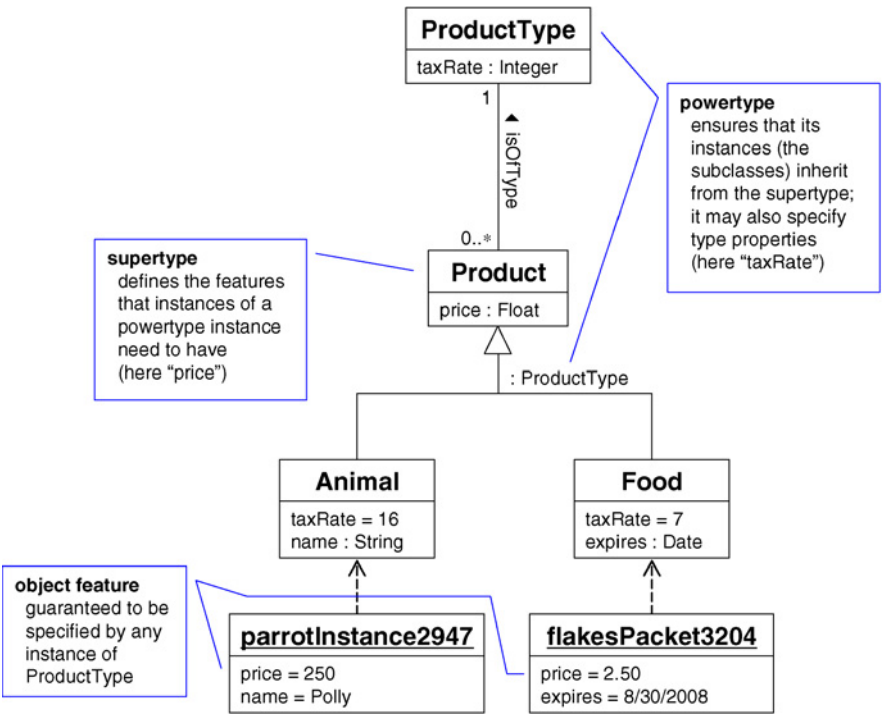


FIG. 34. Deep characterization using powertypes.

become an issue. “Deep characterization” then needs to be addressed, independently of whether workarounds for representing the multiple levels are used or not.

As illustrated in Fig. 34, powertypes support “deep characterization” by requiring that all instances of an ontological metaclass also be subclasses of a particular class. It is this latter superclass which defines the attributes that ensure that all instances of the instance of the powertype abstraction have a particular slot.

6.5 Deep Instantiation

Although the powertype mechanism is able to address “deep characterization” it does not do so in a very elegant way. The basic problem is that in order to define the type facet of a powertype’s instance an extra superclass has to be added. In essence, therefore, two model elements (“ProductType” and “Product”) are being used to capture a single clabject abstraction.

In order to support a more elegant and direct way of representing deep characterization two ingredients are needed. One is a notation which combines the type and instance facets of a clabject in a single model element (see the bottom of Fig. 30). The other is a concept that allows the intended characterization depth of a clabject’s properties to be specified. Figure 35 illustrates our example using the above mentioned ingredients.

The key idea behind the concept of characterization depth is to unify the concepts of “attribute” and “slot” into the notion of a *field* and to indicate whether a field should be thought of as a slot or an attribute by assigning it a potency value. The potency value of a field indicates how many times its can be instantiated (with respect to instantiation depth) with the understanding that instantiation reduces the potency of a field by one. A field with potency 2 (such as the “price” attribute of “ProductType”) needs to be instantiated twice in succession before it becomes a slot, while a field with potency 1 (such as the “price” attribute of “Animal”) becomes a slot after one instantiation. A field with potency 0 (such as the “price” slot of “parrotInstance2947”) cannot be instantiated at all. A field of potency 1 thus corresponds to a regular attribute and field of potency 0 corresponds to a regular slot.

Potency values may not only be assigned to fields, but also methods, and more importantly here to clabjects. The same principles apply as for fields, so for example “parrotInstance2947” can be instantiated from “Animal” since the latter is an instance of a potency two clabject.

Modeling scenarios such as the one depicted in Fig. 35 using clabjects and potency reduces the number of modeling elements needed to address “deep characterization” thus reducing the accidental complexity in a model.

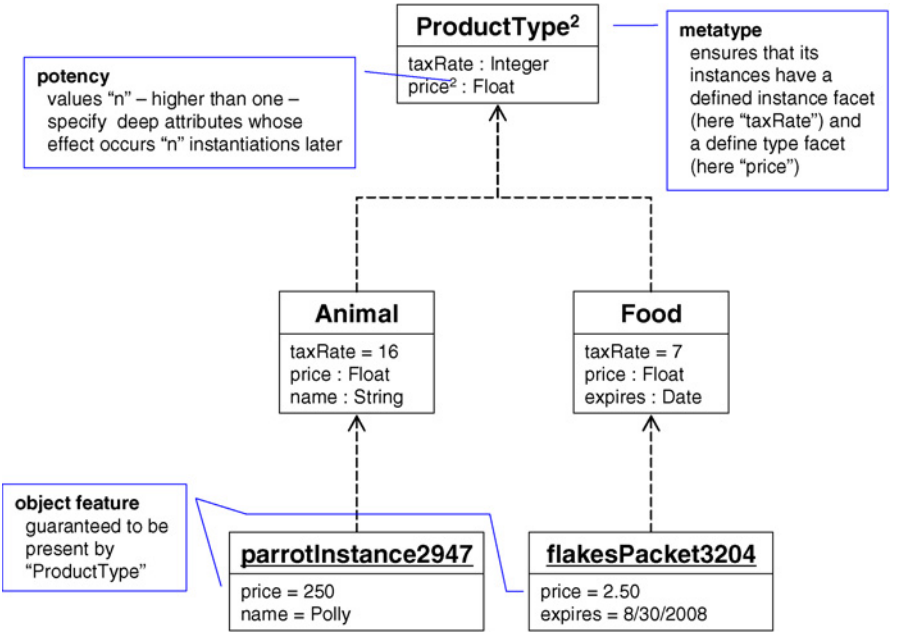


FIG. 35. Deep classification through deep instantiation.

7. Orthogonal Classification Architecture

In the previous section we introduced the notion of ontological metatypes and explained how they can help model domains with multiple classifications levels. In this section we show how ontological and linguistic metamodeling can be accommodated within a single unified modeling framework.

7.1 Strict Metamodeling

The first and most important issue to consider when defining the architecture of a multi-level modeling environment is how the notion of "level" is actually defined. So far we have appealed to intuition and have also used the "instance of" relationship in a liberal way without specifying whether we mean the "linguistic" or the "ontological" case. Without becoming more precise in this respect and, in particular, giving the term "level" a meaning, it is not possible to unambiguously decide where modeling elements, even fundamental ones such as "Class" or "Object", reside in a multi-level classification hierarchy.

There are only two basic ways of approaching this issue. One way is to de-emphasize the idea of levels altogether and simply regard all model elements anywhere in the modeling framework as inhabiting some kind of model "soup". With this unstructured approach, there are no rigidly defined levels and model elements are just placed anywhere. The other way is to rigidly enforce the idea of levels and to adopt the notion that every "instance of" relationship crosses a level boundary and that the "instance of" relationships thus implicitly define the hierarchies of levels. In other words, every classifier must be at a different level from its instances—namely at the level immediately above them. This notion, traditionally referred to as "strict metamodeling", is defined as follows:

> *In an n-level modeling architecture,* $M_0, M_1, \ldots, M_{n-1}$, *every element of an* M_m*-level model must be an instance of exactly one element of an* M_{m+1}*-level model, for all* $0 \leqslant m < n - 1$, *and any relationship other than the "instance of" relationship between two elements X and Y implies that* $\text{level}(X) = \text{level}(Y)$.

7.2 Ontological versus Linguistic Dimensions

Because of its value in helping to organize the various languages and models involved in model driven development, strict metamodeling has for some time been the underlying organizational principle used in most modeling environments including the UML. However, it has traditionally only been used (at least explicitly) to define and relate the models within a single dimension. In the original four-level modeling hierarchy there was no notion of different kinds of "instance of" relationships, and every new level of classifiers was viewed as being stacked on top of just one other level. There was no notion of levels existing within a single level as implied by the idea of ontological versus linguistic classification. As a result, the interpretation of the four-layer architecture in which user instances (i.e., user model elements representing individuals of the universe of discourse) reside at a level M_0, below users types (classifying individuals of the universe of discourse) at level M_1, and the latter below a level M_2, containing the UML language definition, contradicted the strict metamodeling tenet that all types must be directly at the next level above their instances. User objects such as "polly" at level M_0 need to be classified by "Object" ("Instance Specification" since UML 2.0) at level M_2. The corresponding "instance of" relationships crossed two levels, violating strictness and questioning the design and meaning of "levels".

However, once one realizes that the "two-level crossing" "instance of" relationships are linguistic "instance of" operations, an arrangement of levels that preserves strictness is easily achieved. Figure 36 shows how one can think of the top language definition layer as spanning all user modeling levels.

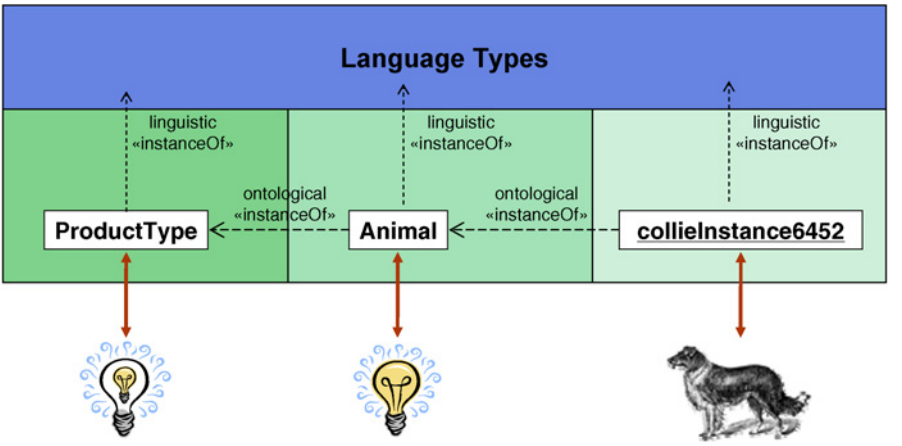


FIG. 36. Orthogonal classification architecture.

One may now argue that the "instance of" relationships within the bottom row of Fig. 36, the ones within the language usage level, break the rules of strictness because they do not cross level boundaries. However, being ontological "instance of" relationships, they form their own separate, orthogonal classification hierarchy. In this way, the notion of strictness is applicable to each (linguistic and ontological) classification hierarchy separately. Figure 36 therefore shows strict metamodeling applied to two dimensions. We refer to an environment like this that retains the strictness doctrine separately in orthogonal modeling dimensions as an *Orthogonal Classification Architecture*.

In principle the number of levels in either direction is unlimited. However, the number of levels usually needed in practice is small. As discussed in Section 7 above, in the ontological dimension the number of levels rarely rises above four. We will consider how many levels are useful in the linguistic level in the following section.

8. Languages versus Libraries

In this section we consider how the orthogonal classification architecture described above may support "language customization", the central theme of this chapter.

8.1 Class Libraries

In Section 6, where we introduced the distinction between ontological and linguistic classification, we observed that in general a model element can have one

(or more) superclass(es) as well as an ontological classifier, and that these impart properties to different facets of the model element. Up to now we have regarded the linguistic metamodel (i.e., the language definition) as defining the conceptual building blocks with which a user creates a model of the domain of interest, however, it is also possible to make concepts available to users in terms of predefined classes in the ontological hierarchy.

In fact this is not just a "possibility" it is actually an approach that has been used with great success in mainstream object-oriented environments such as Smalltalk and Java for some time. Most of the abstractions used by a Java programmer when developing a new application actually come from existing class "libraries" rather than from the language itself. The root of the class inheritance hierarchy in every Java program is the class "Object" defined in the core package. Not all of the properties of a Java class are therefore defined in the Java language definition; a number of them are defined in the class "Object" which every class inherits from. However, in order to take advantage of even more predefined functionality or structure, new Java classes are often defined as subclasses of more specialized classes. The standard Java environment consists of a large number of classes in a collection of different packages providing functionality ranging from I/O to database access and GUI's. In addition, there is a much larger resource of third party class libraries that provide predefined functionality for all imaginable purposes. Moreover, these class libraries can range from being highly general, such as those in the Java standard, to very domain-specific, such as libraries providing financial services or capabilities for different platforms. For examples, at the level of Java code, the EJB platform used in the case study is made available to users in the form of predefined classes not only to use in a client-server fashion but also to inherit from.

In object-oriented programming technology, therefore, libraries play a big role in making domain-customized functionality available in the development of new applications. For some reason, this approach has never been used in the definition of modeling environments. In particular, in the case of the UML, all the predefined modeling abstractions are defined within the metamodel, and there is no notion of predefined classes at the user model level that users can specialize when defining their own classes.

However, there is no good reason why this should not be done. On the contrary, there are many good reasons why it would be advantageous. Figure 37 shows for example, how a (new) class "Animal" can be defined by inheriting from a preexisting class in a predefined library of model elements. Some classes, like "Product", are specific to the retailing domain, whereas others, like "Object", are general purpose.

The only limitation of straightforward class libraries of the kind just introduced is that they can only influence the type facet of user model elements as explained in Section 6. This means that they can define properties that instances of the model

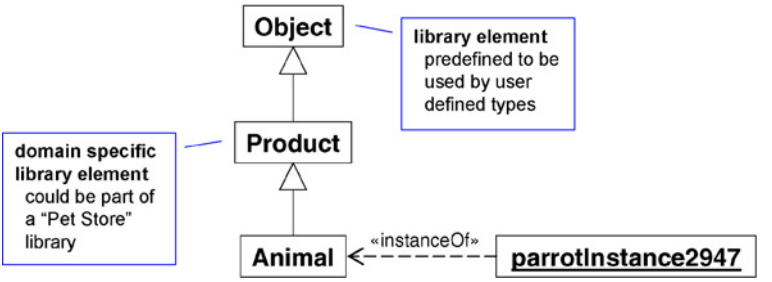


FIG. 37. Predefined library.

elements must have, but they can not specify properties (i.e., slots) for the model elements themselves. However, specifying such properties is easily achieved by introducing a predefined library of model elements at the ontological metalevel as well. In Fig. 38, the model element, "Animal", receives features from its supertypes ("Product" and "Object") as well as from its types ("ProductType" and "Class"). The former influence the type facet of "Animal" and the latter the instance facet (and possibly the type facet through the use of potency values higher than "one").

8.2 Library Customization

In the previous sections the issue of customizing a software engineering environment for a particular domain was always discussed in terms of language customization. However, as we have just seen, libraries can have as big an influence on the building blocks used to describe software applications as a language can. Thus, we also need to consider the role that library customization can play in the overall customization process.

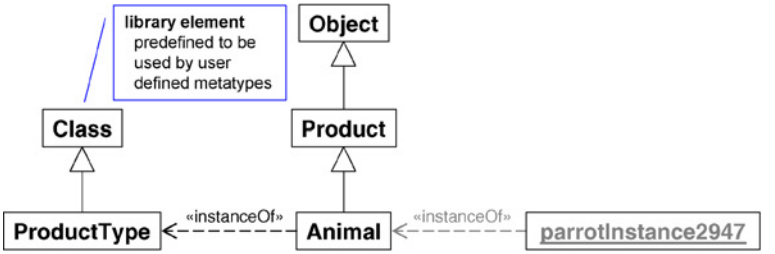


FIG. 38. Meta library example.

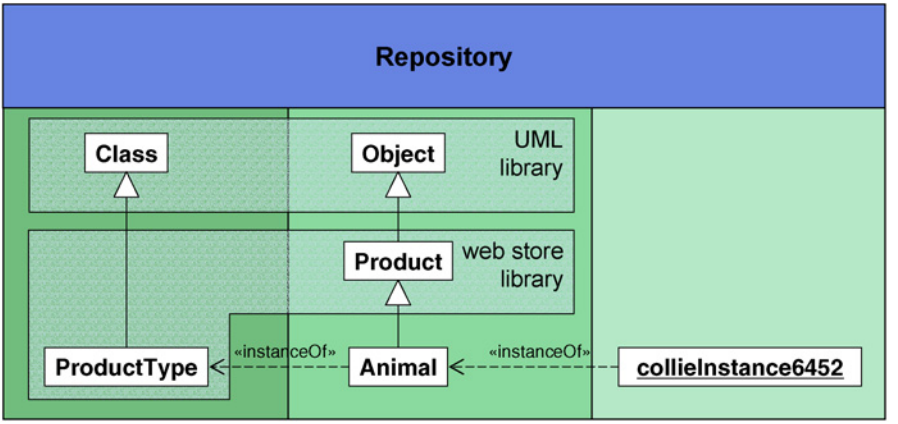


FIG. 39. Language versus library customization.

Figure 39 shows the "big picture" with the complete range of elements that provide the building blocks for, and influence the properties of, the elements modeled by the user (in this case "Animal" and "collieInstance6452"). The elements at the top language definition level (here called "repository", see below) represent the language, per se, while those in the ontological levels below represent the predefined elements at the language user level. Note that predefined elements may appear at any ontological level, including the bottom-most, for cases such as the Boolean values "true" and "false". As shown in Fig. 39, the predefined modeling elements can be organized into different packages (libraries) reflecting different levels of abstraction and domain-specificity. For example, the web store library contains elements that are specific to the domain of web stores.

In general, therefore the customization of a modeling environment based on an Orthogonal Classification Architecture can be done in one of two places: the language level (top) and the user level (bottom). Within the user level, customization may be achieved at any of the present levels. Traditionally, in the context of modeling languages the emphasis has been on language level customization. However, there are advantages to performing as much of the customization as possible at the user level in terms of the ontological levels. This, of course, requires the core concepts (such as "Class", "Object", etc.) to be available as library elements at ontological levels. As these concepts then are no longer needed in the language definition, the latter can be made very small, in fact, it becomes a minimal "repository format" that simply needs to provide a medium for all modeling content in the ontological user levels. As a result, the "language" can become very small and stable, whereas the core modeling concepts become customizable.

Note that if you turn the diagram of Fig. 39 by 90 degrees to the right, and relabel "Repository" with "MOF" then the hierarchy looks a lot like the original four-layer-architecture stack, except that the MOF is not used in its role as a meta-metalanguage at the top, but in its role as a level spanning repository format. However, in the architecture of Fig. 39 no elements in the second (leftmost) ontological level need to classify elements in the bottom (rightmost) ontological level, as was the case in the original four-layer architecture interpretation. There is also no level-spanning language definition, as in the latest four-layer architecture interpretation where M_2 spans both user types and instances at level M_1. The orthogonal classification architecture in combination with a "library metaphor" for predefining modeling concepts therefore completely abandons the idea of a (fixed) language in the linguistic sense and represents everything (flexibly) as user modeled libraries and models.

9. Transformations

In previous sections we have discussed how to best support the derivation of new languages from existing base languages, but only in terms of the definition of the abstract syntax of the new languages. This is justifiable since the abstract syntax forms the basis for the definition of all other language elements. However, it will not be economically viable to create domain-customized languages if the definition of their semantics is difficult and laborious. Since a practically useful semantics in a tool context must be a translational semantics, i.e., a semantics that is based on mapping a new language onto a target language with known semantics, the ease of creating a semantics boils down to the ease of creating a so called "exogenous" transformation [12]. A comprehensive discussion of the various existing transformation approaches is out of the scope of this chapter but we briefly want to illuminate the implications that the various alternatives to language definition have on the effort required to query the source model for the pieces of information required to perform transformations.

9.1 Model Access

For the purpose of discussing different ways of accessing model elements in order to collect relevant information for transformations it is instructive to think of a model database which is queried for elements in the model. In each of the above applications one wants to identify a precise subset of the full set of elements (no less and no more). In transformations, such a set typically represents all elements that are mapped to the target language in the same way, i.e., by applying the same sub-transformation.

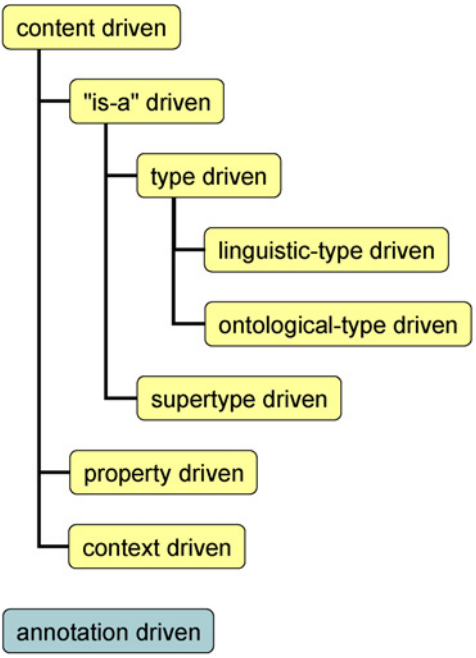


FIG. 40. Transformation taxonomy.

Figure 40 shows a taxonomy of various ways of characterizing such model element subsets.

Most alternatives fit under a common "content-driven" characterization since they use model element values to ascertain whether they belong to a particular subset or not. Note that even an element's type or supertype can be regarded as a value associated with this element. The taxonomy of Fig. 40 generalizes the exploitation of these last two values of an element to "is-a" driven since it is possible to phrase, e.g., "Animal is-a ProductType" and "an Animal is-a Product", and both are ways to reference a set of elements in relation to another element that shapes the elements in question in characteristic ways (through classification or generalization respectively).

In Fig. 40 "property-driven" refers to the ability to use values from the instance facet of elements, such as slot values in UML objects or tagged values in UML classes. This allows a subset of elements (e.g., those branded with a certain stereotype) to be further subsetted depending on whether or not a certain tagged value is present.

The category "context-driven" refers to the consideration of values of the elements which may be reached from them (excluding types and supertypes). This is useful for identifying source *patterns* which will be mapped to target patterns.

The one alternative that represents its own top-level branch is "annotation driven". Annotations, such as marks [11], do not belong to the elements themselves and should be kept independent from them since they will typically change as the target model or the transformation is changed. The underlying assumption here is that a given model will be transformed to many target models depending on choices made for implementation technologies, etc. Even if one manages to use the MDA metaphor of using annotations "thought of as being applied to a transparent layer placed over the model" [11], one will still have to update the bindings of the annotations to the model elements whenever the annotations change. Hence, although the "transparent layer" metaphor allows the source model to be unpolluted by target model details (in the form of annotations), the recurring need to change annotations and their binding to the source model vastly increases the cost of defining transformations. As a result, one should maximize "content driven" alternatives to defining transformations and avoid "annotation driven" ones. In the following, we discuss how the three different approaches to language design address this and other issues related to defining transformations.

9.2 DSL/DSM Paradigm

Domain-specific languages support "type driven" element selections well, since elements are typically classified with a rich domain vocabulary such as "Button, ShortCutButton, MenuButton", etc. Even if the domain does not stipulate a very fine grained classification it is easy to retrofit it into the domain-specific language in order to support fine-grained guidance for transformations. In contrast, in a classic UML model, classes such as "ShoppingCart" and "Video" would just be known as instances of M_2-level element "Class" and typically annotations (in the form of stereotypes or marks) would be used in order to guide the generation tool to e.g., produce a EJB session bean in the one and an EJB entity bean in the other case. In a DSL approach one already has different linguistic classifiers for "ShoppingCart" and "Video" respectively and thus a way to guide the transformation accordingly.

A drawback of the DSL approach regarding transformations is that it makes transformations difficult to reuse. Since any domain-specific language is allowed—and in fact, encouraged—to use a complete new set of modeling concepts, transformations will have to be adapted for the new vocabulary even if the differences are just in the names. This is further testimony to the fact that domain-customized languages should be preferred over orphan domain-specific languages.

9.3 OMG Architecture

The use of a standard modeling language such as UML addresses the above mentioned problem regarding the reuse of transformations by fixing a standardized metamodel. Hence, transformations may use a "linguistic-type driven" approach (see Fig. 40) independently of the current domain the user is working in.

This, however, creates another complication which results from the fact that transformations will typically have to consider both the linguistic and the ontological type for determining the correct target element type. For instance, one may want to guide the transformation of object "Animal" by considering it as an instance of M_2 element "Class" but also exploit its ontological type, e.g. element "ProductType". Transformations therefore need to consider both the linguistic and ontological type of an element in order to determine the correct target element.

This example highlights another problem of the OMG approach to defining transformations. It cannot be assumed that useful information to guide transformations will be expressed with a certain mechanism. As we have seen in previous sections, classes may be classified through powertypes or stereotypes, making it difficult to keep transformations simple and reuse them. Moreover, the way to access information, such as type information changes with the level one is assuming. For example, objects have yet another different way of specifying their type. Access to the instance facet of objects (through slots) is different to that of classes (through tagged values). These are symptoms of lost opportunities to unify the ontological levels, ultimately causing unnecessary complications for transformations which cannot be expressed in a level-independent manner.

Furthermore, since ontological levels only implicitly exist within M_1, often opportunities for a target independent way of annotating classes are lost. Instead of using stereotypes to assign natural metatypes, such as "ActivityType" or "ProductType", they are typically used to directly indicate the generation of "EJB Session" or "EJB Entity". In fact, the MDA guide [11] even assumes that marks will be used to directly indicate target elements. However, if this approach is followed, marks will have to change each time the target model changes, for instance, due to a change in target platform choice. We have already pointed out that this is to be avoided. It is much better to use domain-specific and target-independent types and, if necessary, keep information about which types need to be mapped to which target element types in an external dictionary.

Finally, although there is no immediate technical reason for it, UML users assume rich support for guiding transformations through stereotypes and little or no support for guiding them through supertypes. This may lead to curious attempts to *classify* class "Animal" as a "Product" (by using a stereotype) where a subtype relationship would have been correct as explained in Section 4.3.2.

9.4 Library Approach

The infrastructure approach presented in Sections 7 and 8 and its associated way of hosting predefined concepts (presented as the "library approach") implies an approach to defining transformations that rectifies the problems we have just identified with the other approaches.

Assigning different transformations strategies to different model elements is easy, even though one will often not have the benefit of having a fine-grained linguistic classification as with DSL/DSM approaches. However, when faced with transforming "ShoppingCart" and "Animal" in different ways, for example, one just introduces the ontological metatypes "ActivityType" and "ProductType" respectively. This obviates the need to resort to the undesirable use of annotations.

On the other hand, transformations can always rely on the existence of fixed upper-ontologies (i.e., predefined libraries). Transformations may make use of more specific (meta-)types (such as "ProductType") but they can always rely on the existence of general ones (such as "Class"). This is not only beneficial for creating reusable transformations but also means that it is not required to introduce domain-specific (meta-)types unless one finds a need for them.

In order to guide transformations there is, therefore, no need to consider both linguistic and ontological dimensions. Transformations using models based on the library approach never need to consult linguistic types, as these are in any case too general to provide any useful information. However, this does not imply a loss of information. For instance, model element "Animal" is characterized as both "Product" and "Object" in the ontological generalization hierarchy, since "Object" is a supertype of "Product". This makes it easy to refine and thus also reuse existing mappings. A mapping that only exploited the fact that some elements are subtypes of "Object" can be refined to one that exploits the fact that some elements are subtypes of "Product" without implying a shift from linguistic classification to ontological classification. Such a seamless way of adapting transformations is of paramount importance for providing a cost effective way of defining semantics.

Furthermore, the uniform representation of elements in the ontological domain makes it easy to access information in a level independent manner. For instance, product types with a tax rate of 7% may be transformed into different target elements than those with a taxe rate of 16%. Accessing these values, however, is no different to accessing the price of animal instances, for example.

Finally, the library approach assigns equal weight to the type- and supertype properties of an element and makes deliberate use of supertype relationships for defining the superstructure itself. The usefulness of using "supertype driven" model element selection is indeed indirectly acknowledged in the MDA guide by a "Model Transformation" example [11, Section 3.10.3].

10. Conclusion

In this chapter we have taken a grand tour of the underlying principles and the practical technological issues involved in supporting a domain-customized language approach to software engineering. We started by outlining the conceptual differences between the notions of abstraction and domain-specificity and showed that they conceptually map out a two dimensional language space. We also explained how the notion of *domain-specific* languages relates to the notion of *domain-customized* languages in that the latter are a special form of the former which are explicitly derived from an existing base language. With the notion of an *essentially-customized language* we have furthermore characterized the optimal compromise regarding the diverging goals of "wide intelligibility" (as achieved by modeling standards) and of "desiring the best possible fit regarding an application domain" (as achieved by DSLs). An essentially-domain-customized language is as domain-specific as necessary and as compatible to its base language as possible.

After laying out these basic principles and showing how we used them to develop models for our running case study—a Pet Store e-commerce system—we went on to outline the range of theoretically possible language derivation types and explained their pros and cons for the end user. We then described in detail the particular lightweight language customization mechanism offered by the UML. Known as the "profile" mechanism, this allows the effects of metamodel specialization to be achieved without actually making any changes directly to the metamodel. The section that followed discussed the range of different architectures that can be used in the construction of DCL support tools, ranging from simple "two-level" environments through multi-tool cascading environments to multi-level, flattened architectures in which one or more modeling levels are flattened into a single modeling level.

We then explored the distinction between *ontological metamodeling* and *linguistic metamodeling* and argued that both forms are valuable, despite the fact that ontological metamodeling has been neglected by current mainstream modeling approaches. We then introduced the notion of having multiple ontological modeling levels within a single linguistic level and showed how this naturally led to the notion of the Orthogonal Classification Architecture as a clean way of supporting ontological multi-level modeling. The next section built on this by explaining how customization can be characterized in terms of library specialization as well as language specialization, and explained why it makes sense to realize most of the customization capabilities needed by end users in the former way rather than the latter way. Finally, Section 9 finished with a discussion of how all the various notions discussed before relate to and support the definition of transformations, an essential element of the DCL paradigm.

As the universally accepted general purpose modeling language the UML offers the ideal foundation for the realization of a DCL paradigm of the kind explained in this chapter. However, as pointed out in several sections, there are several places in which the UML language and the UML infrastructure need to be improved. First, the UML infrastructure should be founded on the Orthogonal Classification Architecture and should define most of the predefined modeling constructs in the form of predefined libraries rather than as part of the linguistic metamodel. This will allow a large proportion of *language customization* to be achieved through the much simpler mechanism of *library customization*. Second, the language needs to provide clean notational support for the concepts of clabjects and fields and the related mechanism of potency.

We hope that the ideas and suggestions described in this chapter will help the reader derive more benefits from the DCL technologies which are going to become an importance part of software engineering in the near future.

REFERENCES

[1] Angermann A., Beuschel M., Rau M., Wohlfarth U., *Matlab–Simulink–Stateflow, Grundlagen, Toolboxen, Beispiele*, Oldenbourg Verlag, München, ISBN 3-486-57719-0, 2005.
[2] Atkinson C., Kühne T., Henderson-Sellers B., "Systematic Stereotype Usage", *J. Software and Systems Modeling* **2** (3) (2003) 153–163.
[3] Atkinson C., Kühne T., "Profiles in a Strict Metamodeling Framework", in: Evans A., Kent S., Selic B. (Eds.), *J. Science of Computer Programming* **44** (1) (2001) 5–22.
[4] Boulanger R., *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming*, MIT Press, 2000.
[5] Cook S., Kleppe A., Mitchell T., "Defining UML family members using prefaces", in: *Proc. of TOOLS Pacific*, 1999.
[6] Coad P., "Object-oriented patterns", *Commun. ACM.* **35** (9) (1992) 152–159.
[7] Engels G., Förster A., Heckel R., Thöne S., "Process modeling using UML", in: *Process-Aware Information Systems*, Wiley Publishing, New York, 2005, pp. 85–117.
[8] Johnson R., Woolf B., "Type object", in: *Pattern Languages of Program Design*, vol. 3, Addison–Wesley, 1997, pp. 47–66.
[9] Kühne T., "Matters of (meta-)modeling", *J. Software and Systems Modeling* **5** (4) (2006) 369–385.
[10] Liskov B., Wing J., "Family values: A behavioural notion of subtyping", MIT/LCS/TR-562b, 1993.
[11] OMG, "MDA Guide Version 1.0.1", OMG document omg/03-06-01, 2003.
[12] Mens T., Van Gorp P., "A taxonomy of model transformation", *Electr. Notes Theor. Comput. Sci.* **152** (2006) 125–142.
[13] Object Management Group, "Meta-object facility 2.0 specification", ptc/03-10-04, April 2002.

[14] Odell J., "Power types", *J. Object-Oriented Programming* **7** (2) (May 1994) 8–12.
[15] Parker B., "Introducing EIA-CDIF: the CASE data interchange format standard", in: *Proc. of the Second Symposium on Assessment of Quality Software Development Tools, 27–29 May*, 1992, pp. 74–82.
[16] Object Management Group, "UML 2.0 Infrastructure Specification", formal/05-07-05, http://www.omg.org, 2005.
[17] Object Management Group, "UML 2.1 Superstructure Specification", ptc/06-04-02, http://www.omg.org, 2006.
[18] Warmers J., Kleppe A., *The Object Constraint Language*, Addison–Wesley, ISBN 0-321-17936-6, 2003.

This page intentionally left blank

Advances in Business Transformation Technologies

JUHNYOUNG LEE

IBM T. J. Watson Research Center
Hawthorne, NY 10532
USA
jyl@us.ibm.com

Abstract

The backbone of the World economy has shifted from agriculture to manufacturing to information. It is now entering a new phase known as an *innovation-based economy* where economic value will be created in *services* we provide with information to improve business, government, education, and people's daily workspace. Accordingly, the focus of computing and information technologies is shifting to their applications to help enterprises, governments, and other organizations improve and transform their current practices. This chapter will describe the latest advances in the development of new methods and technologies supporting the service-led economy. First, it will present a model-driven business transformation approach. Business transformation is a key executive management initiative that attempts to align the technology initiatives of an organization closely with its business strategy and vision for a better business performance. The presented approach employs models that map IT (Information Technology) functions and capabilities to business performance, and help articulating the delivered business value of IT solutions and services. Also, it integrates a value model with business models of processes and components to leverage the value model at different levels and phases of business transformation. One of the key requirements for the model-driven business transformation is the capability of representing the semantics of various aspects of the models in a language and enforcing them. Also, efficient engineering of the semantic models that works well with the traditional software engineering mechanisms has become more and more important in business or enterprise IT application development. The second part of this chapter will address these requirements and discuss a novel approach to engineering semantic models, which allows seamlessly supporting existing software engineering models in Unified Modeling Language or other modeling languages in semantic model-based enterprise application development. Finally,

ADVANCES IN COMPUTERS, VOL. 70
ISSN: 0065-2458/DOI: 10.1016/S0065-2458(06)70004-3

this chapter will describe a new approach to business process integration by using Web services. While the Web service technologies facilitate the creation of business process solutions in an efficient, standard way, it is required to automate their discovery and composition to make it useful and scalable. We will present a solution to these problems of the Web service-based business process integration: the discovery of Web services based on the capabilities and properties of published services, and the composition of business processes based on the business requirements of submitted requests.

1. Introduction

The backbone of the World economy has shifted from agriculture to manufacturing to information. It is now entering a new phase known as an *innovation-based economy* [5,39] where economic value will be created in *services* we provide with information to improve business, government, education, and people's daily workspace. Accordingly, the focus of computing and IT (Information Technology) is shifting to the application of technologies to help enterprises, governments, and other organizations improve and transform their current practices. To facilitate the business transformation process, the service-led economy requires the development of new business methods and the technology supporting those methods. Industry and academia, to cope with this paradigm shift in the role of technology, forms a new discipline called *service science, management, and engineering* [19] by converging ongoing work in related fields of computer science, industrial engineering, operations research, management sciences, and social and legal sciences. Services science would merge technology with an understanding of business processes and organization. It would transform business by recognizing an organization's pain points and apply technologies to correct them.

1.1 Model-Driven Business Transformation

Among the emerging methods and the supporting technology for business transformation in the service-led economy is the *model-driven business transformation*, which utilizes a multi-layer model approach to linking business and IT semantics [20, 30]. The upper layers of the model represent business semantics in the terms familiar to business executives, business managers and analysts such as business processes, activities, Key Performance Indicators (KPIs), operational metrics, value drives, and governance. The lower layers of the model represent IT architecture comprising a wide range of services implemented in IT infrastructure such as service-oriented architecture. The vision of this multi-layer model is to enable IT solutions to accurately reflect and be driven by business intent. Figure 1 illustrates the multi-layer model approach to business transformation.

The key to this multi-layer model is that the layers are linked in meaningful ways, so changes in one layer can ripple through other layers. The representation and enforcement of the semantics of the different layers and also of the connections between the layers is essential to the model-driven approach and also is an application area of the semantic Web technology. This model-driven approach provides a convergence of the business and IT models using a multi-layer model, which tightly couples the business and IT models. In many ways, this vision is not new. Technologists have been working towards generalized business process integration and

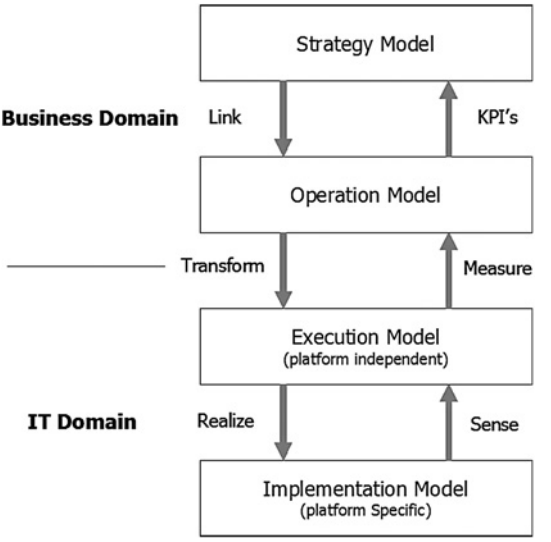


FIG. 1. Model-driven business transformation.

automation for many years. However, this approach is different from the traditional technology-oriented business integration, because it provides a top–down business perspective which enforces a business-orientation of business transformation.

Once equipped with end-to-end tools for the model design, connection and transformation, this approach has the potential to reduce the time-to-value of business solution implementation. It would replace the manual creation of unstructured business documents and informal business models with a guided transformation of a structured multi-layer model. The IT solutions generated by this approach would accurately and precisely reflect the original business semantics and are directly deployable and executable in a service-oriented architecture. This model-driven business transformation approach is a significant step towards closing the infamous "business-IT gap" [35] achieving maintainable alignment between business design and IT solutions.

Recent trends in componentization and modeling of IT and business would boost this model-driven approach as a prominent methodology for the service-led economy. In recent years, enterprises componentize into discrete services to achieve operational efficiency, flexibility, and to sharpen their focus. Also, the consulting industry increasingly utilizes sophisticated modeling techniques to understand and transform businesses. In the IT domain, software modeling technologies and methodologies such as the Object Management Group's *Unified Modeling Language* [41] and

Model-Driven Architecture [4,40] are widely adopted and studied in both industry and academia. In addition, the *Web service* initiative of the World Wide Consortium (W3C) [57] and related technologies accelerate the shift towards *Service-Oriented Architectures* [58] which fit the model-driven business transformation approach. The trends in componentization and modeling of business and software effectively converge to provide new layers of business understanding and responsiveness.

1.2 Semantic Web

Traditionally, a model has been used to represent things in various contexts including studies of physics, mathematics, statistics, economics, geology, psychology, computer science, to name a few. A model often dominates the understanding and solution to the given problem in the domain. Additionally, the language used to specify a model often impacts on (either assists or limits) the thinking process with the model. The most important component of the model-driven business transformation approach is the model, i.e., the representation of the semantics of business and IT resources. With the multiple layers in the model, another key component is the representation of the meaning of the links across different layers. It is crucial to this model-driven approach how we represent in a language and enforce the semantics of the layers and also of their links.

Semantic Web [2,55], another W3C initiative, which intends to create a universal medium for information exchange by giving semantics, in a manner understandable by machines, to the content of resources, provides an appropriate option to address this modeling requirement of the model-driven approach. The Semantic Web is comprised of the standards and tools of markup languages including Web Ontology Language (OWL) [56] and Resource description Framework (RDF) [3,54]. These languages based on Extensible Markup Language (XML) would be used to specify ontological representation of models including the business and IT models and their connections. An *ontology* or a *semantic model* is similar to a dictionary, taxonomy or glossary, but with structure and formalism that enables computers to process its content. It consists of a set of concepts, axioms, and relationships, and represents an area of knowledge. Unlike taxonomy or glossary, a semantic model allows modeling arbitrary relationships among concepts, representing logical properties and semantics of the relationships (e.g., symmetricity, transitivity and inverse), and logically reasoning and querying about the relationships.

The semantic markup languages would be used to specify the convergence of business and IT models, and more importantly, their metamodels. The ontological representation of the metamodel of a constituent model enables reasoning about the instance model, which enables a causality analysis to deduce unknown or implied relationships among entities within the instance model. The analysis would

be extended across multiple layers of models. The semantic model-based causality analysis would reveal which entity has an impact on which entities (e.g., business components and processes, performance indicators, IT systems, software classes and objects, etc.) of the multiple layers of the model. This semantic model-based analysis would be applied to a model that provides an introspective view of the business within an enterprise. Also, it would be applied to a value network which yields an extrospective view of businesses in an ecosystem.

In addition to its use in the model-driven business transformation, the semantic model approach is also useful in business information and process integration. Suppose a business solution requires integrating a number of data sources (or application interfaces for process integration) which provide different but overlapping conceptual models. An approach to integrating them would be using a global semantic model which essentially maps the data sources based on their meaning. The data sources are defined as views into this global model, although there is no guarantee of completeness. A query to the data sources would be expressed in the global semantic model. The result set for the query would be constructed by finding all conjunctive queries over the views that are contained in the top-layer query. A semantic model-based approach to process integration would require a similar set of steps over a set of overlapping application interfaces.

The model-driven business transformation approach proposes a new business method and the supporting technology by coupling business and IT models. It provides a top-down business perspective which enforces a business-orientated business transformation. It has the potential to provide a number of benefits over the traditional technology-oriented approach, including business-IT alignment, reasoning about business design and transformation, real-time visibility into business operation, improved business performance management, rapid and repeatable IT solution implementation, and adaptive IT solution implementation. The key to this model-driven approach is that the layers are linked in meaningful ways, and that the semantics of the links are effectively represented and reasoned. Therefore, changes in one layer can accurately ripple through other layers. The semantic Web technology is an enabler to fulfill the modeling requirements in representation and enforcement of the semantics of the multi-layered model. It poses a key enabling technology for the emerging service science, which will meld technology with an understanding of business processes and organization.

1.3 Synopsis

This chapter first will present a value-oriented, model-driven approach to business transformation that integrates the value model with business models of processes

and components to support multiple phases of value-oriented business transformation. The main idea of this solution is to enable business executives to take better transformation decisions by building a unique model for a business that captures all entities involved in creating value for the business, and defining qualitative and quantitative business analyses that could be performed on the business model. To show how we can realize this idea, we will introduce the *VIOLA* system developed at IBM Watson Research Center.

Then, we will discuss a novel approach to engineering ontologies and semantic models, which allows seamlessly supporting existing software engineering models in UML (or other modeling languages) in semantic model-based enterprise application development. One of the key requirements for the model-driven business transformation is the capability of representing the semantics of various aspects of the models in a language and enforcing them. Also, efficient engineering of the semantic models that works well with the traditional software engineering mechanisms has become more and more important in business or enterprise IT application development.

Finally, we will describe an approach to business process integration by using Web services. Web service technologies facilitate the creation of business process solutions in an efficient, standard way. However, it is required to automate their discovery and composition to make the approach useful and scalable. This chapter will present a solution to these problems of the Web service-based business process integration: the discovery of Web services based on the capabilities and properties of published services, and the composition of business processes based on the business requirements of submitted requests. The solution is comprised of multiple matching algorithms, a *micro-level matching* algorithm, which matches the capabilities of services with activities in a process request, and *macro-level matching* algorithms, which are used to compose a business process by identifying services that satisfy the business requirements and constraints of the request.

2. Value-Oriented, Model-Driven Business Transformation

Business transformation is a key executive management initiative that attempts to align the technology initiatives of a company closely with its business strategy and vision, and is achieved through efforts from both the business and IT sides of the company. However, the technology side of the company often emphasizes functions and capabilities, while the business side focuses on business impact and value. Because of this "business-IT gap" [35], business transformation processes for mid- to large IT and services are long and costly. In addition, failures to demonstrate the busi-

ness value of the transformation processes often hinder the intended transformation or cause an inconsistent and unmanaged investment portfolio.

In order to address this problem, this section presents a novel approach that maps IT functions and capabilities to business performance, and demonstrates the delivered value of IT investments. This approach integrates the value model with business models of processes and components to leverage the value model at different levels and phases of business transformation. It innovatively extends the *Model-Driven Business Transformation* [20,30] by linking values with key IT enablers all the way down to the IT infrastructure, combines the *Component Business Modeling* [21] with the value-oriented business analysis to strengthen its business analysis capabilities, and utilizes the measurement provided by the *value model* to monitor and track value and improvements during and after business transformation.

Component Business Modeling (CBM) is a business decomposition methodology and a business modeling technique increasingly utilized by the consulting industry to understand and transform businesses. CBM models a business as a set of business components. A business component is a part of an enterprise that has the potential to operate independently, in the extreme case as a separate company, or as part of another company. A business component is a logical view of part of an enterprise that includes the resources, people, technology and know-how necessary to deliver some value. A component business map is a tabular view of the business components in the scope of interest. The columns of the table represent business competencies and the rows represent accountability levels. The business components are rectangles within the table. Normally each rectangle is within only one cell of the table. Figure 2 shows a sample component business model.

A component business model represents the entire business in a simple framework that fits on a single page. It is an evolution of traditional views of a business, such as ones through business units, functions, geography, processes or workflow. The component business model methodology helps identify basic building blocks of business, where each building block includes the people, processes and technology needed by this component to act as a standalone entity and deliver value to the organization. This single page perspective provides a view of the business which is not constricted by barriers that could potentially hamper the ability to make meaningful business transformation. The component business model facilitates to identify which components of the business create differentiation and value. It also helps identify where the business has capability gaps that need to be addressed, as well as opportunities to improve efficiency and lower costs across the entire enterprise.

The *value analysis* is another recent trend in today's business environment where enterprises are increasingly focusing on value rather than on the functions and capabilities of IT. They expect service providers to demonstrate value of technologies throughout business transformation phases. An approach to meeting this requirement

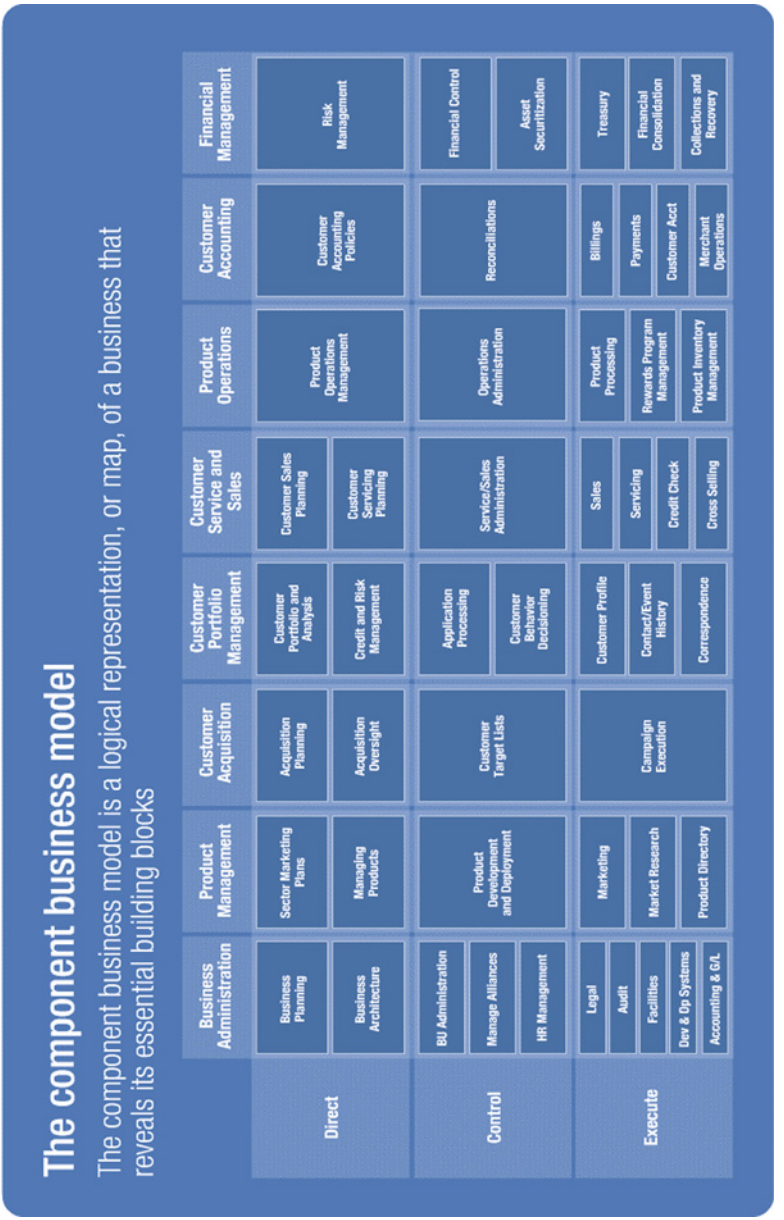


FIG. 2. Component business model.

of business transformation is *value modeling*, which identifies and maps the enterprise's key business and IT value drivers, and links them to the measurable business and financial benefits. The value model can also help tracking the performance and showing realized value during and after the implementation.

This section presents a value-oriented, model-driven approach to business transformation that integrates the value model with business models of processes and components to support multiple phases of value-oriented business transformation. The main idea of this solution is to enable business executives to take better transformation decisions by building a unique model for a business that captures all entities involved in creating value for the business, and defining qualitative and quantitative business analyses that could be performed on the business model. This framework developed at IBM Watson Research Center is referred to as the VIOLA system. In the following sub-sections, we will discuss the VIOLA model of businesses in detail, the qualitative business analysis capabilities VIOLA provides, the quantitative analyses, i.e., value-oriented analyses of VIOLA. Finally, we will discuss the technical details for the design and implementation.

2.1 Value-Oriented Business Model

Figure 3 gives a high-level view of the VIOLA model for enterprises. The model is designed to capture business entities that are involved in creating or defining value and their relationships. The business entities in the model include business components, business processes and activities, operational metrics, Key Performance Indicators, and value drivers. Operational metrics are used to measure the performance of business at the activity and process levels. Key Performance Indicators (KPIs) are higher level business metrics often associated with a number of operational metrics. Value and cost drivers are factors that influence the business performance, i.e., cost and value. They are linked with operational metrics and KPIs. In addition, the model represents their relationships to resources, services, messages, IT infrastructure, and solutions. Often, *solutions* refer to both IT and business capabilities to support business objectives and strategies, or address pain points of enterprises.

To allow the user to explore this rich information captured by this model, the VIOLA system provides multiple views into the model referred to as *business maps*. Each business map shows various entities involved in running and understanding of business and their relationships. The business maps provide visual models which organize the above-mentioned business entities in a structured way. In addition, they provide user interfaces which allows to the user interactively navigate and explore the information space for an analysis purpose.

Figure 4 shows a screenshot of a business map from the VIOLA system that contains the component business map, the value driver tree and the business activities.

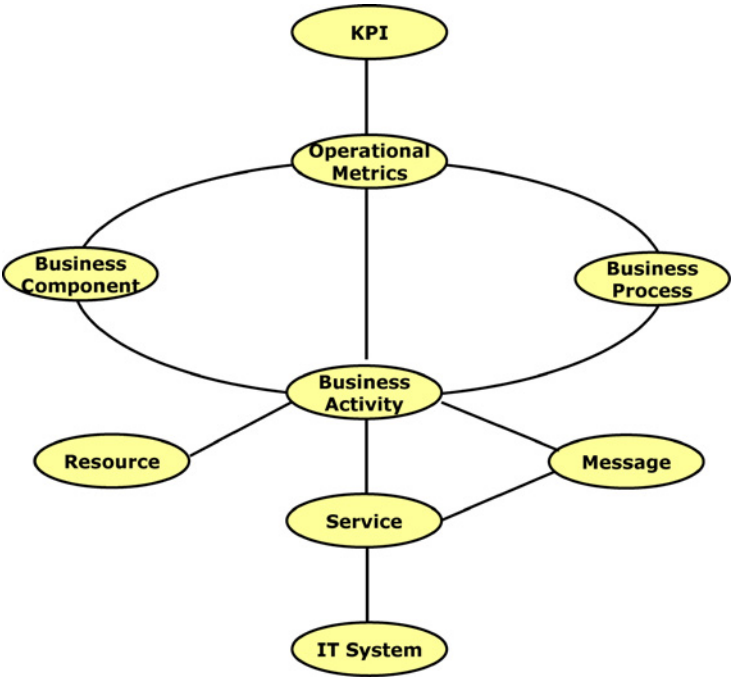


FIG. 3. VIOLA model for businesses.

To build such a business map, we utilize industry standard taxonomies of business processes and metrics such as APQC (American Productivity & Quality Center, Inc.) Process Classification Framework [1], and add the links of the processes and metrics to value drivers and business components. Additionally, the system allows the user to customize the industry standards to the needs of a specific enterprise, and import and export the enterprise-specific value driver trees.

2.2 Qualitative Business Analyses

The main advantage of the VIOLA model is the enablement of various types of analyses that would allow the user to obtain interesting insights into the current state of a business and its possible business transformation opportunities. The VIOLA system is designed on two primary analysis capabilities: the qualitative business analysis based on component business modeling and the quantitative analysis based on value modeling. By linking them together, VIOLA provides an end-to-end suite of business analysis capabilities, enabling business-driven, value-oriented business transforma-

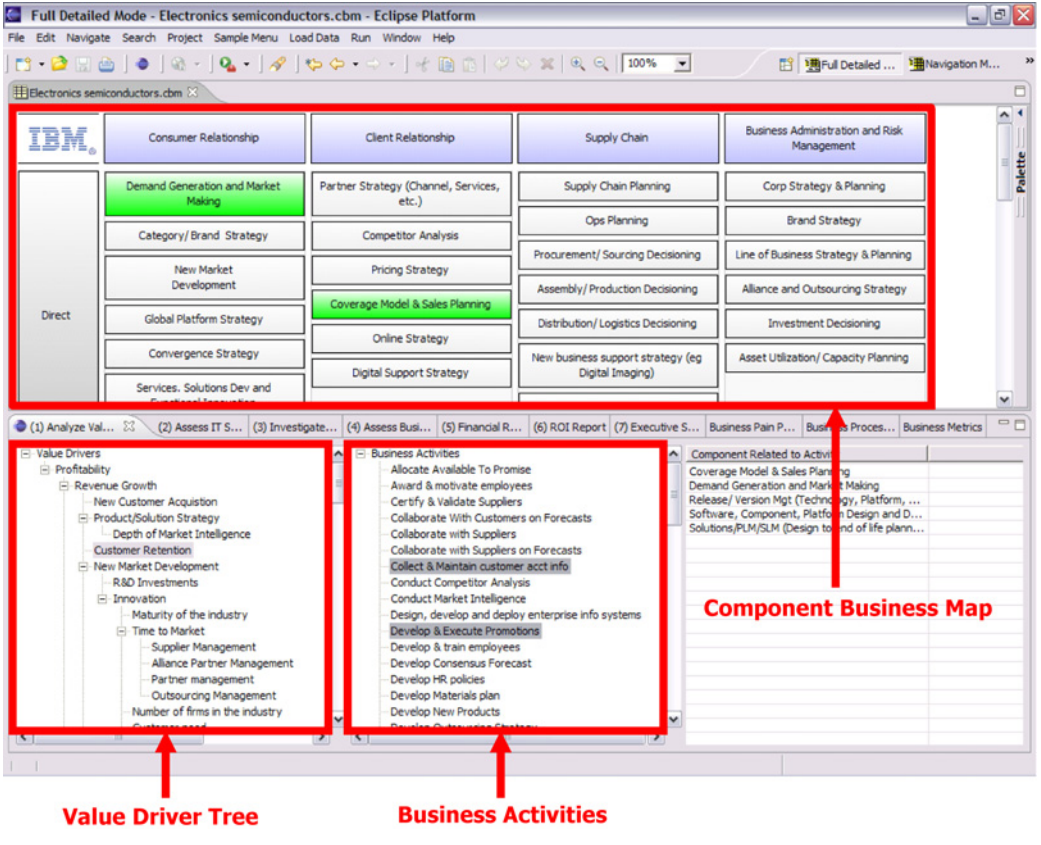


FIG. 4. Business map views.

tion. This section describes a few qualitative analysis capabilities of VIOLA, while the next section focuses on the quantitative analysis.

2.2.1 Dependency Analysis

The dependency analysis allows the user to explore the business maps and understand the correlations and dependencies among business entities. For examples, this capability can interactively identify one or more business components associated with a particular value driver. Conversely, it can find one or more value drivers that are affected by the performance of a particular business component. The associations between value drivers and business components are discovered through their relationships with business processes and activities. Similarly, VIOLA can identify and show dependencies between business activities and IT applications, and also between business activities and solutions, both IT and business-driven. Furthermore, the relationships are transitive, and so it is possible to infer the associations between value drivers and IT applications/solutions, also between components and IT applications/solutions, and so on.

To support the dependency analysis, the VIOLA system captures the basic relationship information in the VIOLA business model. Once the explicit relationship data are populated in the database using the model, the system utilizes a Semantic Query Engine to infer implicit relationships among various business entities by using the explicit relationships and their logical properties. To provide the inference capability, VIOLA utilized W3C's OWL semantic Web markup language [56] and the SnoBase Ontology Management system [31] developed at IBM Watson Research Center.

SnoBase system programmatically supports ontology needs of applications in a similar way a database management system supports data needs of applications. For programmers, SnoBase provides a Java API (Application Programming Interface) referred to as Java Ontology Base Connector (JOBC), which is the ontological equivalent of Java Data Base Connector (JDBC). JOBC provides a simple-to-use but powerful mechanism for application programmers to utilize ontologies without dealing with the details of ontological information. In addition, SnoBase supports a number of query languages including a variant of OWL-QL (OWL Query Language) and RDQL (Query Language for RDF) as ontological equivalents of SQL of relational database systems.

2.2.2 Heat Map Analysis

This analysis is an essential capability of CBM where the user discovers one or more "hot" components that are associated with one or more business strategies

and/or pain points. In the traditional CBM analysis, this step was conducted manually by the analyst depending on his/her knowledge and expertise in the business domain. VIOLA automated the capability by taking values into account with the dependency analysis.

First, the system allows the user to explore the value driver tree to identify one or more value drivers that may be associated with a certain business strategy/pain point. The discovery of "hot" components that affect the business strategy can be accomplished by executing a simple semantic query to the business model represented in OWL. Then the system colors the identified hot components differently to distinguish ones that affect positively or negatively to the strategy. The VIOLA system compares the industry benchmark and the as-is value of the operational metrics and performance indicators associated with the components to decide on their color. Figure 4 displays a heat map showing a couple of hot components affecting *positively* to a value driver, "Customer Retention," which is highlighted in the value driver tree.

2.2.3 Shortfall Assessment

The Shortfall Assessment allows the user to map the existing IT infrastructure against the "hot" components identified in the heat map analysis. It helps understand how the current IT infrastructure, such as applications and network capabilities, supports the business, especially, for those hot components. The analysis requires collecting the information on the current IT infrastructure and representing it in a semantic business model in OWL. Then the mapping of IT applications and capabilities to the components becomes, again, an execution of a simple semantic query to the semantic model.

VIOLA visualizes the mapping on the CBM map by overlaying IT applications on components. Then, the user can visually classify possible IT shortfalls into several types. Typically, four types of opportunities tend to arise. First, a gap indicates that a hot component does not have any IT support. The enterprise may want to consider an IT investment to improve the component's performance and support the intended business transformation. Second, a duplication indicates that a component is supported by multiple IT applications, possibly, deployed over time. The business may want to consolidate the applications to improve performance and reduce cost in communication and maintenance overhead. Third, a deficiency indicates that the current application lacks key functionality, or is poorly designed, and so incurs a project opportunity. Finally, an over-extension indicates that a system designed to support one business component is extended beyond its core capability to support others. Different definitions for the shortfall types may apply. With precise definitions of the shortfall types, the VIOLA system also automates the shortfall classification and recommends to the user the initially identified shortfalls.

It is important to note that an IT system can be involved with multiple situations. The value model of the VIOLA system takes that fact into account, with an optimized plan for implementation projects to maximize the investment. An integrated management approach such as project portfolio management ensures that the project opportunities are effectively taken into account, that the best use is made of available resources by applying them to the highest priority opportunities, that the projects are regularly assessed, and that management actions are taken to keep them aligned with objectives.

2.2.4 Solution Identification

Once IT shortfalls are identified and classified, one or more solution catalogs which provide information on various IT and business solutions to address the shortfalls and support the intended business transformation. VIOLA allows the user to explore the solution space to identify one or more solutions that may address one or more shortfalls of interest. The discovery of solutions for supporting components associated with a shortfall can be automatically conducted by executing a semantic query that correlates solutions and components by using their relationships to business activities. In addition, VIOLA allows the user to manually correlate them, if desired. If there is no prefabricated solution available from existing solution catalogs to support a certain hot component and/or an IT shortfall, the VIOLA system helps the user start composing a new solution, by providing a link to a solution composer tool, such as IBM's WebSphere Business Modeler [22], which utilizes and supports service-oriented architecture [58].

2.3 Value-Oriented Business Analyses

Until now, we described the qualitative business analyses of VIOLA, focusing on the identification of dependencies among business entities such as business components, processes and activities, value drivers, IT applications and solutions. This section highlights the quantitative business analysis capabilities.

2.3.1 Solution Value Estimation

This module allows the user to calculate the expected value of value drivers when one or more solutions are implemented in the context of business transformation. The details of the value modeling supporting this quantitative analysis will be given below. The calculation uses as input the as-is value of value drivers and the contributing factors of solutions to metrics that are associated with leaf nodes in the value driver tree. The expected values are calculated for a subset of the value driver tree, contain-

ing all the leaf node value drivers that are directly affected by the solution(s) and all their ancestors that are indirectly affected.

Once specific improvement opportunities for the target performance indicator are identified, they can be prioritized, based on a value-model analysis of each opportunity. The value model takes into account factors such as implementation cost, potential savings, increased revenue, reduced risk, and other financial metrics such as Return On Investment (ROI) and Net Present Value (NPV), Net Profit Margin (NPM) and Asset Turnover Ratio (ATR), and also improved Key Performance Indicators (KPIs) such as customer satisfaction, time for fulfillment, productivity and product quality. Based on this value and the risk assessment models, the opportunities can be quantitatively understood in terms of measurable value.

An in-depth value analysis of individual opportunities can show the detailed benefits of the IT project in terms of measurable value. For example, a duplication situation provides an opportunity for a consolidation project. The value analysis of this solution offers details of the project, including cost savings by shutting down multiple, inadequate systems, the investment required for implementing a single consolidated system, a comparative analysis of implementing a new system versus integrating and improving existing systems, desirable financial metrics such as ROI, reduced risk, and improved KPIs such as time for fulfillment and increased productivity. This analysis allows the enterprise to make services/solution decisions based on the values, costs and priorities for business transformation.

2.3.2 Value Modeling

The value model of the VIOLA system models both tangible returns such as cost savings and intangible benefits such as productivity enhancement, while most existing ROI analysis tools focus on direct benefits. This capability is important because direct returns in cost reduction make up only half of technology ROI. A benchmark study on document management found that a majority of companies had seen measurable increase in user productivity, whereas less than half recorded direct returns.

The VIOLA value model captures business impact at the measurable metric level and translates it into business value of generic value drivers such as revenue growth, margin improvement, and increased capital efficiency. For this purpose, the model provides a hierarchical structure of value drivers and metrics. The structure is referred to as a *value driver tree* where the root is the shareholder value or profitability, and the leaf nodes are measurable operational metrics. The leaf metrics nodes are connected to the root through multiple layers of performance indicators and value drivers. The initial framework of a value driver tree can be derived from a standard such as the metrics tree associated with the APQC Process Classification Framework.

Alternatively, correlations of value drivers and performance metrics can be identified by a regression analysis of historical, empirical data. Once a basic value driver tree is built, then it can be customized for a specific business in practices. The linkage between any two nodes in the value driver tree is signified by the impact level of a child to its parent, where the impact levels of all children add up to 100%. The impact levels can be derived by mining empirical data or assigned speculative values for sensitivity analyses. A sample value driver tree is shown in Fig. 5.

The completed value driver model is a probabilistic graphical model known as *Bayesian belief network* or simply *belief network*. A Bayesian network is a directed acyclic graph on nodes (e.g., value drivers) representing variables, and arcs representing probabilistic dependency relations among the variables and local probability distributions for each variable given values of its parents. In VIOLA, the directed arcs of the graph are interpreted as representing causal relationships. The belief net-

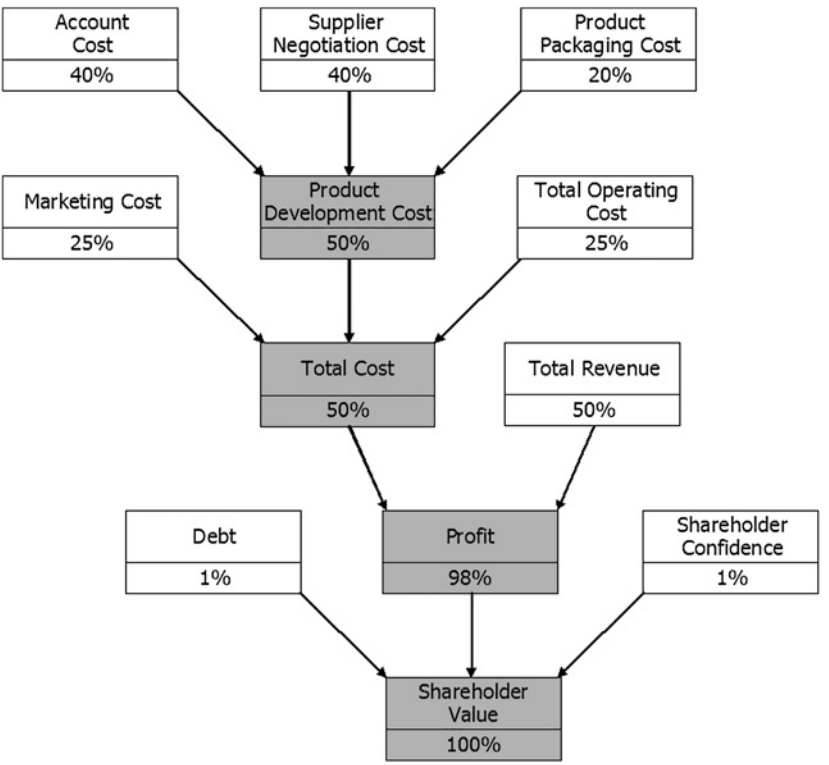


FIG. 5. A sample value driver tree.

work model can be used to answer probabilistic queries about the nodes. In other words, it provides a probabilistic inference, which is a process of computing posterior distribution of variables given evidence. For example, it can be used to find out updated knowledge of the state of subset of variables when other variables (the evidence variables) are observed.

The VIOLA value engine calculates the expected values of a set of value drivers that are affected by the implementation of one or more IT solutions and/or business capabilities. To support this calculation, the value model extends the value driver tree by linking IT capabilities and solutions to business activities, and to operational metrics in the value driver tree. Additionally, the expected value calculation requires user input for certain edges in the trees. First, it requires the *usage factor* on each edge between an IT capability and a business activity which indicates how much of the activity the IT capability is used for. Second, it requires the *improvement factor* on each edge between an activity and a metric node which indicates how much the metric is improved by the IT capability. Again, the user input values can be derived

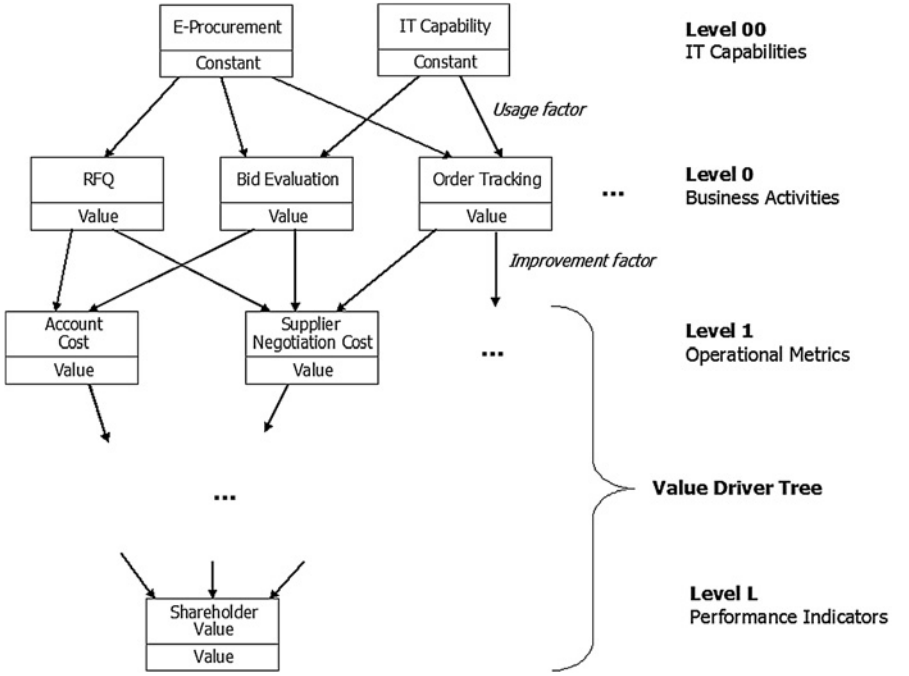


FIG. 6. A simple VIOLA value model.

by mining empirical data or assigned speculative numbers for sensitivity analyses. Figure 6 shows a simple VIOLA value model.

The value engine uses a recursive algorithm to calculate the expected values of value drivers affected by one or more IT capabilities. The mean value of a node, n_p, in the value driver tree is derived as follows:

$$E[n_p] = \sum_{i=1,k} \left(E[n_{c_i}] \times E[e_{c_i}] \right),$$

where $E[n_{ci}]$ and $E[e_{ci}]$ are the mean of a child node n_{ci} and its edge to n_p, respectively, and k is the number of children n_p has.

The VIOLA value engine provides a robust sensitivity analysis for validating various value propositions, because it can take the impact factors in range and present the expected business values in confidence intervals. Also, it can capture synergistic or cannibalizing effects of different IT capabilities and solutions as separate user input (referred to as *inflation* and *deflation factor*) and take it into account for the value calculation. In addition, the engine can capture statistical dependencies via correlations. Finally, the value model can be extended by mapping value drivers to standard financial measures and ROI terms for generating business reports, as described in the previous section.

2.3.3 Business Reports

As results of the business analyses, both quantitative and qualitative, VIOLA can generate a number of business intelligence reports for an executive summary of the analyses, including sophisticated interactive charts, as follows: (1) Value Driver Report, which summarizes the impact of investments in terms of value drivers in a structured format, with a number of illuminating interactive charts; (2) Financial Measure Report, which translates the impact of the solutions on value drivers to a set of standard financial measures, again, with a number of illuminating interactive charts. A sample report is shown in Fig. 7; and (3) ROI Report, which translates the impact of the solutions into a set of cash flow measures over time and provides the benefits of the investments in terms of standard ROI terms.

2.4 Implementation

To validate the proposed business model and the analysis capabilities described in the previous sections, we implemented the VIOLA system. The functional architecture of VIOLA consists of four layers:

- *The Analysis Module Layer*, which provides the presentation of business entities and allows the user to interact with them. It provides a variety of capabilities for

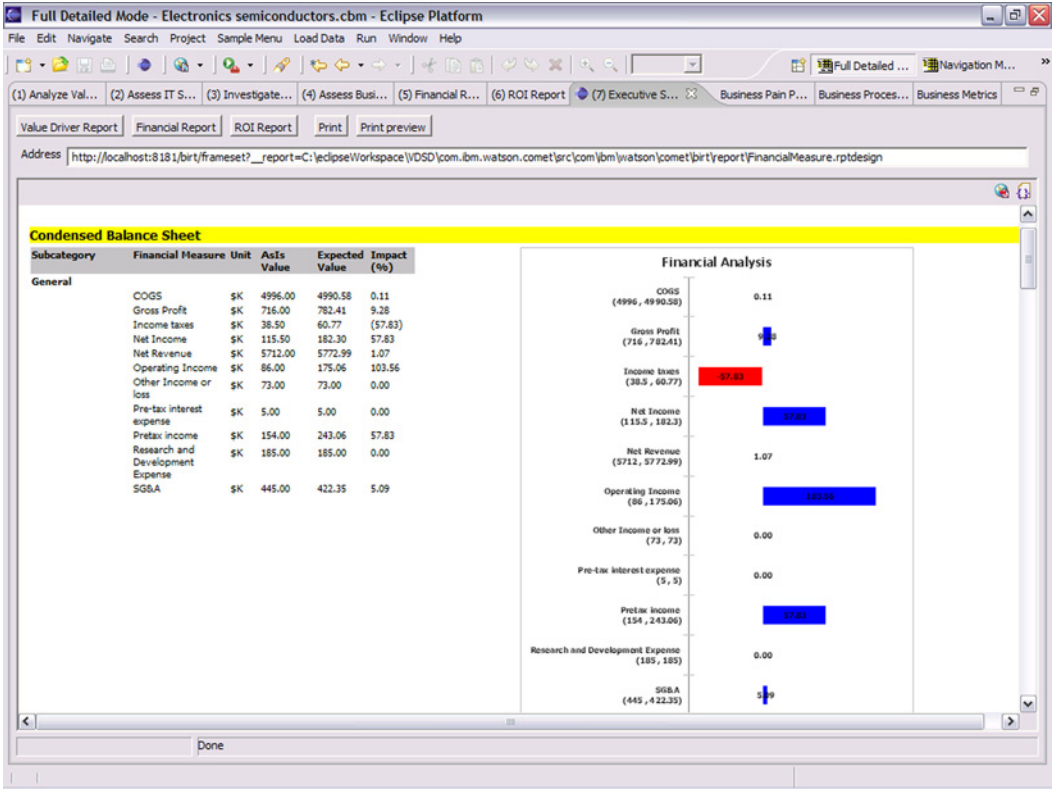


FIG. 7. A sample financial measure report.

both qualitative and quantitative analyses of business, and helps build business transformation roadmaps driven by business strategies and based on measurable values of technologies;

- *The Analysis Engine Layer*, which provides a set of generic algorithms for supporting the analysis capabilities of the presentation and interaction layer, including various engines for value analysis, semantic analysis, optimization and data mining;
- *The VIOLA Model Layer*, which provides data schema for supporting the business views and the analysis capabilities, including models for business components, processes and activities, value drivers, operational metrics and performance indicators, and IT infrastructure, as shown in Fig. 8; and
- *The CBM Tool Layer*, which is a set of CBM-related tools developed by IBM Research including the Core Tool which allows creating and viewing CBM maps, and the Repository which allows sharing of CBM maps.

The VIOLA system, including the CBM Tool Layer, was implemented on the Eclipse platform. Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software. The platform defines the set of frameworks and common services that collectively make up "integration-ware" required to support the use of Eclipse

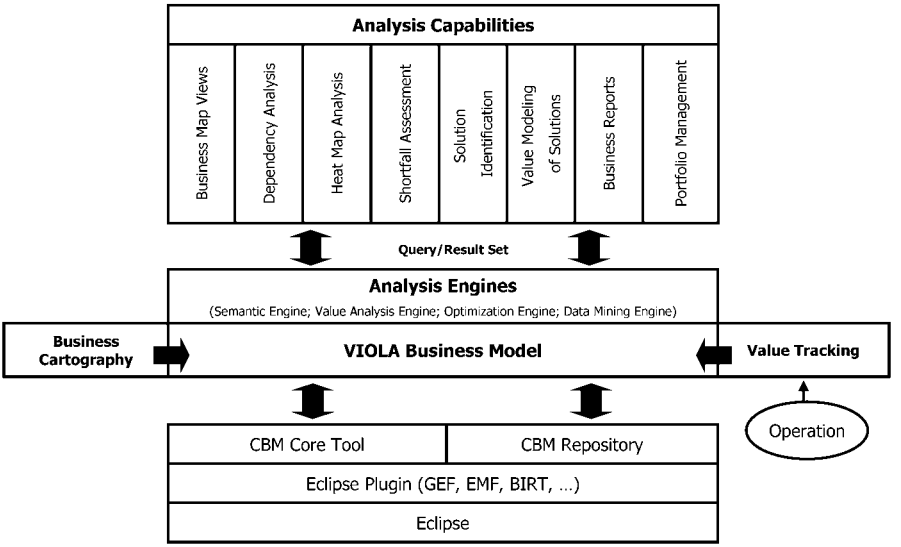


FIG. 8. Functional architecture of VIOLA.

as a component model, as a rich client platform and as a comprehensive tool integration platform. The Analysis Modules of VIOLA are implemented as Eclipse plug-ins by using the Eclipse Plugin Development Environment (PDE). The plug-in development environment provides a number of views and editors that make it easier to build plug-ins for Eclipse. The PDE made integrating plug-ins straightforward for the implementation of the VIOLA system, which utilized the several open source Eclipse plug-in projects, in addition to the business analysis plug-ins.

For a structured data model for implementing the VIOLA model and future needs for model interoperability, the CBM Tool and the VIOLA system use Eclipse Modeling Framework. EMF is a modeling framework and code generation facility for building tools and other applications based on a structured data model. It provides the foundation for interoperability with other EMF-based tools and applications. For the implementation of user interfaces, the CBM Tool and the VIOLA system utilized Graphical Editing Framework [16]. GEF allows developers to create a rich graphical editor from an existing application model. GEF employs an MVC (model-view-controller) architecture which enables simple changes to be applied to the model from the view.

VIOLA employed Derby (its Eclipse plug-in version) for data store. The Derby project develops open source database technology that is pure Java, easy to use, small footprint, standards based, and secure. Finally, the VIOLA system employed Business Intelligence and Reporting Tool (BIRT) for implementing various reporting capabilities. BIRT is an Eclipse-based open source reporting system for Web applications. It has two main components: a report designer based on Eclipse, and a runtime component that runs on an application server. BIRT also offers a charting engine that allows adding charts to applications.

2.5 Related Work

Value-oriented business transformation requires a structured approach to correlating value with business processes and also with IT capabilities, measuring and tracking value, and delivering value through business processes. Also, realization of value-based business transformation depends on an end-to-end approach to identifying business components and IT shortfalls associated with business pain points; modeling to capture and represent relationships among business components and activities, metrics, value drivers and IT solutions; sophisticated value modeling to quantify business values of business activities and IT capabilities and prioritize IT initiatives.

There are precursory thoughts on value-oriented business transformation found in research reports from research firms. For example, a Forrester research report [17] discusses “installing value-based thinking.” The report describes that a strategic IT

organization contributes, directly and indirectly, to the organization's ability to execute its business plan. Therefore, strategic thinking for the IT manager must include an analysis of the impact that major decisions will have outside the IT organization. This focus must be driven down through the organization so that the business implications of decisions are considered at all levels. The research firm also provides related discussion on the application of balanced scorecard for IT and value metrics for IT along with ROI analysis of IT, "IT Value Management," a disciplined approach to quantifying technology benefits, and improving the financial justification process for technology investments, and calculating the value of specific business and IT capabilities such as faster time-to-market [13,18,51].

There are numerous studies conducted for business process improvement, including reengineering, process benchmarking, process management, theory of constraints, total quality management, Six Sigma, and ISO (International Organization for Standardization) 9000 requirements for quality management system, to name a few. However, the presented VIOLA methodology is different from these business transformation methodologies in a number of ways. VIOLA directly focuses on business value rather than is connected indirectly through processes. It provides a holistic approach to realizing value in business. It adopts a multi-layer model linking business and IT semantics. It departs from the traditional process-based model of business, and employs a component-based model of business, which provides a number of advantages for qualitative analysis for business. It employs a multi-level model of key performance drivers, operational metrics and value drivers to support quantitative business analysis. Finally, it utilizes semantic technology for representing semantics and relationships of business components, activities, metrics and value drivers, and enables automated reasoning among them.

Also, there are numerous studies on return-on-investment, metrics measurement and analysis models. *Total Cost of Ownership* (TCO) Model by the Gartner Group provides a deterministic ROI model for calculation designed to help consumers and enterprise managers assess direct and indirect costs as well as benefits related to the purchase of computer software or hardware [14]. Kaplan and Norton's the *balanced scorecard* provides a method intended to give managers a fast, comprehensive view of the performance of a business [28]. The VIOLA methodology is different from these business performance approaches in a number of ways. It provides a comprehensive value model that captures a multi-level model of value drivers associated business activities and components. Also, it provides a holistic approach to a multi-layer model linking business and IT semantics.

Recently, a number of consulting companies also provide products and services in line with the value-based thinking. For example, Stern Stewart & Co. and Accenture offers the high performance business approach, which uses Economic Value Added (EVA) as a metrics for demonstrating the client's expected economic return from

a business and performance improvement project [50]. EVA shares similar objectives with VIOLA and also provides certain elements that are similar to what we are working on such as "client scorecard" to identify client key issues, a simple value model, and connection of business activities with metrics. However, VIOLA is one step ahead with the CBM approach to identify "hot" components and IT shortfalls, the semantic modeling to capture and represent relationships among components, activities, value drivers and IT solution; the value modeling that translates the business value of IT solutions and prioritizes IT projects.

2.6 Summary

Ability to demonstrate the value of IT and services is critical to business transformation initiatives and project portfolio management in enterprises. Value-based transformation requires a structured and holistic approach to correlating value with business processes and IT capabilities, measuring and tracking value, and delivering value through business processes. This section presented a value-oriented, model-driven business transformation methodology referred to as VIOLA. It provides an end-to-end approach to link business value with key IT enablers and provide further business transformation opportunities.

VIOLA comprises four modeling elements. First, *model-driven business transformation* provides a multi-layer model linking business and IT semantics, and enables IT and services to accurately reflect and be driven by business value. The upper layers of model represent business semantics in the terms familiar to business executives, business managers and analysts such as key performance indicators, operational metrics, business processes, activities and governance. The lower layers of model represent IT architecture comprising a wide range of services implemented in IT infrastructure such as service-oriented architecture.

Second, *component business modeling* provides a strategic-level business view of an enterprise in a dashboard, and enables business analyses based on business impacts. The CBM methodology enables a number of qualitative business analysis for identifying "hot" components and IT shortfalls that are associated with business pain points. Third, *value modeling* specifies multiple levels of key performance drivers, operational metrics and value drivers, supports various quantitative business analyses including sensitivity analyses, and enables business optimization and risk assessment.

Finally, *semantic business modeling* put together business components, business activities, performance drivers and IT by capturing their relationships. It formally represents meaning of business components, metrics, and their relationships and enables automated reasoning to identify dependencies and causality relationships among business entities.

The VIOLA methodology and its software solution is a result from an ongoing research project on business design and transformation at the IBM Research Center. With a methodology and a research prototype in place, we work with practitioners to validate them with real-world business transformation initiatives. In addition to the tool and methodology, in practice, the availability of useful and accurate content and information of business components, value drivers, processes and solutions is critical to meaningful analyses.

3. Model-Driven Ontology Engineering

W3C's Semantic Web [2,55] provides a common framework that allows data to be shared and reused across application and enterprise. It is based on the Resource Description Framework (RDF), which describes various resources using XML (Extensible Markup Language) for syntax and URIs (Uniform Resource Identifiers) for naming [29], and Web Ontology Language (OWL), which provides modeling constructs for specifying and inferring about knowledge [49]. As the Semantic Web shapes the future of the Web, it becomes more and more important in software engineering and enterprise application development. To meet the needs, a number of tools and systems for ontology development and management have been developed.

While these ontology engineering tools provide a relatively complete stack of ontology management support and are used successfully in certain domains, there still remains a gap between the ontology engineering tools and the traditional software engineering. For more than a decade, software engineering has been established on different modeling languages and methodologies such as Object Management Group's Unified Modeling Language (UML). This difference in modeling languages and methodologies causes difficulties in large-scale enterprise application development involving the Semantic Web technologies. The existing ontology engineering tools provide only an ad hoc approach to bridging this gap with limited functionality and performance. The creation of ontologies and their use in software engineering projects is currently cumbersome and not seamless. The transformation of UML models to OWL ontologies and vice versa is conducted only in an ad hoc and incomplete way. Therefore, it is difficult to utilize the vast investment of enterprises in software engineering models, which are often accumulated over a decade, in ontology engineering. For the Semantic Web to have impact on enterprises and their business, and also to be widely accepted as a value-adding technology, bridging this gap in software and ontology engineering is critical.

The primary objective of the work presented in this section is to bridge this gap between two different, but complementary engineering disciplines with a system-

atic approach. We leverage OMG's Model-Driven Architecture (MDA) [3,40] and Ontology Definition Metamodel (ODM) [38] to provide model transformation. This approach allows seamlessly supporting existing models in UML and other languages in Semantic Web-based software development. In addition, it allows exploiting the availability and features of UML tools for creation of vocabularies and ontologies. Furthermore, MDA enables code generation and facilitates tool development. This section presents a model-driven approach to ontology engineering. It describes the architecture of the ontology engineering system, and mappings between UML and OWL for model transformation.

The rest of this section is structured as follows: In Section 3.1, we describe a number of existing software tools for ontology development and management. It discusses a gap between these ontology engineering tools and the traditional software engineering tools. Sections 3.2 and 3.3 summarize technical background information on the Model-Driven Architecture and Ontology Definition Metamodel, respectively. In Section 3.4, we explain how EMF-based technologies for MDA and ODM are used to realize the proposed system for ontology engineering. Section 3.5 presents an implementation of the model transformation component. Section 3.6 describes use scenarios illustrating how the features of the developed ontology engineering tool can be utilized in real-world applications. Finally, Section 3.7 provides a summary.

3.1 Traditional Ontology Management Systems

In recent years, there has been a surge of interest in using ontological information for communicating knowledge among software systems. As a result, an increasing range of software systems engage in a variety of ontology management tasks, including the creation, storage, search, query, reuse, maintenance, and integration of ontologies. Recently, there have been efforts to externalize such ontology management burden from individual software systems and put them together in middleware known as an ontology management system. An ontology management system provides a mechanism to deal with ontological information at an appropriate level of abstraction. By using programming interfaces and query languages the ontology management system provides, application programs can manipulate and query ontologies without the need to know their details or to re-implement the semantics of standard ontology languages. Examples of such ontology management systems include Protégé [47], Jena [6], Sesame [48], Pellet [45], KAON [37], Jastor [27], D2RQ [8], RStar [25,33], and SnoBase [31].

While these ontology engineering tools provide a stack of ontology management support, they also show certain limitations in supporting large-scale software engineering projects. Participating in a number of enterprise application development projects by using the SnoBase and RStar Ontology Management System, we learned

firsthand that it is critical to provide a comprehensive development environment including supporting tools and facilities for the application developers. A pick-and-choose approach to the best of the breed tools from different environments does not always work well for the majority of the developers and often results in a longer learning curve for the developers. A comprehensive ontology development environment often means a tight integration of tools for software *and* ontology engineering, and model import and transformation, among others.

Semantic markup languages such as W3C's RDF and OWL are based on the work in the logic and Artificial Intelligence communities, such as Description Logic and Knowledge Representation. The syntax of these languages is less intuitive to those trained for object-oriented programming and simple XML-based languages. The lack of a tightly integrated development environment for software and ontology engineering makes the job of subject matter experts and software engineers difficult, and often affects negatively to the adoption of the semantic technology in industry. An effective ontology application development environment should bridge this gap between software engineering and ontology engineering by providing a seamlessly integrated environment.

Another consideration for industry adoption of the semantic Web technology is the interoperability of the semantic markup languages with the well-established and widely-accepted industry standard modeling languages and methodologies such as Entity Relationship (ER) modeling and Unified Modeling Language (UML). Enterprises developed software models in these languages for more than a decade and invested significantly in building systems around them. Despite all the theoretical advantages the semantic technology brings in, in practice, it is highly unlikely that the enterprises abandon the legacy systems and develop new systems around the semantic Web technology. Instead, users in industry would be interested in the interoperability of the modeling languages, and the reuse of the existing models and data with the semantic Web technology. The traditional ontology management systems currently provide only ad hoc and incomplete methods for the model interoperability. To address the practical requirements of industry, this section presents a novel approach to ontology engineering based on the Model Driven Architecture (MDA), which enables software engineers and users to design, build, integrate and manage ontologies and software applications in an integrated development environment.

3.2 Model-Driven Architecture

Before presenting the model-driven approach to ontology engineering, this section summarizes the Object Management Group's Model Driven Architecture, which is one of the two pillars of the system's architecture, along with Ontology Definition Metamodel.

In the history of software engineering, there has been a notable increase of the use of models and the level of abstraction in the models. Modeling has become separated from underlying development and deployment platforms, making them more reusable and easier to create and modify by domain experts, and requiring less knowledge of specific deployment systems. This trend places software modeling closer to knowledge engineering. The current stage in this evolution is the *Model Driven Architecture*, which grew out of the standards work conducted in the 1990s for the Unified Modeling Language.

The basic idea of MDA is that the system functionality is defined as a platform-independent model, using an appropriate specification language and then translated to one or more platform-specific models for the actual implementation. To accomplish this goal, the MDA defines an architecture that provides a set of guidelines for structuring specifications expressed as models. The translation between platform-independent model and platform-specific models is normally performed using automated tools. Specifically, MDA defines three levels of abstraction: *Computation Independent Model* (CIM), *Platform Independent Model* (PIM) and *Platform Specific Model* (PSM). CIM is a view of a system that does not show the details of a system structure. In software engineering, it is also known as a *domain model*, which is concerned by domain experts. It is similar to the concept of ontology. PIM is a model that is computation dependent, but it is not aware of specific computer platform details. In other words, it is targeted for a technology-neutral virtual machine. Specification of complete system is completed with PSM. The goal is to move human work from PSM to CIM and PIM, and let the detail implementation for a specific platform be generated as much as possible by automated tools which perform the transformation from PIM to PSM.

MDA comprises of a four-layer metamodel architecture: meta-metamodel (M3) layer, metamodel (M2) layer, model (M1) layer, and instance (M0) layer. Also, it utilizes several complementary standards from OMG including *Meta-Object Facility* (MOF), *Unified Modeling Language* (UML) and *XML Metadata Interchange* (XMI). On the top of the MDA architecture is the meta-metamodel, i.e., MOF. It defines an abstract language and framework for specifying, constructing and managing technology neutral metamodels. It is the foundation for defining any modeling language such as UML or even MOF itself. MOF also defines a framework for implementing repositories that hold metadata (e.g., models) described by metamodels [36]. The main objective of having the four layers with a common meta-metamodel is to support multiple metamodels and models and to enable their extensibility, integration and generic model and metamodel management.

All metamodels, standard or custom, defined by MOF are positioned on the M2 layer. One of these is UML, a graphical modeling language for specifying, visualizing and documenting software systems. With *UML profiles*, basic UML concepts

(e.g., class, association, etc.) can be extended with new concepts (*stereotypes*) and adapted to specific modeling needs. The models of the real world, represented by concepts defined in the corresponding metamodel at M2 layer (e.g., UML metamodel) are on M1 layer. Finally, at M0 layer, are things from the real world. Another related standard is XMI. It defines mapping from MOF-defined metamodels to XML documents and schemas. Because of versatile software tool availability for XML, XMI representations of models, metamodels and meta-metamodel facilitate their sharing in software application development.

MOF tools use metamodels to generate code for managing models and metadata. The generated code includes access mechanisms, or application programming interfaces, to read and manipulate, serialize and transform, and abstract the details of various interfaces based on access patterns. *Eclipse Modeling Framework* (EMF) [12] provides a Java implementation of a core subset of the MOF API. EMF started out as an implementation of the MOF specification, and evolved into a generic modeling framework and code generation facility for building tools and other applications based on a structured data model. The MOF-like core metamodel in EMF is called *Ecore*. From a model specification written in XMI, EMF generates tools and runtime support to produce a set of Java classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. Models can be specified using annotated Java, XML documents, or modeling tools like Rational Rose, then imported into EMF. It is important to note that EMF provides the foundation for interoperability with other EMF-based tools and applications. The proposed MDA-based system leverages EMF for implementing ontology management tools which run on the Eclipse environment, and utilizes its support for model interoperability.

3.3 Ontology Definition Metamodel

MDA and its four-layer architecture provide a solid basis for defining metamodels of any modeling language, and so provide a foundation for bringing together software engineering and methodologies such as UML with the semantic technology based on W3C's RDF and OWL. Once a semantic markup language such as OWL is defined in MOF, its users can utilize MOF's capabilities for modeling creation, model management, code generation, and interoperability with other MOF-defined metamodels.

Another OMG standard, Ontology Definition Metamodel (ODM) [38] takes this approach. To comprehend common ontology concepts, ODM uses as a starting point OWL which is the result of the evolution of existing ontology representation languages. ODM defines individual constructs of OWL in MOF, creating an ODM metamodel. To leverage graphical modeling capabilities of UML in dealing with

OWL constructs, ODM also defines an ontology UML profile to support UML notation for ontology definition. This profile enables graphical editing of ontologies in OWL using UML diagrams as well as other benefits of using mature UML CASE (Computer-Aided Software Engineering) tools. Finally, the following bi-directional mappings between metamodels complete the picture:

1. mappings between OWL and ODM,
2. mappings between ODM and the ontology UML profile, and
3. mappings from the ontology UML profile to other UML profiles.

Figure 9 shows a simple example of the bi-directional mappings between metamodels. In practice, both UML and ODM models are serialized in XMI, and OWL model in XML, the two-way mappings can be implemented by transformations based on XSLT (Extensible Stylesheet Language Transformation) [9]. Gasevic et al. summarized existing approaches and tools for transformation between UML models (or UML profiles) and OWL models in [9,15], and pointed out that the XSLT-based transformation is widely used in them. Our work utilized EMF-based transformations, instead of XSLT, to leverage EMF's generic modeling framework and code generation facility for building tools and other applications. We implemented *EODM* (EMF-based ODM), which is the underlying object model generated from ODM by using EMF, for model transformations among OWL, UML and other modeling languages. More details will be given in the next section.

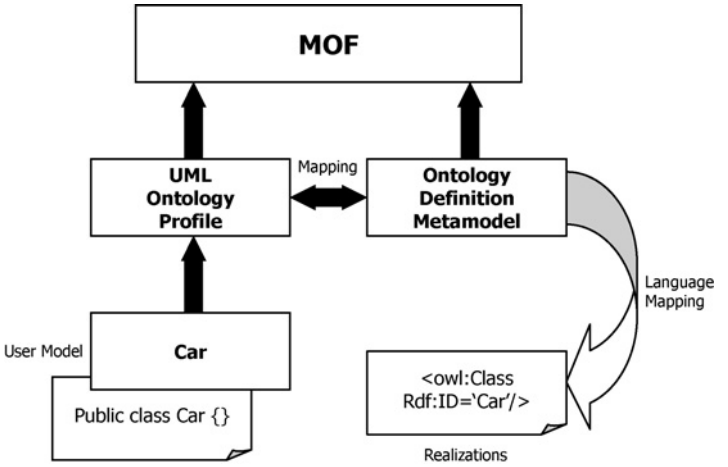


FIG. 9. Bi-directional mapping among metamodels.

Before moving to the main body of this section, it is useful to briefly mention yet another related effort from W3C, namely, *Ontology Driven Architecture* (ODA) [52]. It combines MDA with the semantic technology differently from the ODM approach. It attempts to augment the MDA standards and methodology stack with the semantic technology to improve the discipline. It aims to enable unambiguous representation of domain terminology, distinct from the rules, enable automated consistency checking and validation of invariant rules, preconditions, and post-conditions, and support knowledge-based terminology mediation and transformation for increased scalability and composition of components. This effort still is in its infancy and at a draft stage.

3.4 EMF-Based Ontology Engineering System

For realizing the model-driven ontology engineering, we utilize the Eclipse Modeling Framework, which is open source MDA infrastructure for integration of modeling tools [12]. A model specification described in various modeling languages including UML, XML Schema, and annotated Java source can be imported into EMF. Then EMF produces a set of Java classes for the model, a set of adapter classes that enable viewing and editing of the model, and a basic editor. In its current implementation, EMF does not provide formal semantics definitions, inference and the related model specifications. Our work adds this capability to EMF for providing a comprehensive ontology engineering environment and dynamic application integration.

For adding the semantic model transformation capability to EMF, we leverage the specification of Ontology Definition Metamodel. By using EMF and ODM, we generated a foundational memory model, i.e., Java classes, for the constructs of OWL. This foundational memory model is referred to as *EODM* (EMF-based Ontology Definition Metamodel). By adding several necessary helper classes and methods to EODM, we can use it to create, edit, and navigate any models in OWL.

Also, we added an OWL parser to EODM, which can load OWL files into EMF and generate OWL files from EMF, i.e., serialize EMF models to standard OWL files in XML. The parser utilizes an XMI adaptor which enables the transformation between the OWL models and EODM Ecore models. The transformation is made possible by the bi-directional mapping between OWL and the Ecore metamodel. The transformation opens a way to interoperability between OWL models and other EMF supported models, which currently include ones defined in UML, XML Schema, and annotated Java classes. The support of other models such as Entity Relationship models in EMF will be provided in the near future. By leveraging the OWL parser and the bi-directional transformation between the OWL models and the Ecore models,

ontology application developers can develop ontologies using their favorite model building tools, import them into EMF, transform their models into OWL ontologies, enrich them with semantics, leverage their inference capability, and utilize the comprehensive development facility of Eclipse and EMF.

To be more specific, the EODM Ecore model is the MOF core model that represents ontologies in memory. It is an intermediate model for imported and transformed legacy models, as well as the generated ontology, Java code, Java editor and Java edit. The development environment allows its users to manipulate EODM Ecore models, enrich it with semantic specification, and generate Java code. A default set of bi-directional mappings between metamodels of legacy models and OWL are developed in EMF. Eclipse plug-in developers can extend the mappings to handle other types of legacy models, or other elements in legacy models specifying semantics. The generated Java editor and Java edit provide ready-to-use visual tools to populate or manipulated instances of OWL models. The visual tools are actually copies of the standard methods of supporting application development in EMF. Figure 10 illustrates logical operations of the EMF-based ontology engineering system.

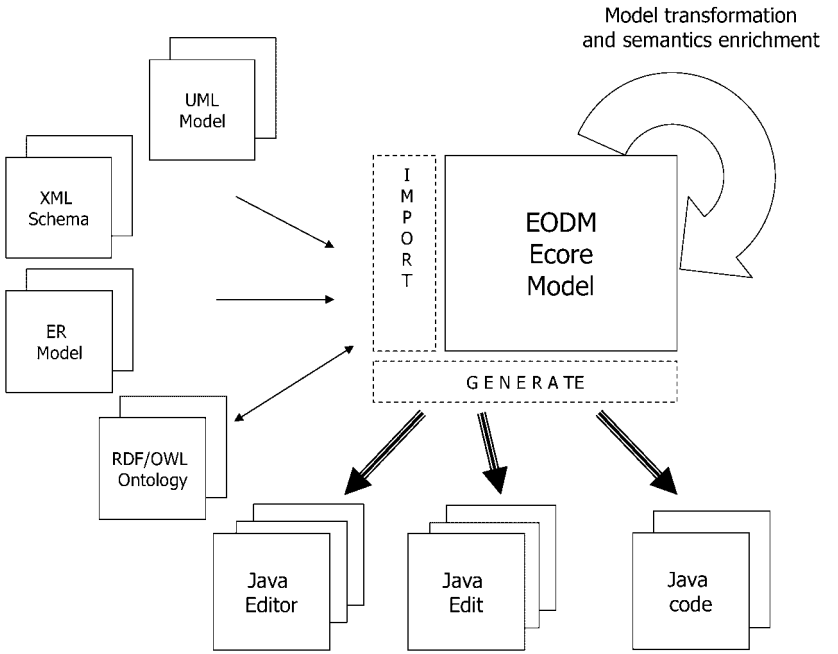


FIG. 10. EMF-based ontology engineering system (logical view).

We had two primary design objectives for the EMF-based ontology engineering system: first, support for the entire lifecycle of ontology engineering, and, second, avoiding reinvention of tools and facilities that are already proven to work in software engineering. To achieve these objectives, we designed a software stack which consists of six interdependent layers.

At the core of this EMF-based ontology engineering system is the EODM model, which is derived from the Ontology Definition Metamodel and implemented in Eclipse Modeling Framework. The bottom layer, *EODM core model*, provides the basic Java programming model for OWL ontologies with all the necessary getter and setter functions. It is automatically generated by EMF from the UML models for OWL. To this generated core model implementation, certain *utility classes and methods* are added, to benefit Java programmers. On top of the EODM core model comes the *OWL Parser* which parses OWL ontologies, translates them into EODM models, and serializes EODM models to standard RDF/XML files. EODM core and OWL Parser form the foundation for the entire software stack. The top layer is composed of three relatively independent components that are build on top of this foundation. The first component is the *OWL Inference Engine*. It takes an EODM model as input, and executes user queries, reasoning about instances and relationships among instances and classes. The second component is the *Model Transformation*. It imports existing conceptual models represented in various modeling languages such as UML, ER diagrams, and Java interfaces. Then, it transforms the models into one or more EODM models. Finally, the *OWL Editor* provides a graphical ontology authoring environment where OWL ontologies in graphic notations are serialized to OWL

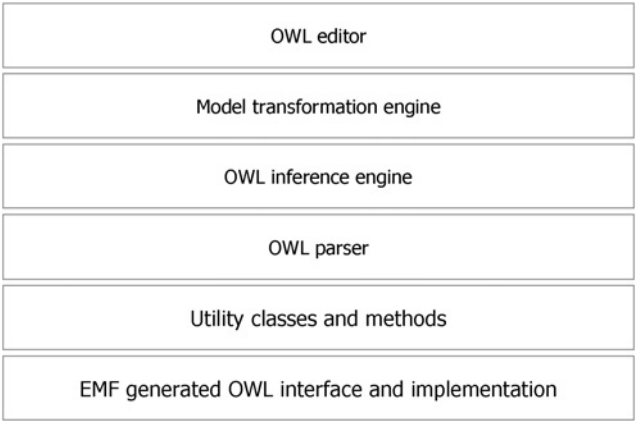


FIG. 11. EMF-based ontology engineering system layers.

files in a standard XML format. Figure 11 shows the components of the EMF-based ontology engineering system.

3.5 Model Transformation

The EMF-based ontology engineering system provides tightly integrated environment for software and ontology engineering, providing a stack of useful components. EODM provides the run-time library that allows applications to input and output OWL ontologies, manipulate them by using Java objects, invoke the inference engine and access result sets, and transform among ontologies and other legacy models.

The EODM core model provides useful classes and methods to access OWL ontologies and their instances. Its metamodel is defined in the Ontology Definition Metamodel (ODM) specification [38]. It is an MOF2 compliant metamodel that allows users to define ontologies by using those constructs defined in RDF Schema and OWL. ODM comprises of two packages that define the metamodels of RDF and OWL, respectively. The OWL package inherits classes from the RDF package, and extends it. Figure 12 illustrates the class definition of the RDF package. The UML model of the packages is augmented by a number of bi-directional references to generate APIs that leverage notification and messaging mechanisms in EMF. Also, there are certain design patterns, such as Factory and Singleton, embedded in the code generation engine of EMF. Therefore, the EODM core model automatically complies with the design practices and benefit software engineers.

This model transformation module in the EMF-based ontology engineering system addresses the ontology acquisition and model interoperability issues, as we discussed earlier. Enterprises developed IT models in various modeling languages such as UML and ER diagrams for several decades and invested heavily in building systems around them. It is important for the enterprises to protect their investment in the legacy systems. Also, it is important to leverage domain knowledge captured in the existing IT models. Thus, users in industry are interested in the interoperability of the modeling languages and the reuse of the existing models with the semantic Web technology. The interoperability allows exploiting the availability and features of UML tools for creation of vocabularies and ontologies. In addition, it allows augmenting the legacy models with formal semantics, and enabling an inference capability with the models, which can return sound and complete query results.

Figure 13 shows the Ecore metamodel and its role in MDA. Ecore is a java implementation of the MOF model. Therefore, we can we utilize Ecore as an intermediate model to support model transformation between OWL and other modeling languages. For example, a UML class diagram can be, first, transformed into an Ecore model by using the mapping between the UML metamodel and the Ecore metamodel,

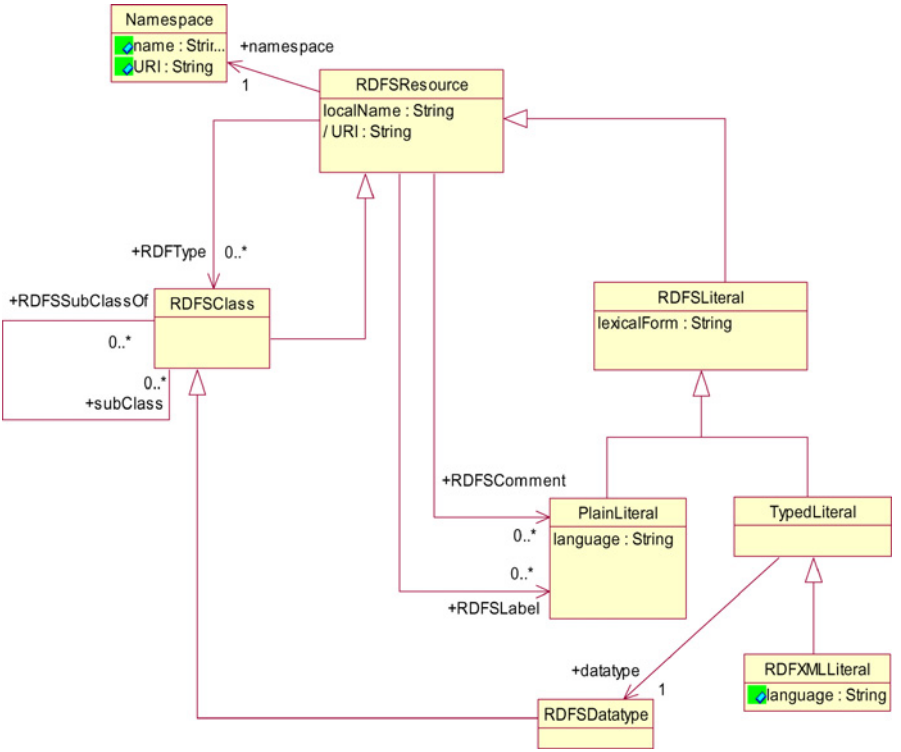
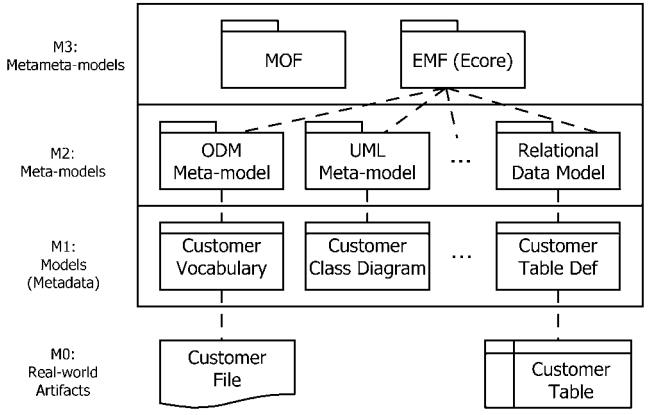
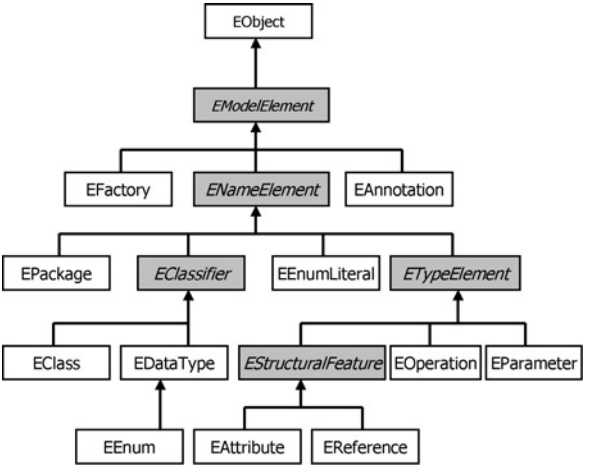


FIG. 12. Class definition in ODM.

and, then, the resulting Ecore model can be transformed into an EODM model by using the mapping between the ODM metamodel and the Ecore metamodel. This way, it is possible to construct an ontology from legacy models. On the other hand, an OWL ontology can be transformed into a UML diagram. There already exist transformations defined between the Ecore model and other modeling languages such as UML, XSD and Java interfaces. In EODM, a mapping between the Ecore metamodel and the ODM OWL metamodel is defined. Then, we can implement model transformation by leveraging well-developed facilities of EMF as much as possible. XSLT-based approaches are more or less affected by the syntax of OWL and XMI, because the files written in these languages can be represented in different forms, but with the same semantics [9,15]. Our approach is based on a memory model mapping approach, and, thus, independent of the syntax of OWL and XMI. However, it is fair and important to note that a problem of model transformation is about expressive-



(a) Ecore metamodel as an implementation of MOF



(b) Ecore metamodel structure

FIG. 13. Ecore metamodel in MDA.

ness differences between models. With the current Ecore model, an OWL ontology cannot be fully transformed into a UML model without loss of semantics, and vice versa. The expressiveness of the Ecore model is gradually improved to cover more models.

Figure 14 depicts the bi-directional mappings defined between the Ecore metamodel and the EODM OWL metamodel. An OWL ontology is transformed to an

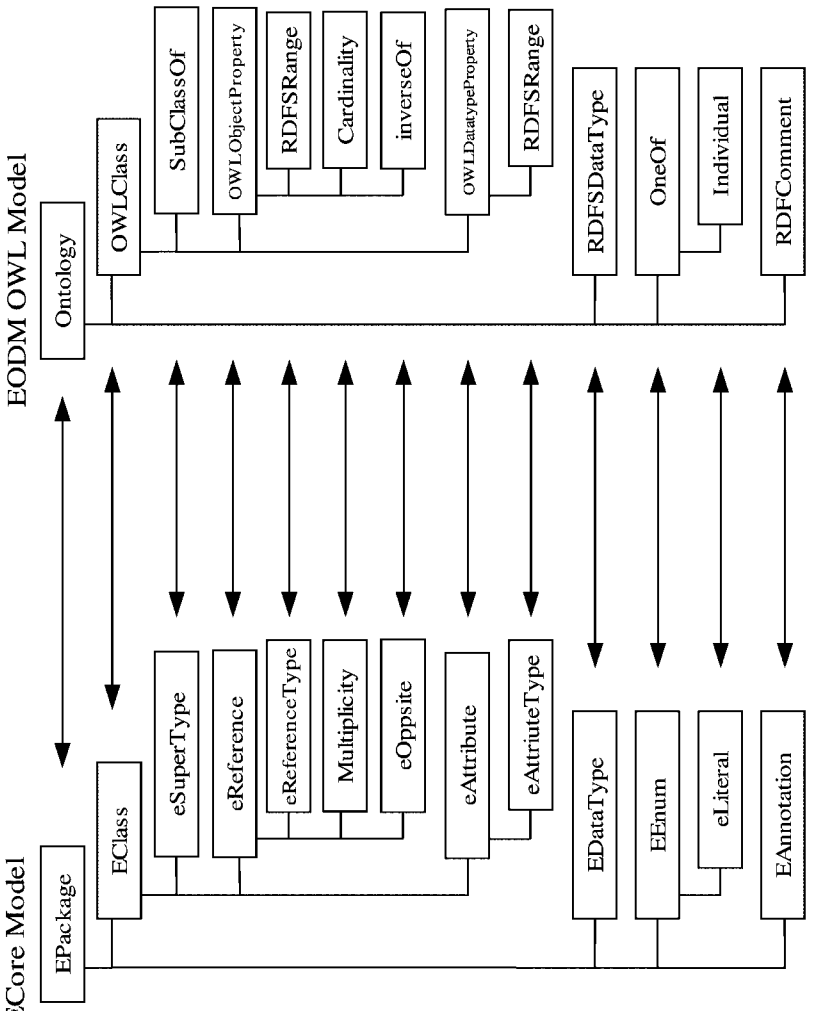


FIG. 14. Transformation between OWL and Ecore.

EPackage and vice versa; an OWL class to an EClass, etc. While the transformation from OWL to Ecore model looks straightforward, there are a few gaps. As in UML, Eclass is a first-class entity in the Ecore model. All other entities such as properties are subordinates to Eclass. In OWL, however, all entities in OWL are equal. Thus, different entities must have different names in OWL. For example, if two properties belonging to two different Eclasses have an identical name, a straightforward transformation will cause a name conflict problem. The EODM Transformation engine renames properties with an identical name to ensure a unique name for every entity. Another gap comes from the difference in expressiveness from different modeling languages. OWL is a formal language which is based on Description Logic. OWL is more expressive than the Ecore model. There are several OWL constructs that the Ecore model does support, e.g., OWL property restrictions used for precise definition of concepts. Therefore, some semantics are lost inevitably when conducting transformation from OWL to Ecore. Also, The Ecore model does not support inference of OWL. Particularly, anonymous classes created by using OWL restrictions make the situation with inference even more difficult. To address these gaps, the EODM Model Transformation engine currently employs the following tactics:

- It appends all unsupported OWL constructs as comments;
- It utilizes the inference engine during transformation to capture all implicit subsumption relationships;
- It only transforms named OWL classes, and discards all anonymous classes; and
- It renames properties with an identical name to ensure a unique name for every entity.

3.6 Use Scenarios

This section presents use scenarios illustrating how the features of the proposed EMF-based ontology engineering tool can be utilized in real-world applications. Our example is the *model-driven business transformation* [20,30]. Business transformation employs business models such as *component business models* [21] to identify opportunities for reducing costs or improve business processes. The model-driven approach to business transformation requires a model representation of a variety of business entities such as business processes, components, competencies, activities, resources, metrics, KPIs (Key Performance Indicators), etc. and their relations. Semantic models or ontologies provide useful representation of business models because they can effectively represent different types of relations among business

entities. Also, the automatic reasoning capability of semantic models provides an effective method for analyzing business models for identifying cost-saving or process improvement opportunities.

For example, business performance metrics are associated with business activities. By using the relations between business activities and metrics, and also the relations between business components and business activities represented in a semantic model, a business analyst can infer relations between business components and metrics. This type of analysis provides business insights into how the corporate can improve its performance metrics by addressing issues with the business components associated with the selected set of metrics. Then, by identifying, again in the semantic model, IT systems associated with the business components, the analyst may be able to suggest recommendations about IT system management to improve performance metrics.

The first step in realizing this model-driven business analysis scenario is the construction of semantic models of various business entities including business processes, components, competencies, activities, resources, operational metrics, KPIs. In many cases in most enterprises, the classes and relations of these business entities are already captured in certain legacy modeling languages such as UML class diagrams, ER diagrams, relational data models, Java interfaces, spreadsheets, or text documents. Therefore, the task of semantic model construction simplifies to transforming the legacy models and merging them into OWL ontologies. The merged OWL ontologies can be enriched with certain semantics such as generalization and specification, and cardinality constraints to enhance the effects of business analysis queries.

To summarize the model transformation process using the EMF-based ontology engineering system, it starts by capturing formal and informal semantics of legacy models. The model transformation engine transforms *formal* semantics of input legacy models into OWL models, by utilizing pre-defined mappings between OWL and the metamodels of the input models. The ontology engineering system allows an expert to look into annotations and code of legacy models, and represent the semantics in OWL models. *Informal* semantics are captured as additional axioms and added to the OWL models by using the OWL editor. Optional functions of the system, such as the source code analysis or natural language processing, facilitate automatically capturing of certain informal semantics and improve the productivity of human experts. The overall process of capturing of formal and informal semantics of legacy models and representing them in OWL models is referred to as *semantics enrichment*.

3.7 Summary

As the Semantic Web shapes the future of the Web, it becomes more and more important in software engineering and enterprise application development. However, the adoption of Semantic Web by industry has been slowed by a gap between ontology engineering tools and the traditional software engineering. Ontology engineering and software engineering have been established on different modeling languages and methodologies, which has caused difficulties in large-scale enterprise application development involving the Semantic Web technologies. Currently, transformation of UML models to OWL ontologies and vice versa is conducted only in an ad hoc and incomplete way.

This section presented a novel approach to bridging this gap between two different, but complementary engineering disciplines with a systematic approach. We leveraged OMG's Model-Driven Architecture and Ontology Definition Metamodel to provide model transformation, utilizing underlying standards including MOF-based metamodels, XMI representation, UML extension with profiling, and EMF implementation of MOF. This approach allows seamlessly supporting legacy models in UML and other languages in Semantic Web-based software development. In addition, it allows exploiting the availability and features of UML tools for creation of vocabularies and ontologies. Furthermore, it supports code generation and facilitates tool development. This section presented the methodology and architecture of the EMF-based ontology engineering system, and mappings between UML and OWL for model transformation. Finally, it presented use scenarios illustrating how the features of this system can be utilized in real-world applications.

This model-driven ontology engineering is still in its infancy. For this approach to meet its promises and scale for industry applications, a number of technical challenges need to be addressed. Some directions for further investigation include:

- a complete definition of bi-directional mappings between the Ecore metamodel and semantic metamodels to support sound and complete model transformation;
- support for more legacy modeling languages and methodologies in addition to UML, XSD (XML Schema Definition) and Java interfaces which we have addressed in the current system, e.g., relational data models and spreadsheets traditionally popular in the business environment;
- validation of the proposed advantage of utilizing features of visual UML tools for creating and editing ontologies in real-world applications;
- evaluation of EMF's capability of code generation for facilitating tool development;

- augmenting the proposed model transformation method with capabilities for source code analysis and text mining to facilitate acquisition of certain informal semantics of legacy models; and
- maturation of the holistic EMF-based ontology engineering framework by applying and validating it in real-world business applications.

4. Business Process Composition with Web Services

A *business process* refers to a process in which work is organized, coordinated, and focused to produce a valuable product or service. Business processes comprise both internal and external business partners and drive their collaboration to accomplish shared business goals by enabling highly fluid process networks. A *business process solution* consists of a model of the underlying business process (referred to as a *process model* or a *flow model*) and a set of (flow-independent) business logic modules. The abstractions of the elementary pieces of work in a flow model are called *activities*; the concrete realizations of these abstractions at process execution time are referred to as *activity implementations*. The prevalent technique for creating business process solutions follows a manual and tedious approach involving assimilation of varied process design and vendor specifications and writing vast amount of code that produces a tight inflexible coupling between processes.

Web services provide a set of technologies for creating business process solutions in an efficient, standard way. The promise of Web services is to enable a distributed environment in which any number of applications, or application components, can interoperate seamlessly within an organization or between companies in a platform-neutral, language-neutral fashion. From the perspective of business process solutions, a Web service could represent an activity within a business process, or a composite business process comprising a number of steps [32].

Building a business process solution by using Web services involves specifying the potential execution order of operations from a collection of Web services, the data shared among the Web services, which business partners are involved and how they are involved in the business process, and joint exception handling for collections of Web services. A basis for these specification tasks is the discovery, composition, and interoperation of Web services, which are primary pillars of automatic process integration and management solutions. In this section, we focus on the following two problems of the Web service-based business process automation: (1) the location of services based the capabilities of published services, and (2) the composition of business processes based on the business requirements of submitted process requests. This section discusses solutions to these problems, and, especially, focuses on the following aspects: (1) the specification of the capabilities of services and the

requirements of requests, and (2) algorithms for matching published services and submitted process requests in terms of service capabilities and requested business requirements.

The rest of this section is structured as follows: Section 4.1 summarizes the previous work on the problems of interest, discusses their limitations, and explains how the work presented in this section addresses them. Section 4.2 addresses issues involved with the specification of business requirements in process request documents. Section 4.3 discusses the specification of the attributes and capabilities of Web services in UDDI (Universal Description, Discovery and Integration). Section 4.4 presents a matching algorithm for locating services based on service capabilities and properties. Sections 4.5 and 4.6 present matching algorithms that are deigned to satisfy the business requirements and provide optimal solutions in terms of meeting certain business objectives. Section 4.7 describes a multi-attribute decision analysis for the given problem. Finally, in Section 4.8, conclusions are drawn and future work is outlined.

4.1 Related Work

Recently, there have been active studies related to the Web service-based process automation in both academia and industry. Industrial effort for the business process automation is centered on the Business Process Execution Language for Web Services (BPEL4WS), which is an XML-based workflow definition language that allows companies to describe business processes that can both consume and provide Web services [23]. Along with complementary specifications, WS-Coordination and WS-Transaction [24], BPEL4WS provides a basis for a business process automation framework, and is viewed to become the basis of a Web service standard for composition. With the BPEL4WS specification, vendors such as IBM provide workflow engines on which business processes written in BPEL4WS can be executed. Running on Web application servers such as Apache Tomcat, the workflow engines support the coordinated invocation, from within the process, of Web services.

The focus of BPEL4WS is limited to the specification of flow models and the coordinated invocation of Web services via the workflow engine. The BPEL4WS specification does not directly address the discovery of services and the composition of business processes fulfilling various business objectives. Instead, it assumes that these tasks are executed separately, and that the business manager who creates a BPEL4WS document has the information on the selected Web services before creating the process document. Also, BPEL4WS is limited in specifying business requirements and preferences that can be critical in selecting services in real-world applications.

We claim that for automating business processes and, hence, for supporting "real-time enterprises," the discovery and composition of services should be seamlessly integrated with the capabilities BPEL4WS provides, i.e., the flow model specification and the coordinated invocation of services. If this integration is achieved, a business manager can create a business process model without knowing upfront specific Web services that will be used for implementing activities of the process. Instead, s/he only needs to specify business requirements and preferences for the process along with the flow model in the process document. Then, a set of services, which matches the specified requirements and optimizes certain selected business objectives, will be automatically identified and specified in the final version of the process document, which will be passed to the workflow engine for the execution of the process.

In this section, we discuss issues involved with the specification of business requirements and objectives in process request documents: what are the types of requirements that need to be specified in request documents, how can BPEL4WS be used and extended for specifying the requirements, and how can the specified requirements be used in the discovery of services and the composition of processes? Also, we discuss a related topic, i.e., how to specify the capabilities of Web services using an existing Web service standard, most notably UDDI (Universal Description Discovery and Integration), an XML-based standard, which provides a registry of businesses and Web services [53].

There are studies, mostly from academia, done for the specification of service capabilities and process requests by using semantic knowledge-based markup languages, notably, OWL-S (OWL-based Web service ontology language) [42]. It is a service description language, providing a semantic view of Web services including the abstract description of the service capabilities. Based on the semantic representation of services and requests, several algorithms for matching published Web services and submitted service requests were proposed [10,43,44]. In this section, we propose to specify the business requirements and preferences in an extended format of the business process specification standard, i.e., BPEL4WS, instead of depending on OWL-S. We believe that it will facilitate the seamless integration of service discovery and workflow specification, which have been traditionally studied separately.

For matching published services and submitted process requests in terms of service capabilities and requested business requirements, we propose a system multiple matching algorithms, a *micro-level matching* algorithm, which matches the capabilities and attributes of published services with activities in a process request, and *macro-level matching* algorithms, which are used to compose a business process by selecting one service for each activity among the candidate services selected by the micro-level algorithm. The output from the macro-level matching algorithms satisfies the business requirements and constraints of the submitted request, and provides optimal solutions in terms of meeting a certain business objective, e.g., minimizing

the cost or execution time, or maximizing the total utility values of business-related properties of interest.

Some previous work envisioned the task of business process composition as an AI-inspired *planning* problem [10,46]. They represent a Web service by a rule that expresses the service capable of producing a particular output, given a certain input. Then, a rule-based expert system is used to automatically determine whether a desired composite process can be realized using existing services, and construct a plan that instantiates the process. We believe that this is an interesting approach in its own right, but we do not discuss this approach further in this section.

4.2 Business Requirement Specification

In this section, we address issues involved with the specification of business requirements and objectives in process request documents. We discuss what information on business requirements and preferences need to be specified in process request documents and how the information may be used in the discovery of services and the composition of processes. We extend the BPEL4WS specification to accommodate this information in business process documents. Instead of providing a complete schema for specifying the requirements in BPEL4WS documents, we simply provide a few motivational examples.

Business process documents written in BPEL4WS mostly consist of the following parts, which are primary components of BPEL4WS [23]:

- Process definition,
- Partner definition,
- Container definition,
- Flow model, and
- Fault handling.

The process definition allows the user to give a name to the current process, and to include the name spaces where service messages passed among the process activities are defined. The partner definition declares the parties involved in this process. The container definition declares the structures for holding messages among partners. The flow model specifies the activities and the interaction among them. Finally, the fault handling defines the joint handling of exceptions among services.

Note that the primary components of BPEL4WS do not allow companies who use the business process to specify any business requirement on the process such as the cost and the execution time limit of the process, and any preference such as one on partner relationships. We propose to extend BPEL4WS for specifying the

requirements of business process requests, which are important to composing optimal business process. The declarative specification of the business requirements in BPEL4WS (along with the specification of the attributes and capabilities of services in UDDI) facilitates the automatic service discovery and process composition. In the rest of this section, we will provide a few examples in a variant of BPEL4WS.

Figure 15 shows an example of specifying several requirements for a business process, i.e., cost, time and quality of services. Each requirement is presented with a certain limit value (either maximum or minimum) and a weight, which represents the relative importance of the requirement in selecting services for implementation. In Section 4.6, we will explain how this requirement information is used in selecting services for composing a business process with an algorithm for optimizing (e.g., minimizing or maximizing) certain business objectives, or a multi-attribute decision analysis algorithm that maximizes the total utility value of selected service combinations. Note that a set of business requirements such as given in Figure 15 can be assigned both to the entire process and individual activities used in the process. For simplicity, we will consider only the former cases in this section.

In addition to business requirements, users of business processes sometimes need to express their preferences in selecting Web services for implementing processes. An example is the preference regarding whom a company prefers (or does not prefer) partnering with in a business process depending on its existing business relationship with service providers. Examples of relationship among business entities include

```
<businessRequirements>
  <requirement name="processBudget"
               type="cost"
               value="30000.00"
               unit="USD"
               limit="maximum"
               weight="10" />
  <requirement name="processTime"
               type="time"
               value="365"
               unit="days"
               limit="maximum"
               weight="7" />
  <requirement name="processAvailability"
               type="quality"
               value="98.0"
               unit="%"
               limit="minimum"
               weight="5" />
</businessRequirements>
```

FIG. 15. Specification of business requirements in BPEL4WS.

```
<partnerRelationships>
  <relationship name="competitorNotAllowed"
                sourcePartner="approver"
                targetPartner="assessor"
                relationship="competitor"
                modifier="not" />
  <relationship name="partnersOnly"
                sourcePartner="approver"
                targetPartner="assessor"
                relationship="partner"
  <relationship name="notSame"
                sourcePartner="approver"
                targetPartner="assessor"
                relationship="same"
                modifier="not" />
</partnerRelationships>
```

FIG. 16. Specification of business relationship requirements in BPEL4WS.

business partners, competitors, and parent-child. In some cases, same two companies may have both partner and competitor relationships depending on products and/or services. Also, there may be non-traditional relationships among business entities depending on situations of the involved parties. Figure 16 shows examples of partner business relationships specified in a variation of BPEL4WS, where four preferred business relationships between partners, i.e., the service providers of two Web services, "approver" and "assessor."

We claim that the specification of the relationships among business entities is useful and necessary in creating business process solutions that fit involved partners' taste. To make use of the business entity relationships in selecting Web services, the information on the relationships of Web service providers also need to be specified in registries, i.e., UDDI, as well as the preference specified in the process request document in BPEL4WS. We will discuss this topic in the next section.

4.3 Service Profile Specification

In this section, we discuss the specification of the attributes and capabilities of Web services using an existing Web service standard, i.e., UDDI, so that the information can be used for the matching algorithms that select Web services in terms of matches between service capabilities and requested business requirements. Previous work in [44] provided a useful example of service profile schema ("upper ontology") comprising the actor of the service, the functionalities of the service expressed in terms of the transformation produced by the service, and a set of functional attributes providing additional information, requirements, and constraints. This work also provides a mapping of the service profile in OWL-S into UDDI records for performing se-

```
<add_publisherAssertions xmlns="urn:uddi-org:api_v3">
    <authInfo>FFFFF</authInfo>
    <publisherAssertion>
        <fromKey>Business A</fromKey>
        <toKey>Business B</toKey>
        <keyedReference
            tModelKey="uddi:uddi.org:relationships"
            keyName="Holding Company"
            keyValue="parent-child" />
</publisherAssertion>
    <publisherAssertion>
        <fromKey>Business A</fromKey>
        <toKey>Business C</toKey>
        <keyedReference
            tModelKey="uddi:uddi.org:relationships"
            keyName="Partner Company"
            keyValue="partner"
            keyCondition="product category 1" />
</publisherAssertion>
</add_publisherAssertions>
```

FIG. 17. Business relationship specification in UDDI.

mantic matching between services and requests. Our approach is based on this work in [44]. However, our approach does not depend on OWL-S specification of service profiles. Instead, we directly store semantic information of services in UDDI records by using tModels data structures, which describe additional features of services in UDDI.

In addition, our approach uses the publisherAssertion messages of UDDI specification (version 3) to specify relationships of business entities in UDDI records. Furthermore, we propose to extend the feature to express various business relationships (e.g., parent–child, partners, competitors, or any user-defined relationship), so that certain requirements on business relationships among partners can be expressed in process requests, if necessary, and met in Web service discovery. UDDI API for the publisher assertions can also be extended accordingly. Figure 17 shows an example of the specification of business relationships, i.e., parent–child and partnership, by using a publisherAssertion message. Note that the partnership of two business entities specified in this example depends on a condition, i.e., a product category.

4.4 Service Discovery with Micro-Level Matching

For the location of services for business activities based on service capabilities and attributes, we adopt an intuitive matching algorithm that uses semantic knowledge as well as other information retrieval filters. This algorithm returns a (pre-specified) number of services that *sufficiently* match with an activity in the request. It is based

on the previous work in [10,43] which allows service providers to advertise their services in OWL-S service profile markup, and match submitted requests again in OWL-S profile markup with appropriate services. Unlike this previous work, our work does not depend on OWL-S profile, but utilizes the specification of service capabilities and request requirements directly stored in UDDI records and BPEL4WS documents, respectively. This algorithm is referred to as a *micro-level matching* algorithm, because it mostly deals with a single atomic process of a request.

Figure 18 depicts the architecture of the micro-level matching algorithm. The Parser module is capable of parsing an input BPEL4WS document and creates objects storing business requirements specified in the documents. The Inference Engine module parses and reasons with ontologies that provide the working model of entities and interactions in knowledge domains of interest, specified in the OWL language [56]. The Capability Matching Engine interacts with the Inference Engine and the UDDI augmented by the service profiles that specify the capabilities and attributes of services. The engine generates a pre-selected number of non-dominated services matched with the target attributes in the input request. The output is stored in the Non-Dominated Match Vector. The matching process is customized by the specification of Match Criteria that is parsed by the Criteria Setup module. The Capability Matching Engine generates a Non-Dominated Match Vector for each Match Criteria.

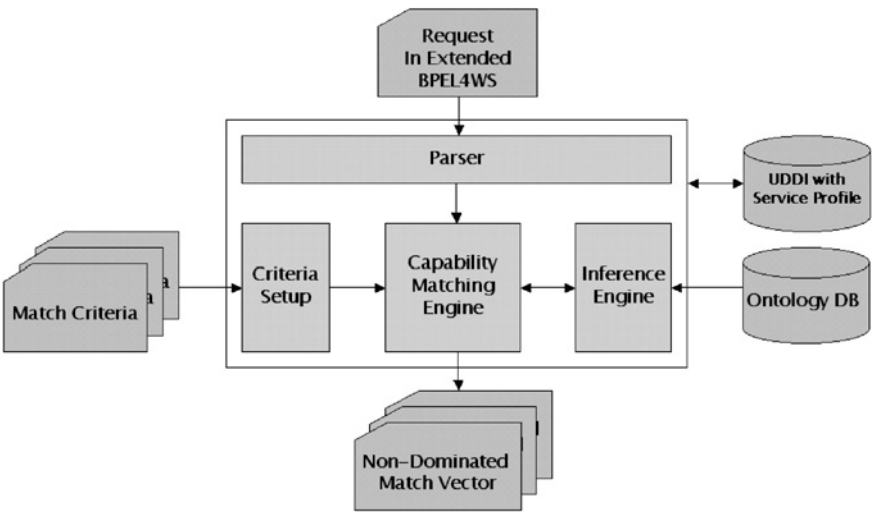


FIG. 18. Micro-level matching algorithm.

The Capability Matching Engine is based on the semantic matching algorithms outlined in [10,43]. While the matching algorithm presented in [43] is constrained to match only input and output messages of Web services, the algorithm proposed in [10] generalized the previous algorithm to match for any attribute of services and requests by parameterizing the match criteria such as quality, service categories as well as input and output messages. Figure 19 outlines the main control loop of the matching algorithm which is based on the work in [10]. The *degree of match* is a measure of the semantic distance between the conceptual meanings of the service attributes [10,43]. Each attribute has a lexical concept attached to it that is defined in the Ontology Database available to the Inference Engine. We use three different degrees of matches based on specialization relationship as defined in [43]. As given in the degreeOfMatch module of Fig. 19, the degrees of match are preferentially ordered based on the semantic distance that the degree represents: an EXACT match between concepts is preferred to a PLUG_IN match, and a PLUG_IN match is preferred over a SUBSUMES match [43].

As briefly mentioned above, the matching process is customized by the Match Criteria component, which specifies a set of target attributes for match and their least preferred degree of match, i.e., matchLimit. Any match on an attribute whose degree falls below matchLimit is considered a fail. An example of the Match Criteria is given in Table I.

```
matchAttribute(request, service, matchCriteria) {
    for each criteria in matchCriteria do {
          requestAttributes = request(attributeCriteria);
          serviceAttributes = service(attributeCriteria);
          for each requestAttribute in requestAttributes do {
            for each serviceAttribute in serviceAttributes do {
                degreeMatch = degreeOfMatch(requestAttribute, serviceAttribute);
                if (degreeMatch < matchLimit)
                    return fail;
                if (degreeMatch < globalDegreeMatch)
                    globalDegreeMatch = degreeMatch;
            }
        }
    }
    return success;
}

degreeOfMatch(requestAttribute, serviceAttribute) {
    if requestAttribute is SameClassAs serviceAttribute return EXACT;
    if serviceAttribute is SubClassOf requestAttribute return PLUG_IN;
    if requestAttribute is SubClassOf requestAttribute return SUBSUMES;
    else return FAIL;
}
```

FIG. 19. Capability matching algorithm.

TABLE I
MATCH CRITERIA

Attributes	matchLimit
Category	PLUG_IN
Input message	SUBSUMES
Output message	EXACT

4.5 Process Composition with Macro-Level Matching

The micro-matching algorithm works with other matching algorithms, *macro-level matching* algorithms, which are used to compose a business process by selecting one service for each activity in the request. The output from the macro-level matching algorithms satisfies the business requirements of the submitted request, and provides optimal solutions in terms of meeting a certain objective, e.g., minimizing the cost or execution time, or maximizing a certain quality measure. In this section, we model the macro-level matching problem as a variation of the *multiple-choice knapsack problem* [34], and design a configurable, generic optimization engine, which can be repeatedly run with variations of configuration criteria in search for a business process solution best fit the need. In addition, we alternatively model the macro-level matching problem as a *multi-attribute decision making problem*. This model is particularly useful when it is not sufficient to provide an optimal solution for a single measure, but requires maximizing the total utility value of multiple business measures of interest. Our algorithm is based on *multi-attribute decision analysis*, which computes the scores of the candidate service combinations by considering their attributes values and capabilities, ranks the candidates by score, and selects services among the top-rankers.

4.6 Multiple-Choice Knapsack Algorithm

Figure 20 displays the architecture of the macro-level matching algorithm. The input to the matching algorithm is a set of Non-Dominated Match Vectors, one vector for each atomic activity in the request, which were generated by the micro-level matching algorithm. The output of the optimization engine is a set of services selected from the input, one service from each Non-Dominated Match Vector. The match engine can be customized for different business objectives and constraints as specified in another input to the engine, the Configuration.

We model the macro-level matching problem as a variation of the *multiple-choice knapsack problem* [34]. The "multiple-choice" term in this problem designation refers to the requirement of selecting exactly one service from each candidate list,

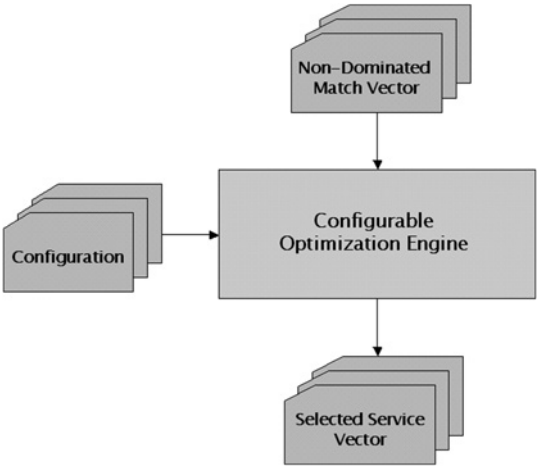


FIG. 20. Macro-level matching algorithm.

i.e., each Non-Dominated Match Vector. For a specific example, consider the following problem:

We are given a set of m business activities in our business process request, $a_1, \ldots, a_m$ such that activity, a_i, contains n_i candidates of Web services from the micro-level matching step. The jth candidate for activity a_i has cost c_{ij}, and execution time t_{ij}. Given the total execution time limit T for this business process, the goal of this macro-level matching algorithm is to compose an implementation plan for this business process by selecting one and only one Web service candidate from each candidate list such that the overall cost is minimized without exceeding our total execution time limit.

If we use indicator variable x_{ij} to indicate whether the jth service from the candidate list for activity a_i was selected, we can formalize the problem with the following equations:

$$\text{minimize} \quad C = \sum_{i=1}^{m} \sum_{j=1}^{n_i} c_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{m} \sum_{j=1}^{n_i} t_{ij} x_{ij} \leqslant T \tag{2}$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, \quad i = 1, \ldots, m, \tag{3}$$

$$x_{ij} \in \{0, 1\}, \ \forall i, j. \tag{4}$$

The multiple-choice knapsack problem is known to be NP-hard [34]. It is possible to exactly solve the above problems using branch-and-bound algorithms, but because the worst-case running time of these algorithms is exponential in both the number of activities and the number of candidates on each list, branch-and-bound algorithms are often too slow to be useful. An alternative approach is to use dynamic programming techniques, and there are a number of algorithms known in this direction [34]. By using off-the-shelf software packages of optimization algorithms such as IBM's OSL (Optimization Solutions and Library) [26], the given problem can be implemented in a straightforward manner.

With this model in place, we can vary the problem with different objective functions and constraints. The variation of the problem can be implemented by using the Configuration component in Fig. 20. For example, some processes may need to be optimized for execution time, while other measures such as cost will be treated as a constraint. In this case, the problem can be re-formulated as follows:

We are given a set of m business activities $a_1, \ldots, a_m$ such that activity a_i contains n_i candidates of Web services. The jth candidate for activity a_i has cost c_{ij}, and execution time t_{ij}. Given the total cost budget C for this business process, the goal of this algorithm is to compose an implementation plan for this business process by selecting one and only one Web service candidate from each candidate list such that the overall execution time is minimized without exceeding our total execution time limit.

If we use indicator variable x_{ij} to indicate whether the jth service from the candidate list for activity a_i was selected, we can formalize the problem with the following equations:

$$\text{minimize} \quad T = \sum_{i=1}^{m} \sum_{j=1}^{n_i} t_{ij} x_{ij} \tag{5}$$

$$\text{subject to} \quad \sum_{i=1}^{m} \sum_{j=1}^{n_i} c_{ij} x_{ij} \leqslant C, \tag{6}$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, \quad i = 1, \ldots, m, \tag{7}$$

$$x_{ij} \in \{0, 1\}, \ \forall i, j. \tag{8}$$

Yet another variation of this problem is an optimization on an interesting metric such as the degree of match described in the previous section. For example, the problem can be formulated as follows.

We are given a set of m business activities $a_1, \ldots, a_m$ such that activity a_i contains n_i candidates of Web services. The jth candidate for activity a_i has combined degree of match d_{ij}, cost c_{ij}, and execution time t_{ij}. Given the total cost budget C and the total execution time limit T for this business process, the goal of this algorithm is to compose an implementation plan for this business process by selecting one and only one Web service candidate from each candidate list such that our overall degree of match is maximized without exceeding our total cost budget and the total execution time limit.

If we use indicator variable x_{ij} to indicate whether the jth service from the candidate list for activity a_i was selected, we can formalize the problem with the following equations:

$$\text{maximize} \quad D = \sum_{i=1}^{m} \sum_{j=1}^{n_i} d_{ij} x_{ij} \tag{9}$$

$$\text{subject to} \quad \sum_{i=1}^{m} \sum_{j=1}^{n_i} c_{ij} x_{ij} \leqslant C, \tag{10}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} t_{ij} x_{ij} \leqslant T, \tag{11}$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, \quad i = 1, \ldots, m, \tag{12}$$

$$x_{ij} \in \{0, 1\}, \ \forall i, j. \tag{13}$$

Sometimes in a business process, the degree of match of an activity can be more important than those of other activities. In such a case, the variant importance of degree of match of different activities can be reflected in the model by the assignment of weight w_i for each a_i. Then the objective model is slightly modified as follows:

$$\text{maximize} \quad D = \sum_{i=1}^{m} w_i \sum_{j=1}^{n_i} d_{ij} x_{ij}. \tag{9'}$$

In addition to varied business objectives, the optimization problem of matching can be subject to *business rules* such as:

- the total number of service providers participating in a business process should be limited to a certain number to avoid depending too heavily on just a few partners,

- the total number of service providers participating in a process should be limited to a certain number to control the administrative overhead of managing providers,
- the number of services granted to a service provider should be limited to a certain number, and
- at least one (or some fixed number of) minority provider(s) need to be chosen.

These business rules can be expressed in linear inequalities with binary variables, and added as constraints to the multiple-choice knapsack problem model in a straightforward manner. The problem can be solved in the same manner for identifying the service set satisfying the given business objective and fulfilling the constraints of the given business rules.

4.7 Multi-Attribute Decision Analysis

Another approach to solving the macro-level matching problem is a *multi-attribute decision analysis*. This method is particularly useful when it is not sufficient to provide an optimal solution for a single measure, but requires maximizing the total utility value computed by considering multiple business measures such as cost, execution time, degree of match, quality, category, and business entity relationship. While this algorithm provides an alternative solution to the macro-matching problem, it can also be used with the optimization algorithm (of the multiple-choice knapsack problem model) in a complementary way.

The input to this algorithm is a set of n service combinations $s_1, \ldots, s_n$ such that service combination s_i contains m Web services, one service for each activity in the given business process. Also, each service combination has k business attributes $x_1, \ldots, x_k$ such that business attribute x_j is assigned a relative weight w_j (remember the weight attribute of the <requirement> tag in Fig. 15). Then this algorithm uses additive value function in order to compute the scores of the alternative service combinations. The system then ranks the alternative combinations by score, and selects the winning combinations among the top-rankers.

The basic hypothesis of this multi-attribute decision analysis algorithm is that in any decision problem, there exists a real valued function U defined along the set of feasible alternatives, which the decision maker wishes to maximize. This function aggregates the criteria $x_1, \ldots, x_k$. Besides, individual (single-measure) utility functions $U_1(x_1), \ldots, U_n(x_n)$ are assumed for the k different attributes. The utility function translates the value of an attribute into "utility units." The overall utility for an alternative is given by the sum of all weighted utilities of the attributes. For an outcome that has levels $x_1, \ldots, x_k$ on the k attributes, the overall utility for an

alternative i is given by

$$U(x_1, \ldots, x_k) = \sum_{i=1}^{k} w_i U(x_i). \tag{14}$$

The alternative with the largest overall utility is the most desirable under this rule. Each utility function $U(x_i)$ assigns values of 0 and 1 to the worst and best levels on that particular objective and

$$\sum_{i=1}^{k} w_i = 1, \quad w_i > 0. \tag{15}$$

Consequently, the additive utility function also assigns values of 0 and 1 to the worst and best conceivable outcomes, respectively. A basic precondition for the additive utility function is preferential independence of all attributes, which has been the topic of many debates on multi-attribute utility theory [7,11]. Even in cases with inter-dependencies, the additive utility function is often used as a rough-cut approximation for a more complex non-linear utility function.

4.8 Summary

In this section, we addressed two primary problems of the Web service-based business process automation: the location of services on the basis of the capabilities of published services, and the composition of business processes on the basis of the business requirements of submitted process requests. We proposed a solution, which comprises multiple matching algorithms, a *micro-level matching* algorithm and a *macro-level matching* algorithm. The first algorithm reasons with semantic information of services and returns services that *sufficiently* match with an activity in the request. The second algorithm solves a variation of the multiple-choice knapsack problem that models the macro-level matching problem for optimizing a business objective and fulfilling other business constraints. In addition, we proposed a *multi-attribute decision analysis* algorithm, which can be used with the optimization algorithm in a complementary fashion for a better process composition result. This algorithm is particularly useful when it requires maximizing the total utility value computed by taking multiple business measures into account. For securing information required for the execution of the matching algorithms, we explained how existing standards, UDDI and BPEL4WS, could be used and extended to specify service capabilities of services and business requirements, respectively.

ACKNOWLEDGEMENTS

The author thanks Ally Lu, Anca Ivan, Ankur Chandra, Chun Hua Tian, David Cohn, David Yao, Gordon Xie, Grace Lin, Guy Rackham, Ko-Yang Wang, Kumar Bhaskaran, Li Ma, Mei Gong, Rakesh Mohan, Rama Akkiraju, Richard Goodwin, Rob Guttman, Rong Zeng Cao, Rosemarie Truman, Thomas Li, Timur Nurullaev, Vivian Ding, Yahong Gu, Yue Pan, and Yunhee Jang for their constructive discussion.

Appendix A: Acronyms

API	Application Programming Interface
APQC	American Productivity & Quality Center, Inc.
BIRT	Business Intelligence and Reporting Tool
BPEL4WS	Business Process Execution Language for Web Services
CASE	Computer-Aided Software Engineering
CIM	Computation Independent Model
EMF	Eclipse Modeling Framework
EODM	EMF-based Ontology Definition Metamodel
ER	Entity Relationship
EVA	Economic Value Added
GEF	Graphical Editing Framework
IBM	International Business Machines Corp.
IBM OSL	IBM Optimization Solutions and Library
ISO	International Organization for Standardization
IT	Information Technology
KPI	Key Performance Indicator
MDA	Model-Driven Architecture
MOF	Meta-Object Facility
MVC	Model-View-Controller
ODA	Ontology Driven Architecture
ODM	Ontology Definition Metamodel
OWL	Web Ontology Language
OWL-QL	OWL Query Language
OWL-S	OWL-based Web service ontology language
PDE	Eclipse Plugin Development Environment
PIM	Platform Independent Model
PSM	Platform Specific Model
RDF	Resource Description Framework

RDQL	Query Language for RDF
ROI	Return On Investment
SOA	Service-Oriented Architecture
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WS	Web Service
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language Family
XSLT	XSL Transformations

REFERENCES

[1] APQC, "Process classification framework", http://www.apqc.org.
[2] Berners-Lee T., Hendler J., Lassila O., *The Semantic WEB*, Scientific American, 2001.
[3] Brickley D., Guha R., "RDF vocabulary description language 1.0: RDF schema", W3C Recommendation, http://www.w3.org/TR/rdf-schema/, 2004.
[4] Brown A., "An introduction to Model Driven Architecture – Part I: MDA and today's systems", http://106.ibm.com/developerworks/rational/library/3100.html, 2004.
[5] Burton-Jones A., *Knowledge Capitalism: Business, Work and Learning in the New Economy*, Oxford University Press, UK, 1999.
[6] Carroll J.J., Dickinson I., Dollin C., Reynolds D., Seaborne A., Wilkinson K., "Jena: Implementing the semantic Web recommendations", in: *Proc. of WWW*, 2004.
[7] Clemen R.T., *Making Hard Decisions: An Introduction to Decision Analysis*, Wadsworth Publishing Company, Belmont, CA, 1996.
[8] D2RQ V0.4: Treating non-RDF databases as virtual RDF graphs, http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rq/, 2004.
[9] Djuric D., Gaševic D., Devedžic V., "Ontology modeling and MDA", *J. Object Technology* **4** (1) (2005).
[10] Doshi P., Goodwin R., Akkiraju R., Roeder S., "A flexible parameterized semantic matching engine", IBM Research Report, 2002.
[11] Edwards W., "How to use multi-attribute utility measurement for social decision making", *IEEE Transactions on Systems, Man, and Cybernetics SMC* **7** (1977) 326–340.
[12] EMF (Eclipse Modeling Framework), http://www.eclipse.org/emf, 2004.
[13] Erickson J., Hughes L., "A disciplined approach to quantifying technology benefits", Forrester Report, December 13, 2004.
[14] Gartner, "Total cost of ownership", http://www.gartner.com/4_decision_tools/measurement/decision_tools/tco/tco.html.

[15] Gasevic D., Djuric D., Devedzic V., "Bridging MDA and OWL ontologies", *J. Web Engineering* **4** (2) (2005) 119–134.
[16] GEF (Graphical Editing Framework), http://www.eclipse.org/gef, 2004.
[17] Gliedman C., "Instilling value-based thinking, asking the 20 key questions", Forrester Report, June 29, 2004.
[18] Gliedman C., "Calculating the value of faster time-to-market", Forrester Report, February 19, 2004.
[19] Horn P., "The new discipline of services science", Business Week Online, http://www.businessweek.com/technology/content/jan2005/tc20050121_8020.htm, January 21, 2005.
[20] IBM, "Architecture of business", IBM Global Technology Outlook 2004, http://www.research.ibm.com, 2004.
[21] IBM, "Component business modeling", http://www-1.ibm.com/services/us/bcs/html/bcs_componentmodeling.html.
[22] IBM, "WebSphere business modeler", http://www-306.ibm.com/software/integration/wbimodeler/.
[23] IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems, "Business process execution language for Web services, version 1.1", http://www-128.ibm.com/developerworks/library/specification/ws-bpel/, February 2005.
[24] IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA, "Web services transactions specifications", http://www-128.ibm.com/developerworks/library/specification/ws-tx/, August 2005.
[25] IBM Integrated Ontology Toolkit, http://www.alphaworks.ibm.com/tech/semanticstk, 2005.
[26] IBM Optimization Solutions and Library, http://www-3.ibm.com/software/data/bi/osl/index.html.
[27] Jastor, http://jastor.sourceforge.net/.
[28] Kaplan R.S., Norton D.P., *The Balanced Scorecard: Translating Strategy into Action*, Harvard Business School Press, September, 1996.
[29] Klyne G., Carroll J., "Resource Description Framework (RDF): Concepts and abstract syntax", W3C Recommendation, http://www.w3.org/TR/rdf-concepts/, 2004.
[30] Lee J., "Model-driven business transformation and semantic Web", *Commun. ACM (special issue on semantic eBusiness vision)* (December 2005).
[31] Lee J., Goodwin R., "Ontology management for large-scale enterprise systems", *J. Electronic Commerce Research and Applications* **5** (3) (2006).
[32] Leymann F., Roller D., Schmidt M.T., "Web services and business process management", *IBM Systems J.* **41** (2) (2002).
[33] Ma L., Su Z., Pan Y., Zhang L., Liu T., "RStar: An RDF storage and query system for enterprise resource management", in: *Proc. of ACM CIKM*, 2004, pp. 484–491.
[34] Martello S., Toth P., *Knapsack Problems*, John Wiley & Sons, Chichester, 1990.
[35] McDavid D., "The business-IT gap: A key challenge", IBM Research Memo, http://www.almaden.ibm.com/coevolution/pdf/mcdavid.pdf.
[36] MOF: Meta-Object Facility, Version 1.4, http://www.omg.org/technology/documents/formal/mof.htm.

[37] Oberle D., Volz R., Motik B., Staab S., "An extensible ontology software environment", in: Staab S., Studer R. (Eds.), *Handbook on Ontologies*, Springer, 2004, pp. 311–333, (Chapter III).
[38] ODM: Ontology Definition Metamodel, http://www.omg.org/docs/ontology/04-08-01.pdf, 2004.
[39] OECD, "A new economy? The changing role of innovation and information technology in growth", Paris, 2000.
[40] OMG, "Model Driven Architecture (MDA) guide, version 1.0.1", http://www.omg.org/cgi-bin/doc?omg/03-06-01, March 6, 2001.
[41] OMG, "Unified Modeling Language (UML)", www.omg.org/uml/.
[42] OWL Services Coalition, OWL-S: Semantic markup for Web services, http://www.daml.org/services/owl-s/1.0/owl-s.html.
[43] Paolucci M., Kawamura T., Payne T.R., Sycara K., "Semantic matching of Web services capabilities", in: *Proc. of the 1st International Semantic Web Conference*, June 2002.
[44] Paolucci M., Kawamura T., Payne T.R., Sycara K., "Importing the semantic Web in UDDI", in: *Workshop on Web Services, e-Business, and the Semantic Web: Foundations, Models, Architecture, Engineering and Applications, Toronto, Ontario, Canada*, May 2002.
[45] Pellet: An Open-Source Java Based OWL DL Reasoner, http://www.mindswap.org/2003/pellet/index.shtml.
[46] Ponnekanti S.R., Fox A., "SWORD: A developer toolkit for web service composition", in: *Proc. of the 11th World Wide Web Conference, Honolulu, Hawaii*, May 7–11, 2002.
[47] Protégé, http://protege.stanford.edu/index.html, 2004.
[48] Sesame, "An open source RDF database with support for RDF schema inferencing and querying", http://www.openrdf.org/, 2002.
[49] Smith M.K., Welty C., McGuinness D.L., "OWL Web ontology language guide", http://www.w3.org/TR/owl-guide/, 2004.
[50] Stern Stewart & Co, "What is EVA?", http://www.sternstewart.com/evaabout/whatis.php.
[51] Symons C., "The balanced scorecard for IT: Value metrics", Forrester Report, November 15, 2004.
[52] Tetlow P., et al., "Ontology driven architectures and potential uses of the semantic Web in systems and software engineering", http://www.w3.org/2001/sw/BestPractices/SE/ODA/, 2005.
[53] UDDI Version 3 Published Specification, http://www.uddi.org/pubs/uddi-v3.00-published-20020719.htm, 19 July 2002.
[54] W3C, "Resource Description Framework (RDF)", http://www.w3.org/RDF/.
[55] W3C, "Semantic Web", W3C Document, http://www.w3.org/2001/sw/.
[56] W3C, "Web ontology language (OWL)", http://www.w3.org/2004/OWL/.
[57] W3C, "Web services", http://w3c.org/2002/ws/.
[58] W3C, "Web services architecture", http://www.w3.org/TR/ws-arch/.

This page intentionally left blank

Phish Phactors: Offensive and Defensive Strategies

HAL BERGHEL

School of Informatics and Internet Forensics Laboratory
University of Nevada, Las Vegas
USA

Department of Computer Science and Software Engineering
University of Canterbury
Christchurch
New Zealand

JAMES CARPINTER

Department of Computer Science and Software Engineering
University of Canterbury
Christchurch
New Zealand

JU-YEON JO

School of Informatics and Internet Forensics Laboratory
University of Nevada, Las Vegas
USA

Abstract

Phishing attacks attempt to fraudulently solicit sensitive information from a user by masquerading as a known trustworthy agent. They commonly use spoofed emails in association with fake websites in order to coerce a user into revealing personal financial data. Phishing is now a serious problem with criminals adopting the well-developed and well-known techniques to exploit Internet users with sophisticated attacks. Phishers are known to have successfully attacked an estimated 1.2 million users and stolen an estimated US$929 million in the twelve months to May 2005.

ADVANCES IN COMPUTERS, VOL. 70

ISSN: 0065-2458/DOI: 10.1016/S0065-2458(06)70005-5

This chapter aims to provide the current status of phishing attack techniques and defense methods. We first provide an overview of the fundamental phishing techniques for delivering a successful attack, such as bulk emailing, fake websites and detection avoidance using a variety of obfuscation techniques. We then survey more sophisticated methods that may deceive even knowledgeable and vigilant users. These techniques do not rely on naïve email users and simple websites, but use highly realistic fake websites, generic hacking techniques (such as DNS poisoning or cross site scripting) or actively exploit browser vulnerabilities. For example, a Man-In-The-Middle attack or the use of DNS poisoning can easily fool even an advanced user who may be aware of phishing attacks.

Quite a few defensive methods have been developed, although many are still in the early stage of development. URL obfuscation can be rather reliably detected using analysis algorithms. Fake websites can also be detected automatically with a low false positive ratio by comparing them with the real websites. Clients can utilize anti-phishing-capable devices or software such as anti-virus, anti-spam, anti-spyware, or IDS. Web browsers can be armed with anti-phishing plug-ins such as Spoofstick or SpoofGuard. Given the damage that can potentially done by a phishing attack, a diverse range of efforts are being made to protect ordinary users (such as in user education, reporting and response and legal protection).

The outlook is not entirely bleak against phishing given the technical and social remedies being pursued. If organizations prepare well, remain vigilant and follow attack trends carefully, they can respond quickly and effectively with a range of techniques to defend their customers' data. If individuals take a responsibility for their protection and adopt a defense-in-depth approach, they can shield themselves against the most sophisticated attacks. Although there is no simple solution, active and aware users and organizations have the ability to form a strangle-hold on this ever-growing threat.

1. Introduction

Phishing attacks attempt to fraudulently solicit sensitive information from a user by masquerading as a known trustworthy agent [5,60]. They most commonly use 'spoofed' emails in association with fake websites in order to coerce a user into revealing personal financial data, such as credit card numbers, account user names and passwords, or social security numbers [49]. By masquerading as well-known banks, e-retailers and credit card companies, phishers often convince recipients to respond [5]. Phishing attacks range in sophistication, from simply fooling a user with a seemingly legitimate communication, to deliberately exploiting weaknesses in software to prevent users from determining the true nature of the attack.

The idea of obtaining user information through fraudulent means is not unique; phishing is merely a subset of two larger problems that exist in both the electronic and 'real-world' domain:

- Social engineering: is any attempt to obtain confidential information by manipulating legitimate users. Phishing uses email and counterfeit websites to achieve this goal [61]. While most Internet security threats take advantage of software vulnerabilities, this attack exploits trust relationships previously developed[1] between the user and other users or organizations.

[1] In some circumstances, the trust relationship is created and then immediately abused. For example, an attacker might attempt to reset login credentials from an organization's helpdesk by convincing the tech-

- Identity theft: uses the information gained through techniques such as social engineering in a deliberate attempt to use another person's identity. This can then be used, for example, to gain access to their finances or frame them for a crime [59]. Techniques used involved include stealing mail, rummaging through garbage ('dumpster diving'), stealing personal information from computer databases, or infiltrating large organizations that store large amounts of information. Phishing is merely a mechanism of obtaining this information.

Phishing shares many characteristics with two similar techniques: pharming and the abuse of alternate data streams. Both require a higher level of skill to execute successfully than simple phishing schemes. Pharming is a more active form of phishing, with the user automatically directed away from the legitimate website to the fraudulent website without warning [9]. Alternate data streams can be used to secretly associate hostile executables with legitimate files; this is effectively 'file phishing' [8]. With minimal effort, a hidden executable can be masked and its function obscured. Like phishing, the resulting environment is not entirely as it appears.

Another related technique is an independent scam website that lures victims through voluntary web navigation or through a search engine instead of using active emailing. An unsuspecting user may buy a product from a scam website, or make an investment on a foreign company through a scam website. While its effect is similar to phishing, the process of luring the victims is different. A scam website is a passive form of phishing, silently waiting for a prey, but it can become more effective when supplemented with phishing techniques. Many anti-phishing techniques covered in this chapter are also useful for identifying those independent scam websites, for example, Trustbar (see Section 4.4) displays the logos and certificate authority of the website.

1.1 History

The word 'phishing' is a derivative of the word 'fishing' and describes the process of using lures to 'fish' for (i.e., obtain) sensitive user information [2]. Exchanging 'f' for 'ph' is a common hacker replacement; it is most likely an acknowledgement of the original term for hacking, known as 'phreaking'. The original form of hacking, known as phone phreaking, involved sending specific tones along a phone line that allowed users to manipulate phone switches. This allowed free long distance calls, or the billing of services to other accounts, etc.

The first recorded use of the term 'phishing' was in January 1996, in a posting to the alt.2600 newsgroup by drspamcake@aol.com. It was in reference to the theft

nician that they are a legitimate user in some kind of unusual situation that requires standard procedures to be bypassed.

of AOL user accounts [13] by scamming passwords off unsuspecting users. The technique itself predates the reference by drspamcake@aol.com: AOL users were already being targeted via instant messages sent by users masquerading as AOL staff members, who would request a user's account details [60]. By 1995, AOL software contained a 'report password solicitation' button, which gives an indication of the magnitude of the threat.

Those seeking free AOL accounts initially took advantage of poor credit card validation techniques and used algorithmically generated credit card numbers to acquire accounts that could last up to a month. They turned to phishing for legitimate accounts after AOL bought in measures in 1995 to prevent this type of behavior. Hacked accounts were referred to as 'phish' and by 1997, phish were being actively traded as a form of electronic currency [46]. For example, phish could be traded for hacker software or 'warez'.

Since that time, the definition of phishing has widened to cover not only obtaining user account details, but also obtaining access to all personal and financial data. The sophistication of the field has also grown: modern schemes go far beyond simple instant messages, and typically target thousands of users using mass mailings and fake websites.

1.2 Current Status

Phishing is now more than a mere annoyance: it is a common online crime that is relatively easy to perform, has a low chance of being caught, and has a potentially very high reward [27]. It is for these reasons that phishing has been embraced by organized crime, both in the United States and in Eastern Europe (particularly in Russia and the former Soviet bloc). It is also believed [23] that terrorist sympathizers, operating out of Africa and the Middle East, are using phishing to steal identities and cash.

Phishers typically send out massive emails in the hope that some naïve recipients will respond. Although the majority of the recipients feel suspicious on such phishing emails, some recipients are successfully convinced into the scam. Phishers successfully attacked an estimated 1.2 million users and cost an estimated US$929 million in the twelve months to May 2005 [38]. US businesses lose an estimated $2 billion a year as their clients become victims [39]. The Anti-Phishing Working Group[2] [3]

[2] Anti-Phishing Working Group is a global association of industrial and law enforcement organizations focused on eliminating the fraud and identity theft that result from phishing, pharming, and email spoofing. It includes more than 1,600 companies and agencies worldwide including 8 of the top 10 US banks and 4 of the top 5 US ISPs. It offers anti-phishing education, maintains phishing data, evaluates the anti-phishing methods, and work with law enforcement and legislature. Its website is available at http://www.antiphishing.org.

received 20,109 reports of phishing scams in May 2006, primarily targeting financial institutions (92% of all reports). In year to May 2006, the average growth rate for phishing attacks was 34% [3].

1.3 Phishing Illustrated

There are several steps that phishers follow. Two examples are illustrated here, for posers and mongers, respectively. The posers are the bottom-feeders in the phishing community that exhibit a very low level of sophistication. The phish mongers are those who deploy these phish scams in such a way that they stand a measurable chance of success against a reasonably intelligent and enlightened end-user [7].

1.3.1 Posers

The essential requirements of effective phishing require that the bait:

1. look real;
2. present itself to an appropriate target-of-opportunity;
3. satisfy the reasonableness condition (i.e., going after the bait is not an unreasonable thing to do);
4. cause the unwary to suspend any disbelief;
5. clean up after the catch.

Figure 1 is modeled after some live phish captured on the net and meets all of the five criteria identified above. First, the email looks real—at least to the extent that it betrays nothing suspicious to a typical bank customer (a.k.a. target-of-opportunity). The graphic appears to be a reasonable facsimile of a familiar logo, and the salutation and letter is what we might expect in this context. Second, the target is the subset of recipients who are Bank of America customers. The fact that the majority of recipients are not is not a deterrent because there is no penalty for over-phishing in the Internet waters. Third, the request seems entirely reasonable and appropriate given the justification. Customers reason that if they were a bank, they might do the same thing. Fourth, the URL-link seems to be appropriate to the brand. Unwary Internet users might readily trade off any lingering disbelief for the opportunity to correct what might be a simple error that could adversely affect use of a checking or credit card account. The link to "verify.bofa.com" may be assumed to take us to an equally plausible web form that would request an account name and password or PIN.

The unwary in this case is M. Jones whose harvested web form appears to the phisherman as in Fig. 2. This is a screenshot of an actual phishing server in our lab.

In order to complete the scam the fifth condition must apply. In this case, after the private information is harvested, the circle is completed when the phishing server

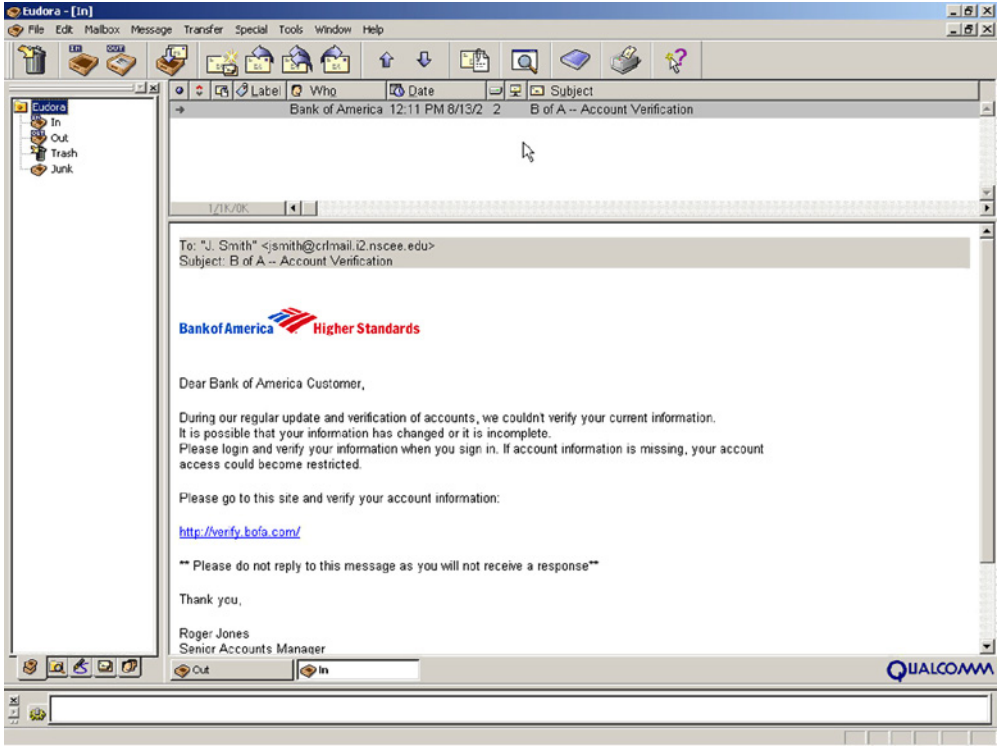


FIG. 1. Phishing email that satisfies our five effectiveness criteria.[3]

[3] Any legitimate emails from Bank of America (as well as most other corporate names used in this chapter) to a customer never put a reply URL in the message. Instead they ask the customers to go to their corporate website directly, avoiding a direct response to what may be a fraudulent email.

FIG. 2. Phishing from phisherman's perspective.

redirects the victim to the actual bank site. This has the effect of keeping the bank's server logs roughly in line in case someone makes an inquiry of the help desk. Figure 3 illustrates this activity.

1.3.2 Mongers

Mongers employ more sophisticated schemes. Look carefully at the cursor in Fig. 4. The cursor seems to be sensing the link even though it is not particularly close to it. The fact is that it is not sensing the link at all, but rather an image map.

A quick review of the source code, below, leads us to a veritable cornucopia of trickery.

```
<x-html>
<html><p><font face="Arial"><A
HREF="https://signin.ebay.com/ws/eBayISAPI.dll?SignIn&sid=verify&co_p
artnerId=2&siteid=0"><map name="xlhjiwb"><area coords="0, 0, 646,
569" shape="rect" href="http://218.1.XXX.YYY/.../e3b/"></map>
<img SRC="cid:part1.04050500.04030901@support_id_314202457@ebay.com"
border="0" usemap="#xlhjiwb"></A></a></font></p>
<p><font color="#FFFFF3">Barbie Harley Davidson in 1803 in 1951 AVI
</x-html>
```

Several features make it interesting. First, the image map coordinates take up nearly the whole page. Second, the image that is mapped is the actual text of the email. So what appeared to be email was just a picture of email. Thus, the redirect was actually not a secure connection to eBay at all as it appeared, but an insecure connection to `218.1.XXX.YYY/.../e3b/`. While Windows users see the "dots of laziness" frequently when path expression is too long for the path pane in some window, this is not a Windows path in a path pane. These "dots of laziness" are a directory name. It is not clear why someone would create a directory named ". . ." as

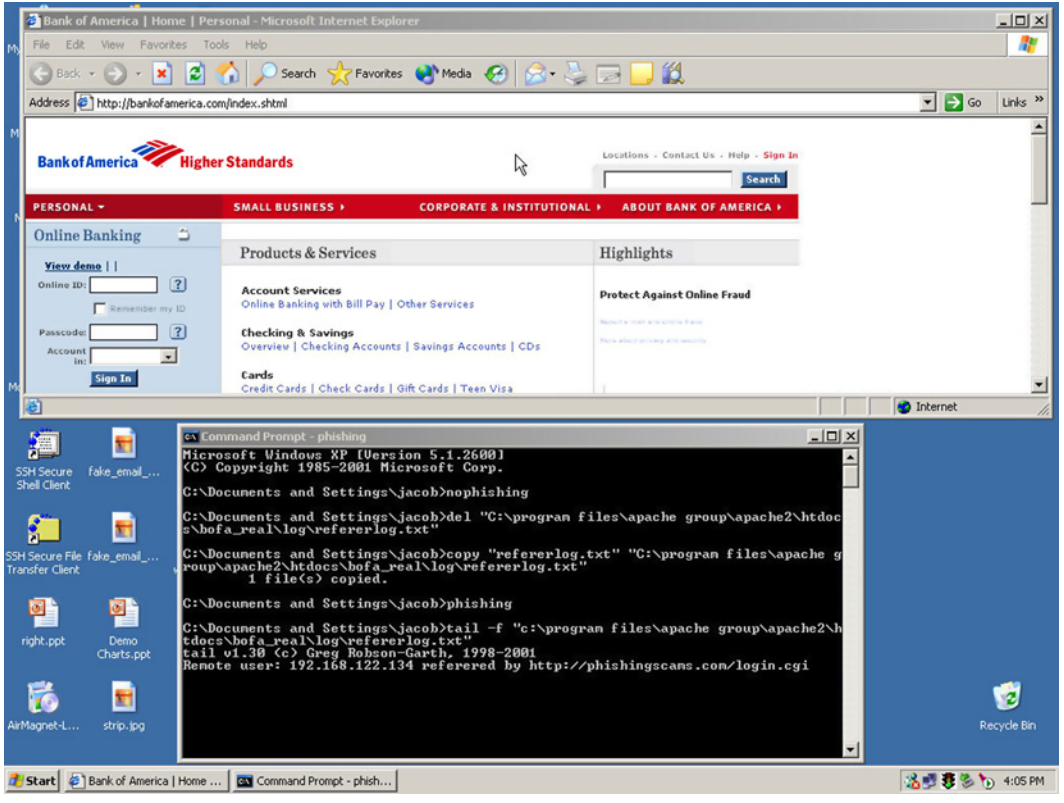
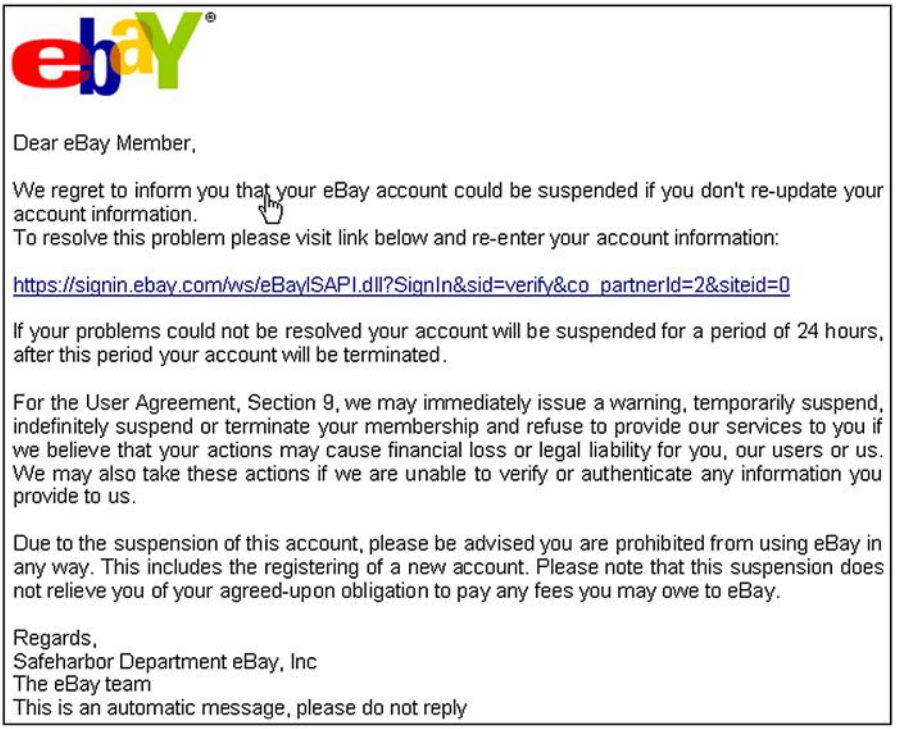


FIG. 3. Phish clean-up.



Dear eBay Member,

We regret to inform you that your eBay account could be suspended if you don't re-update your account information.
To resolve this problem please visit link below and re-enter your account information:

https://signin.ebay.com/ws/eBayISAPI.dll?SignIn&sid=verify&co_partnerId=2&siteid=0

If your problems could not be resolved your account will be suspended for a period of 24 hours, after this period your account will be terminated.

For the User Agreement, Section 9, we may immediately issue a warning, temporarily suspend, indefinitely suspend or terminate your membership and refuse to provide our services to you if we believe that your actions may cause financial loss or legal liability for you, our users or us. We may also take these actions if we are unable to verify or authenticate any information you provide to us.

Due to the suspension of this account, please be advised you are prohibited from using eBay in any way. This includes the registering of a new account. Please note that this suspension does not relieve you of your agreed-upon obligation to pay any fees you may owe to eBay.

Regards,
Safeharbor Department eBay, Inc
The eBay team
This is an automatic message, please do not reply

FIG. 4. Phish mongering.

it certainly falls short of the mnemonic requirements most of us learned in intro to programming.

On the other hand, it might blend in stealthily with the other *nix hidden files, "." and "..", and possibly escape an onlooker's suspicion. This suggests that the computer at the end of `218.1.XXX.YYY` may not be the phisher at all, but another unsuspecting victim whose computer has been compromised (for that reason, the final two octets of the IP address have been concealed). Another sign of intrigue is the font color of almost pure white "`#FFFFF3`" for "Barbie Harley Davidson in 1803 in 1951 AVI." Though their names are sullied, neither Barbie nor Harley Davidson had anything to do with this scam. This white-on-white hidden text is there to throw off the Bayesian analyzers in spam filters. As the email text is actually a graphic, the Bayesian analysis likely concludes that this is about Barbie and her Harley given that it has no other text to base its decision on. As opposed to the posers, this phish monger is moderately clever.

So far the most common of phishing attacks have been illustrated. The rest of this chapter is organized as follows. In Section 2, the fundamental techniques used in phishing are explained, such as bulk emailing, alternative delivery techniques, and obfuscation techniques for masking the fake websites. Section 3 addresses advanced phishing techniques, including Malware, man-in-the-middle attack, and website-based attacks, etc. In Section 4, anti-phishing techniques are discussed; however, technical solutions are only part of the picture in anti-phishing efforts and in Section 5, more comprehensive efforts are examined. The chapter is then concluded in Section 6.

2. Core Phishing Techniques

In order to achieve their goals, phishers typically use a mixture of two techniques: social engineering and technical subterfuge [5]. Social engineering is the primary technique used and appears to some extent in most attacks. Arguably, the use of this technique distinguishes phishing from other forms of electronic fraud. Technical subterfuge exploits software-based weaknesses in both servers and clients in order to mask the true nature of the transaction from the victim or plants crimeware onto PCs to steal credentials directly (often using Trojan keylogger spyware). Pharming crimeware misdirects users to fraudulent sites or proxy servers, typically through DNS hijacking or poisoning [5].

Both techniques are employed in the pursuit of the same goal: the victim must be convinced to perform a series of steps to reveal confidential data. However, these two techniques seek to attack from opposite ends of the spectrum: one targets human weaknesses, the other technical vulnerabilities. In this section and the next, how phishers exploit these vulnerabilities is examined.

2.1 Bulk Emailing Combined with Fake Websites

A basic phishing scheme needs four elements: a bulk mailing tool, a standard email, a ghost (fake) website and a database of email addresses [20,49]. Typically, the ghost website is set up, and then the bulk email tool distributes the phishing email to all those addresses in the email database. The most successful phishing scams have genuine looking content in both their e-mail (if used) and the fake website. This includes:

- using images from the real website;
- in the e-mail, use official text (social engineering techniques apply here);
- many phishing sites simply copy the real website (using `wget` or the like).

The standard email sent is branded so it appears as though it was sent by a trusted and reputable party (e.g., a financial institution). The most commonly spoofed companies include Citibank, eBay and PayPal. It is likely that not all those users within the email database will have accounts with the spoofed organization; this is somewhat unavoidable and it can reveal the operation of a phishing attack. A number of techniques can be used within the email [20,34,46,49]:

- The email will use authentic logos and graphics obtained from the legitimate website in order to imitate the company's visible branding (see Fig. 5). The email itself is likely to be a modified copy of a previous corporate mailing.
- It will also use a spoofed 'mail from' address to make the email appear to originate from the proper domain. This is a well-known flaw in the SMTP protocol: phishers can set the 'mail from' and 'reply to' headers to an email address of their choice.
- It will typically contain a URL that appears to link to the legitimate site; however, the URL will likely relay the user to the ghost website (the URL in Fig. 5 sends the user to `http://218.246.224.203/`). This obfuscation will generally require the use of HTML email. The use of HTML formatting also allows the attacker to create a more authentic email by using legitimate graphics. This would allow the attacker to include a HTML form inside the email itself to solicit user information, although this is relatively uncommon.
- Using an HTML email to imitate a plain-text email can further increase the difficultly an average user faces in identifying the hidden qualities of the email (see Fig. 5).
- The email is also likely to contain URLs that refer the user to the legitimate site (for example, to a help or contact page) in order to better mimic a mailing from the legitimate organization (see Fig. 5).
- Assurances are included within the email to gain trust, such as "we will not ask you for sensitive personal information. . . in an email" or the use of the **TRUSTe** symbol (which identifies organizations with a high level of personal information protection). Other security assurances are also used. For example, that the email is free of viruses and is not spam.
- The HTML code behind the email is usually long, which limits the ordinary user's ability to locate and check the actual target of the URL contained in the email (if they decide to do so).

The key objective of the email is to create a plausible premise that persuades the user to release personal information. The contents of the email must be designed to illicit an immediate user reaction, which prompts them to follow the enclosed link to the website (see Fig. 5). For example, the email may [20]:

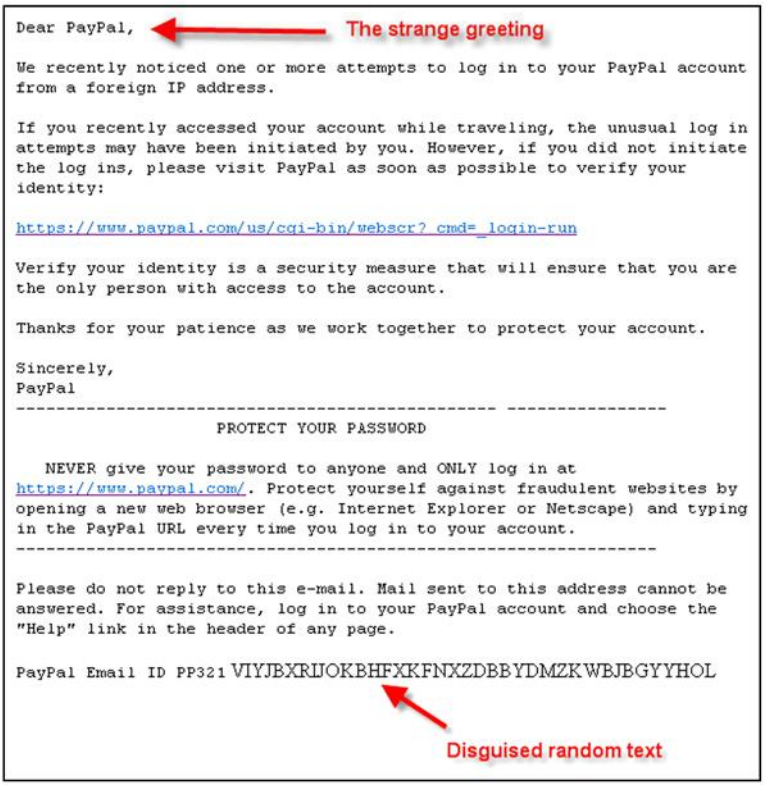


FIG. 5. An example phishing email recorded by the Anti-Phishing Working Group (10/05/2005).

- refer to an unauthorized transaction from the victim's bank account,
- reveal the user has won a prize,
- indicate the organization has lost their account details, requiring the user to manually update them, or
- threaten to charge a fee without an immediate reply.

Ironically, many phishing emails take advantage of the user's fear of online fraud [20], using a premise that requires users to update their information due to a security system upgrade or similar (see Fig. 5). While there are many different approaches, each must create a scenario to convince the user to provide the requested information in a timely manner (i.e., before the phishing site is shutdown).

Using a bulk email tool and an email database, the standard email can be sent to millions of legitimate, active email addresses within a few hours. The use of a network of trojaned machines can speed the process up considerably [34,46]. The email database can be acquired from a number of sources on the Internet, for free or for a fee. Such vendors target their email databases at spam distributors; however, the databases they distribute are equally useful for both purposes.

A ghost, or fake, website is typically hosted by a hijacked PC,[4] compromised by other means [34,46,49]. A mechanism will need to be in place to facilitate information retrieval (by the phisher); it is speculated anonymous login or email may be used for this activity. Ideally, this machine would reside in a different country to that of legitimate website's organization as this increases the difficulties involved in shutting the phishing website down. The domain name and email URL are designed to prevent the target from noticing they are transacting with a ghost website rather than the legitimate site. Subtle character replacements can achieve this: for example, `www.paypal.com` could be imitated using `www.paypa1.com` (note the "one" in the name) or `www.paypal.cc`. More sophisticated methods will be discussed in Section 2.3 (URL obfuscation).

The content of the website is likely to be a near-exact copy of the legitimate site, updated to allow the phisher to record user details. The ghost website is also likely to contain introduction pages, processing pages and pages thanking the user for submitting their data, in a further attempt to increase authenticity. It may also use a legitimate server-side certificate, signed by Verisign or similar, issued to the ghost website. Alternatively, it could use an unsigned certificate under the assumption that most users will be unable to interpret the security warning (if the security warning has not already been disabled). Even invalid or fake certificates are likely to make users feel more secure. The absence of SSL/TLS security may alert some users to the true nature of the website; however, security indicators within the user's browser can potentially be forged using browser exploits (see Fig. 6).

Upon submitting their details to the ghost website, the user is often redirected to the legitimate site to encourage the user to continue to believe they have revealed their personal data to a legitimate organization. Alternatively, the phisher may use a post-submission page to encourage the user not to access or use their accounts for a specific timeframe, in order to mask the phisher's exploitation of their sensitive information (e.g., use of a credit card number). It is critical that the user does not realize they have submitted their data to an illegitimate organization. If this occurs, the personal data can be quickly rendered useless (e.g., their password will be changed or accounts closed).

[4] This can sometimes be detected by the use of a non-standard HTTP port embedded in the target URL.

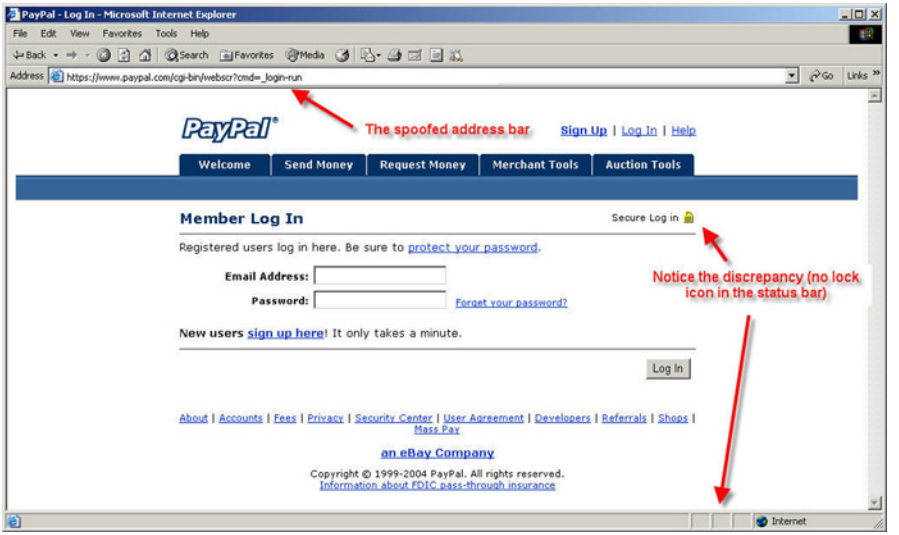


FIG. 6. A phishing website targeted at PayPal customers from the Anti-Phishing Working Group.

2.2 Alternative Delivery Techniques

Communication via email remains the most common and successful form of attack; however, other electronic communication mechanisms are becoming increasingly popular, such as web pages, IRC and instant messaging [46]. In all forms of communication, the phisher must imitate a trusted source in order for the victim to release their information.

2.2.1 Web-based Delivery

Rather than distributing the malicious URL (or similar) via email, an increasingly popular technique is to place it in website content [46]. The website itself can be hosted by the phisher, or by a third party host (which could be acquired freely, for a fee, or via a Trojan horse attack). The level of sophistication varies: a malicious URL could be disguised and placed on a popular website or comment board, or a website developed for the express purpose of luring in potential victims.

If specialist website is employed to lure victims, the phisher may employ several techniques [46]:

- Hidden items within the page (e.g., tiny graphics) may be used to identify suitable victims.
- Pop-up or frameless website may be used to hide the true source of the website.
- Malicious content may be embedded to exploit a vulnerability of the user's browser. This exploit can be leveraged to install software onto the user's computer without their knowledge. For example, software such as key loggers, screen grabbers, back doors or other trojan horse programs may be installed.
- Trust relationships inside the user's browser configuration may be abused in order to allow scriptable components or access data storage areas.

In order to attract users to their website, fake banner advertising could be employed. Banner images belonging to the company the phisher is attempting to mimic can be placed on popular websites and direct users to the phisher's website, rather than the legitimate website. Standard URL obfuscation techniques can be used to hide this subtle redirection from the user. Many vendors provide online registration for banner advertising; with a stolen credit card (or similar), a phisher can easily acquire advertising while remaining concealed from law enforcement agencies.

2.2.2 IRC and Instant Messaging (IM)

While these techniques were popular in the early days of phishing, email has become the technique of choice for modern day phishers. However, it is predicted [27, 46] that the use of these techniques will become more popular in the future, given that these technologies are popular with home users and are gaining in their complexity on a regular basis. Embedded dynamic content, such as multimedia, graphics, and URLs, can now be sent with many IM programs, allowing standard email and web-based phishing techniques to be easily mapped to this domain. Automated bots, that interact unsupervised with IRC participants, could also be used by phishers to coerce users into visiting their fake websites.

2.3 Obfuscation Techniques

In addition to the techniques previously mentioned, phishers have other techniques to deliberately disguise the true nature of the message from the recipient.

URL obfuscation is an essential part of most phishing attacks. It fools the user into believing they are following a link to a legitimate website; in actual fact, they are being transported to the phisher's fake website. The simplest technique for URL obfuscation uses HTML; the legitimate website's address is displayed to the user in plain text, but the link is targeted at the phisher's website. For example:

```
<a href=''http://www.evilsite.com''>http://www.citibank.com</a>
```

is displayed to the user as:

http://www.citibank.com

but links to `http://www.evilsite.com` (see Fig. 5). This technique would fool a basic user, who many not be aware than the display address of the hyperlink can be different to its target. Other URL obfuscation techniques include [20,46,49]:

- Simple character replacement can obfuscate URLs, as using the legitimate URL as a prefix to another domain (e.g., adding bank.com to `www.citibank.com` to form `www.citibank.com.bank.com`). Variations of the legitimate domain name can also be used (e.g., `www.citibank-accounts.com`). All of these simple techniques would go unnoticed by an inexperienced user.
- An extension of the aforementioned technique involves the vulnerabilities of the ASCII character set. Foreign characters are encoded using 2-byte Unicode rather than 1-byte ASCII. Attackers can utilize visually similar characters in different Unicode sets to exploit confusion, which is called Unicode attack [26]. Domain names can be registered in different languages: some foreign character sets look identical to ASCII characters, but are interpreted differently during the domain name lookup process. According to Fu et al. [26], there are eight possible representations of alphabet character "s", "o", "u", "p" and so on. Users may not be able to distinguish the differences at a quick glance. A recent scam allowed `microsoft.com` to be registered, using the Cyrillic 'o' instead of the ASCII version; visually these characters are identical.
- Most browsers also accept alternative encoding schemes for hostnames, in order to allow support for local languages.
  - Escape encoding allows the inclusion of characters that may need special syntax in order to be correctly interpreted (e.g., a space in a URL string may indicate the end of the URL or it may be part of the URL). These are included as %xx, where xx is the hexadecimal ASCII code for the character. This also allows normal characters to be encoded in this way (e.g., %41 is 'A' and %20 is a space).
  - Unicode encoding allows characters to be stored in multiple bytes. This permits a far greater number of characters (65,536) that can be encoded in comparison with ASCII (128), and allows a unique identifier for every character no matter what language or platform. In a Microsoft Windows environment, these characters can be encoded as %u0000, where 0000 is the hexadecimal code for the character (e.g., %u0056 is 'V').
  - UTF-8 encoding is a commonly used format of Unicode, and preserves the full ASCII character code range. This allows standard characters to be en-

coded (and obfuscated) in longer escape-coded sequences (for example, '.' can be encoded as %F8%80%80%80%AE).

- Multiple encoding can occur when applications incorrectly parse escape-encoded data multiple times and at multiple layers of the application. This vulnerability can be exploited by phishers encoding characters multiple times and in different fashions (e.g., %%35%63: the second part of the string, '%35%63', decodes to '5C'. This string, combined with the prefix '%', gives '%5C', which decodes to '\').

- The standardized URL encoding format allows for the insertion of a username and password within the string (e.g., `http://username:password@mysite.com`). Effectively, everything between the protocol name and the '@' character is ignored; this allows the construction of obfuscated URLs such as `http://citibank.com:mybank@fakesite.cc`. Due to the threat this encoding format presented, some browsers no longer allow links of this form (such as Microsoft Internet Explorer).
- Some online websites provide redirection URLs: these allow the construction of URLs that give no indication of the actual target. Redirection URLs forward users onto another predefined site when they are accessed. For example, the link `http://r.aol.com/cgi/redir?http://jne9rrfj4.CjB.neT/?uudzQYRgY1GNEn` was found in a Citibank phishing attack, and includes a double redirect. The browser is first sent to `r.aol.com`, and then redirected to `jne9rrfj4.cjb.net`, which redirects the user to the fake website.
- The website host name can be obfuscated by encoding it as an IP address rather than a domain name. The use of a standard decimal IP address in place of the host name will go provide some measure of obfuscation; however, encoding the IP address in dword (e.g., `http://3532038435`), octal (e.g., `http://0322.0206.0241.0043`), hexadecimal (e.g., `http://0xD2.0x86.0xA1.0x23`) or a mixed format (e.g., `http://0322.0x86.161.0043`) will confuse even more users.

Given the bulk nature of these emails, and the threat they pose to users, most organizations opt to treat them as spam, and filter them before they reach users. Several techniques can be used by phishers to make the filtering task more difficult, and therefore reach more potential victims [46]:

- Text can be obfuscated to avoid spam filter detection. For example, lower-case 'L's could be replaced with upper-case 'I's, both of which appear visually similar to humans, but are interpreted quite differently by software. The similarity between the letter "`l`" and the numeric "`1`" may also be exploited as in `www.ao1.com` instead of `www.aol.com`.

- Where possible, the phisher may seek to personalize the email to the intended user, or at least make it unique (e.g., by inserting random text. See Fig. 5 for an example of this). This will largely depend on the email database used. By ensuring each email is unique will make it more difficult for the email to be filtered by hash-based anti-spam techniques (e.g., Cloudmark [14]).
- The use of HTML email allows the spammer to hide random words within the email (see Fig. 7). They can be included as comments or colored white to avoid detection by the user. These hidden words can make the email seem more legitimate to the spam filter, without altering the message eventually viewed by the user.

JavaScript can be used by the phisher to execute a number of attacks. In terms of obfuscation, it can be used to further hide the true destination of the link from the

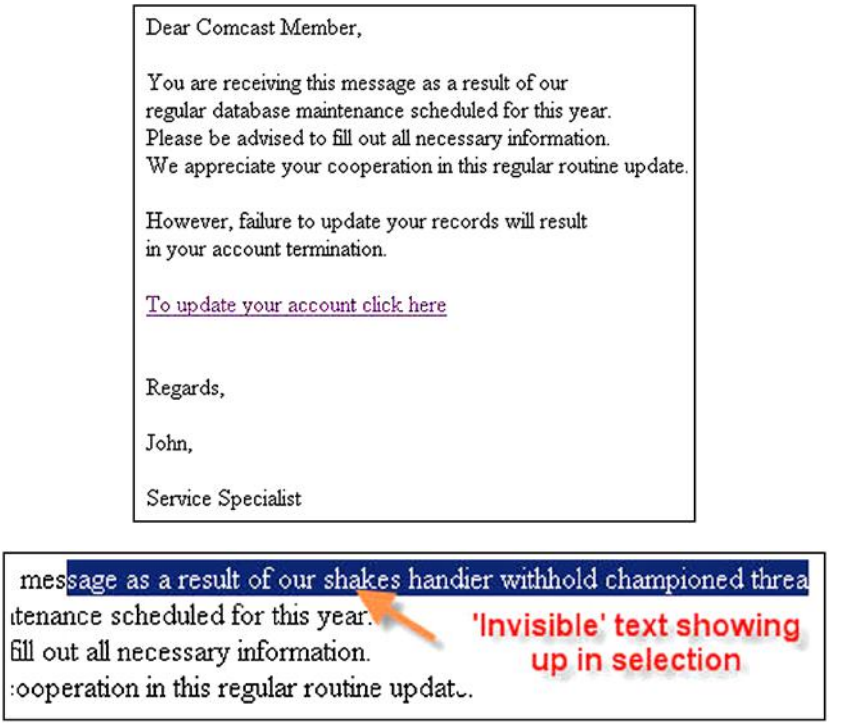


FIG. 7. An example from the Anti-Phishing Working Group that illustrates the use of hidden text in order to avoid detection by spam filters.

user. Most email and browser applications show the true destination of the link in the status bar at the bottom of the windows when a user moves their mouse over the link; this behavior can be overridden, so the legitimate link is displayed rather than the link to the fake site.

Many of the obfuscation techniques and the attack techniques discussed here can be readily identified if the user is sophisticated enough to read and interpret the source code. JavaScript can be used to deter the user from doing so as it can disable the page's right click menu. Those users who right click can instead by greeted with a pop-up message box (with a copyright notice or similar). However, this does not prevent the user access the page source from alternative locations (such as through the browser's menu bar, if it is present).

3. Advanced Phishing Techniques

3.1 Malware

Malware is a term used to describe any form of malicious software, including viruses, worms, trojans and others. For phishers, this software represents a new route to defraud their victims that may complement or even replace the social engineering techniques that phishing often relies upon. The potential for fraud here is greater: rather than asking the victim for their information, they simply take it. The complete replacement of social engineering in a phishing attack with malware arguably represents an entirely different class of attack. However, much malware still relies on the targeted user to approve its installation and/or execution; it is for this reason social engineering is likely to remain a core skill relied upon by phishers.

Known malware worms used for phishing include [42]:

- W32.Mimail.I,J,P,Q,S: these worms attempt to fool users into revealing credit card information in response to a Microsoft Windows expiration notification or a PayPal application. The requests are displayed as web pages served from the local machine. Typically, the worm is attached to an email message (passive worm), and a social engineering approach is used to encourage the user to run the attachment. The worm is then copied to the local drive, where it embeds itself in the machine's startup routine. It is self-propagating: it searches the user's documents for email addresses, and sends itself to all found addresses. Some variants also attempt to retrieve other information (Internet account information, RAS phone book entries, E-Gold information, and other personal information such as the user's credit card numbers, their birthday and their social security number) and relay it to the phisher using HTTP POST/GET.

- Backdoor.Lala and Backdoor.Lala.B: are trojan horses that permit unauthorized access to remote computers. They also attempt to steal confidential information (such as cached passwords and cookies), log keystrokes, and allow remote file execution. Cookies associated with financial institutions, such as PayPal, E-Bullion, Evocash, WebMoney and various banks, are targeted.
- PWSteal.Bancos and W32.Bibrog: are worms designed to monitor websites visited by the user. If the user attempts to visit particular bank websites in their browser, the worm redirects them to a phishing site. This website records and steals the user's information. It replicates by sending itself to all email addresses in the user's Outlook contacts folder.
- More recently, the Korgo [56] worm was used to infect unpatched systems with a keylogging trojan designed to steal online banking information and securely relay it back to its creators. It collected any data entered into a web form by the user. Mikko Hypponen, of F-Secure, advised users infected by Korgo to change all their passwords and to cancel their credit cards. "This is not a joke," he said.
- Other trojans and worms are known to record keystrokes or record data entered in web form input fields (e.g., W32.Dumuru.Y, W32.Dumuru.Z, PWSteal.Tarno, PWSteal.Banpaes, PWSteal.Banpaes.B, the TROJ_WINCAP series and W32.Mimail.C).

Interestingly, Brazilian banks appear to be over-represented in malware-based phishing schemes. In the six months to March 2004, twenty different malware applications were identified that targeted Brazilian banks [42].

Malicious users have long used software designed to log keystrokes and record screen captures to obtain sensitive data. These utilities are being employed more frequently in phishing attacks. These utilities can remain on a user's computer for an indefinite amount of time, and can record a far greater amount of information than any one basic phishing attack. Depending on the extent to which the attacker is willing to analyze the recorded logs, account information from a variety of sites can be harvested (rather than a single account typically recorded by a standard phishing scheme). Given the volume of data these techniques can potentially generate, the attacker has three options to retrieve recorded information:

- Data streaming: data is sent to the attacker as soon as it is generated. This requires a continuous connection between the attacker and victim.
- Batch collection: information is uploaded to the attacker's server on a regular basis, using FTP, HTTP, SMTP or similar.
- Backdoor collection: remote access software is placed on the victim's computer to allow the attacker to connect and download the recordings on demand.

Key loggers record all keystrokes entered by a user. With the use of appropriate filtering techniques, the attacker can isolate credentials used to access various online services. They vary in sophistication: some will record all key presses, while other will only record key presses entered in the web browser. The Anti-Phishing Working Group [3] recorded 215 phishing attacks in May 2006 (out of a total of 20,109) that used key logging malware.

Screen capture utilities were, in part, a response to advanced anti-key logging techniques used by some organizations; they record the other primary form of user input. These utilities record a screen image on a regular basis, or part of a screen image (i.e., the relevant observational area, such as the authentication area of a particular website). Partial screen captures minimize the size of the require upload to the attacker. Such techniques are successful against organizations such as Barclays Bank; they require users to select several, randomly selected, characters from their 'memorable word' from drop-down lists (e.g., the second and fourth letter) as part of their technique.

Phishers and spammers typically share some commonalities: both typically want to distribute substantial amounts of email quickly and without being detected. There is some evidence [49] that these groups are exchanging techniques. Some techniques applicable to phishing are [42]:

- Spam relays: are machines that accept email and forward it on to another SMTP server. The trojan horse Backdoor.Hogle turns unsuspecting machines into spam relays. By 'recruiting' a number of spam relay machines, the phisher could send messages quicker and make it more difficult for authorities to trace the message's origin. It is estimated that phishers can use up to 1,000 computers in their attacks [29].
- Reverse HTTP proxies: are used to hide the true location of the web server. The domain name included in the mass email message is configured to point to the IP address of a machine running the reverse HTTP proxy (such as a machine infected with the Backdoor.Migmaf trojan horse). The machine proxies any HTTP requests back to the actual web server, and sends any responses back to the client; at no time does the client know the IP address of the actual web server. Additionally, the IP address that the hostname points to is changed on a regular basis, making locating and neutralizing the actual web server particularly difficult. The Backdoor.Migmaf trojan has been used in a PayPal phishing scam.

3.2 Man-in-the-middle Attacks

The principles behind man-in-the-middle attacks are simple: the attacker acts as an intermediary between the victim and the legitimate site and records the information

exchanged between the two parties [46]. The attacker achieves an ideal vantage point on the transaction, and can potentially remain unnoticed by both parties. The idea behind this attack is not unique to this domain: it is used throughout network security (e.g., TCP hijacking).

The intermediary machine utilized by the attacker is referred to as the proxy. Ideally, it is transparent: it does not effect the communication between the legitimate parties, and is not easily detected. Such proxies can be located on the same network segment as the target, or en route to the legitimate website. To ensure the client routes traffic through the proxy, browser proxy settings can be overridden (either by a software exploit, or through the use of social engineering); however, this is now obvious to the client. Proxy configuration is generally performed before the phishing email message is sent: this ensures the transmitted data is recorded if the user follows the enclosed link.

This form of attack can be successful for both HTTP and HTTPS (i.e., SSL/TLS) connections [46]. SSL/TLS provides application-level security between the client and the legitimate website; standard proxies between these two parties can only record the cipher text. However, if the phishing email can ensure the user connects to the proxy, rather than legitimate website, their data can be recorded. URL obfuscation techniques are useful in achieving this. The proxy passes all of the user's requests to the legitimate website, and responses from the legitimate website are passed back to the user. In the case of a SSL/TLS connection, a secure connection is established between the proxy and the legitimate website. A secure connection can also be maintained between user and the proxy via the methods described previously.

Using the legitimate website to process information submitted by the victim also aids the phisher as it allows invalid data to be discarded. It not only makes the phisher's job of identifying valid accounts easier but it also makes the site appear more authentic to the user [20].

DNS cache poisoning [46] attempts to corrupt the local cache maintained by a specific DNS server. When a user requests the IP address of a domain name, the request is forwarded to the DNS server. If the DNS server does not have the IP address of the domain in its cache, it will query an authoritative domain name server for the information. The BIND attack, an example of DNS cache poisoning, requires the attacker to spoof the reply from the authoritative name server; in the reply, the attacker can set the IP address of the queried domain to any desired machine. By exploiting DNS vulnerabilities, the phisher could potentially redirect traffic directed at a site such as www.citibank.com to their fake website. DNS cache poisoning can be particularly effective, as most ISPs operate one DNS server for all of their subscribers. If the network's DNS server is poisoned, all of the ISP's customers will be redirected to the fake website.

3.3 Website-based Exploitation

After the user has been successfully lured to the fake website, the phisher has a variety of technologies to further disguise and obfuscate the true identity and nature of the website. Website scripting and markup languages such as HTML, JavaScript, DHTML, ActiveX, VBScript, etc. give the phisher tremendous power to completely mimic the appearance of the legitimate website [46].

HTML frames can be used to obscure attack content. They enjoy wide browser support and are simple to use, and therefore are ideal for phishing websites. For example:

```
<frameset rows="100%,*", framespacing="0">
  <frame name="real" src="http://www.citibank.com" scrolling="auto">
  <frame name="hidden" src="http://fakesite.com" scrolling="auto">
</frameset>
```

The legitimate Citibank site is all that is viewable within the browser window; however, this code snippet also loads HTML from fakesite.com. The additional code could [46]:

- deliver additional material, such as overriding page content or graphics,
- retrieve session IDs,
- execute screen captures, log keystrokes or monitor user behavior in the real website,
- provide a fake HTTPS wrapper that would force the browser to display the SSL/TLS padlock (or other security indicator),
- prevent the user from viewing the HTML source code,
- load images and HTML code in the background for later use, or
- imitate the functionality of the browser toolbar (if it is overlaid with a graphical representation in order to hide the actual location of the content) in combination with client-side scripting software.

Hidden frames can also hide the address of the phisher's content server. Only the URL of the document containing the frameset will be accessible from the browser interface (e.g., from the location bar or the page properties dialog).

The use of DHTML allows the phisher to override the content of the legitimate site, effectively building a new site on top of the real page [46]. The DIV tag allows content to be placed within a virtual container, which can then be given an absolute position within the document. It can be positioned to obscure existing content with careful positioning. JavaScript can be used to dynamically generate the content. For example:

```
var d = document;
d.write('<DIV id="fake", style="position:absolute; left:200;
    top:200; z-index:2">
    TABLE width=500 height=1000 cellspacing=0
    cellpadding=14><TR>');
d.write('<TD colspan=2 bgcolor=\#FFFFFF valign=top  height=125');
```

This particular example uses JavaScript to generate the first few lines required to construct a DIV that will be positioned to obscure existing website content.

Users are increasingly aware of the visual clues that mark a secure and legitimate site [46]. For example: the https identifier at the beginning of the URL, the URL itself, the zone of the page source (e.g., My Computer, Trusted, Internet, etc.), and the padlock icon somewhere in the browser (indicating secure SSL/TLS communication). These visual clues can sometimes be difficult to mimic using traditional techniques; however, specially created graphics can be loaded and positioned over specific areas of the browser 'chrome' (the window frame, menus, toolbars, scroll bars and other widgets that comprise the browser user interface) using scripting languages.

For graphical substitution to be successful, the graphics must be consistent with the browser. It is trivial to detect the browser the user is using[5]; from this information, the correct graphics can be overlaid. Areas of interest for graphical overlays include [46]:

- location bar: altered to report the legitimate URL, rather than of the fake site (see Fig. 6);
- SSL/TLS indicator: a padlock is overlaid in the correct location to (falsely) indicate a secure connection;
- certificate details: fake details are displayed if a user reviews page properties or security settings, and
- zone settings (Microsoft Internet Explorer): this can be altered from "Restricted" or "Internet" to "Trusted."

The release of Microsoft Windows XP SP2 prevented Internet Explorer from being susceptible to some of the techniques for achieving these overlays, and other browser makers are following suit [43]. Alternatively, the location bar can be spoofed with JavaScript by [20]:

- Closing the actual location bar, and replacing it with a table. The first row of the table will contain the address bar (as an image), and the second row of the table will contain the rest of the page.

[5] This also allows the phishing scam to only focus on the users that use browsers with specific security vulnerabilities or that use browsers with specific functionality.

- Opening a small browser window that contains a white box with the legitimate website address inside; this window is then positioned over the browser's location bar.

Furthermore, the attacker can use JavaScript to create popup windows that display supplementary content. On arriving at the fake website, a phishing popup is created while the main browser is redirected to the legitimate site. This gives increased credibility to the popup window [20] (see Figs. 8 and 9).

Unlike Internet Explorer and other browsers, Mozilla and Firefox do not compile their graphical user interface into the browser itself. Instead, it is stored as XUL: XML User Interface Language. The XUL data for these browsers is readily available, and can be rendered inside the browser's content area. This could potentially allow a phisher to perfectly mimic the appearance of the browser, but allow them to arbitrarily set the location bar text or SSL/TLS padlock [43].

JavaScript can also be used to hijack inconspicuous events generated by the browser [43]. File upload controls can be embedded as form elements in website in order for phishers to retrieve specific files from the user; however, these elements cannot have default values. By attaching an event handler to the OnDragStart event (an Internet Explorer extension), the upload control can be appropriately populated if the user drags their mouse. Ensuring they do so is a task left up to social engineering. On the conclusion of the drag event, the form can be automatically submitted, along with the stolen file. Several attacks are known that work on the same basic principle, some of which are no longer possible after certain Microsoft security updates. Other exploits of this technique include inserting, and then activating, active content in a user's Favorites folder, inserting executable files into a user's start up folder, or installing a backdoor trojan (more specifically, installing Backdoor.Sokeven).

3.4 Server-side Exploits

Any discussion of the exploitation of server-side vulnerabilities to assist in a phishing attack quickly transcends phishing and enters the realm of general hacking and cracking; this would be somewhat beyond the scope of this chapter (see [34] for some additional details). Suffice to say there are numerous techniques for exploiting operating systems, applications and network protocols that a phisher could use if they were determined to comprise a legitimate website in order to conduct a phishing attack. However, two 'non-invasive' techniques of relevance to phishers will be discussed: cross site scripting and preset sessions [46].

Cross site scripting (CSS or XSS) seeks to inject custom URLs or code into a web-based application data field, and takes advantage of poorly developed systems [27]. Three techniques are typically used [46]:

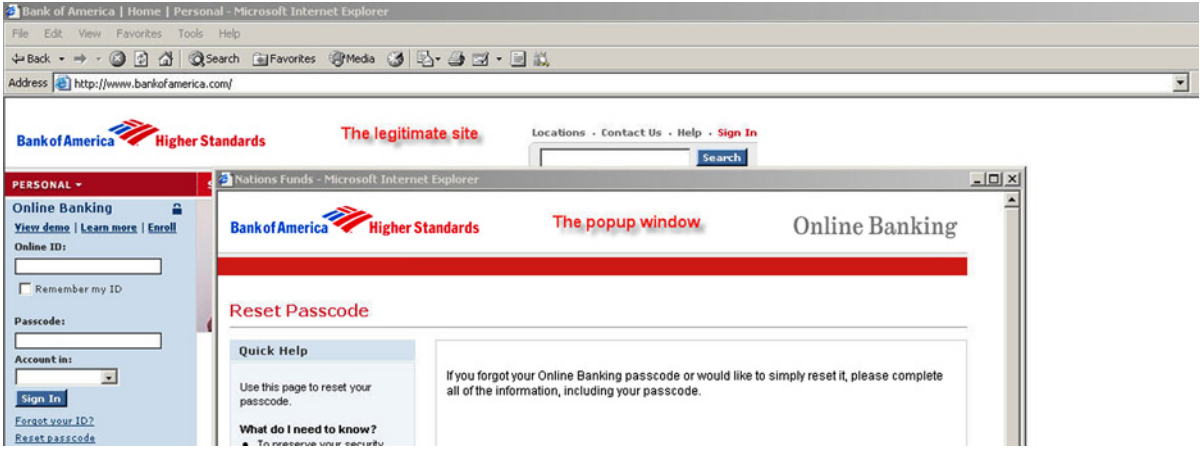


FIG. 8. This illustrates the use of a pop-up window over the legitimate site in the hopes of increasing the scheme's credibility. Obtained from the Anti-Phishing Working Group.

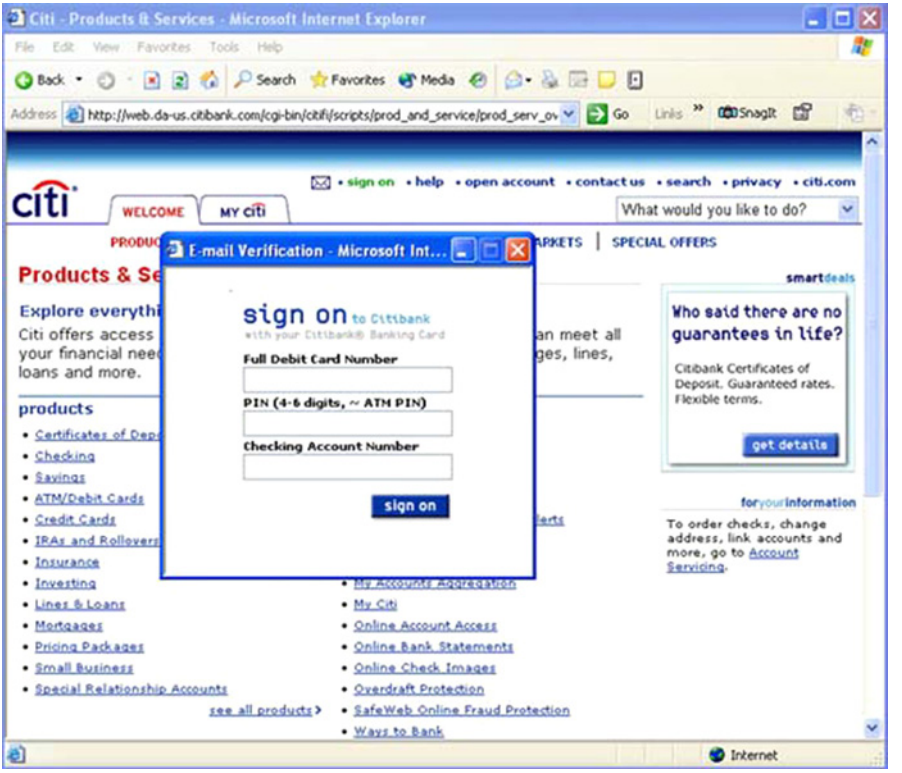


FIG. 9. This shows another pop-up window over a legitimate webpage. Using scripts, it opens up the real webpage and then opens a bare window popup asking for information.

- HTML substitution:

  `http://www.citibank.com/ebanking?URL=fakesite.com/login.htm`

  In this example, the standard legitimate website content is rendered, but the web application uses a parameter to identify where to load specific page content (for example the login box); in this case, that content is fetched from fakesite.com (whose URL could be obfuscated using previously described techniques).
- Forced loading of external scripts:

  ```
  http://www.citibank.com/ebanking?page=1\&response=fakesite.com
  \%21secretScript.js\&go=2
  ```

  In this example, a script to be executed is passed to the web application.
- Inline embedding of active content:

```
http://www.citibank.com/ebanking?page=1\&client=<SCRIPT>...
</SCRIPT>
```

In this example, the script is placed in the URL and executed by the web application.

Preset sessions use session identifiers. Session identifiers are typically used in HTTP and HTTPS transactions as a mechanism for tracking users through the website and to manage access to restricted resources (i.e., manage state). Session IDs can be implemented in a variety of ways; for example, cookies, hidden HTML fields or URL parameters. Most web applications allow the client to define the session ID. This allows the phisher to embed a session ID within the URL (that refers to the legitimate server) sent as part of the initial email [46]. For example,

```
https://mybank.com/ebanking?session=3V1L5e5510N
```

Once the email is sent, the phisher polls the legitimate server with the predefined session ID; once the user authenticates against the given session ID, the phisher will have access to all restricted content.

3.5 Client-side Vulnerabilities

Any discussion of client-side vulnerabilities is similar to that of its server-side counterpart: there are a multitude of vulnerabilities that a smart phisher could take advantage of in order to execute arbitrary code or to manipulate the browser. Given their exposure to the Internet, it is not surprising browsers suffer from a significant number of security vulnerabilities. Most browsers also support a number of plug-ins, each of which carries its own security risks. While patches are typically available in a timely manner, home users are notoriously poor at applying them quickly; therefore, phishers have ample time to exploit most security vulnerabilities, if they choose to do so.

Some past exploits used by phishers include [42,46]:

- Microsoft Internet Explorer URL mishandling: a URL such as:
  The real URL: `http://www.citibank.com%01@fakesite.com/phisher.html`
  What the user sees: `http://www.citibank.com`
  Where the browser goes: `http://fakesite.com/phisher.html`
  By inserting a %01 string in the username portion of the URL, the location bar displays `http://www.citibank.com`, while redirecting the user to `fakesite.com`. Earthlink, Citibank and PayPal were all targeted using this particular flaw.

- Microsoft Internet Explorer and Windows Media Player combination: this vulnerability allowed the execution of a Java JAR archive, disguised as a Windows Media Player skin, which could access local files.
- RealPlayer heap corruption: RealPlayer is available as a plug-in for most browsers, and allows the user to view the proprietary RealMedia format. By creating a malformed RealMedia file, and embedding it in a website to ensure it is automatically played, it is possible to cause a heap corruption, which would allow the execution of arbitrary code.

While malware can often be eliminated with a regularly updated antivirus utility, browser (or any client-side) exploits cannot be defended against until a patch is available and applied.

3.6 Context Aware Attacks

Context aware attacks [37] manipulate the victim into readily accepting the authenticity of any phishing emails they may receive. The first phase, which may involve interaction with the victim, will be innocuous and not request any sensitive information. Rather, the goal here is to ensure the victim will expect the message sent in the second phase. The second phase marks the dispatch of the actual phishing email; however, the email is expected by the victim, and therefore more likely to be considered authentic. The actions suggested in the phishing email would often arouse suspicion in the victim if viewed in isolation, but the preset context allows this to be avoided. Jakobsson [37] presented a context aware phishing scenario to 25 users, and recorded a 46% success ratio.

A simple example of a context aware attack involves targeting an eBay seller [37] (also see Fig. 10). Firstly, a seller is located who has an active auction and accepts payments via PayPal (but preferably not by credit card). At the end of the auction, a spoofed message is sent by the phisher from PayPal, indicating the successful buyer has paid for the goods won at the auction, but using a credit card (which the seller does not support). The email gives the seller two choices: either reject the payment, or upgrade their account to support credit card transactions; both these options require the seller to log into their account. By embedding an obfuscated URL to a fake website within the email, the phisher can easily record the seller's credentials. In this situation, the seller was expecting an email confirming payment; therefore, the spoofed email is expected, and is therefore viewed with less skepticism.

3.7 Empirical Results

Dhamija and Tygar [17] characterize the most common successful techniques employed by phishers. They reviewed the phishing attacks archived by the Anti-

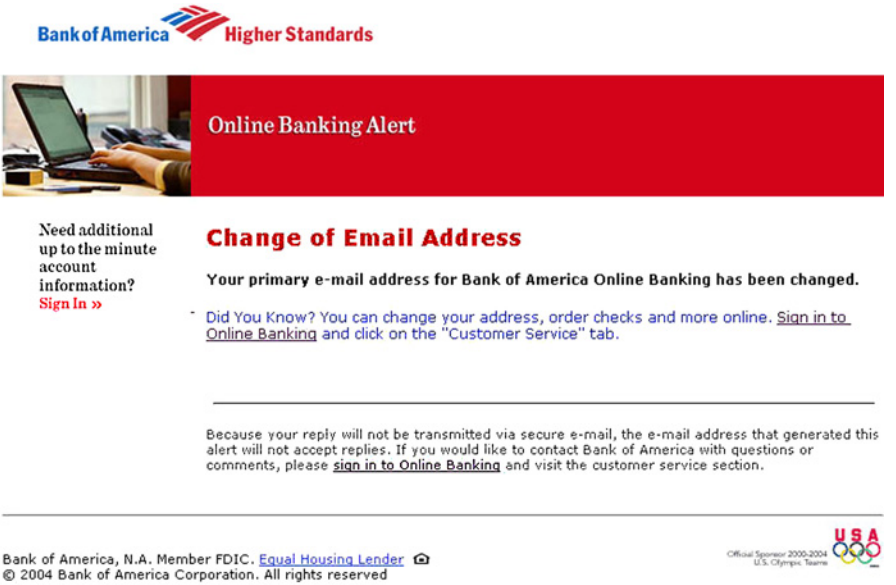


FIG. 10. This email is particularly well done, and illustrates a context-aware attack. On arriving at the site, the user is presented with a pop-up over the legitimate site, which gives the user the option of changing account details. It is not coercive and therefore not suspicious. They accept its legitimacy, as they require the ability to change their details. Obtained from the Anti-Phishing Working Group.

Phishing Working Group [4] over a period from September 2003 to mid 2005. Their findings were consistent with what is known about phishing: these attacks exploit human tendencies to trust certain brands and logos and that many phishing schemes prey on the widespread sense that the Internet is unsafe and that users must take the steps suggested by the attacker to ensure the security of their data. Furthermore, they concluded that the effectiveness of phishing schemes is raised when users cannot reliably verify security indicators. Unfortunately, this often the case, as browsers have generally not been designed with security usability in mind. More specifically, they identified the following phishing techniques as particularly serious:

- spoofed sender email addresses cannot be reliably detected,
- mimicked websites, with the same 'look and feel' as the legitimate site, cannot be reliably identified,
- obfuscated domain names are often undetected,
- images of URLs cannot be reliably distinguished from actual URLs,
- browser chrome cannot be reliably distinguished from web page content,

- images of legitimate security indicators (e.g., the padlock icon) can be mistaken for images of these icons,
- the meaning, and therefore the importance, of the SSL/TLS icon is not understood, nor is the concept of a certificate,
- the absence of security indicators is not reliably noticed, and
- multiple windows and their attributes cannot be reliably distinguished.

In related work, Friedman et al. [25] established that users found it difficult to determine whether a connection was secure under normal conditions. Intentional phishing and spoofing attempts will only make this task more difficult.

4. Anti-Phishing Techniques

The realm of phishing techniques is large and constantly expanding [16]; however, anti-phishing systems are not commonplace. Dhamija and Tygar [17] identify five basic principles that illustrate why designing secure interfaces is difficult:

1. Limited human skills: any security system design should begin by considering the strengths and weaknesses of the user, rather that starting from a traditional cryptography, 'what can we secure', point of view. For example, it has been shown [28] that users screen out commonly occurring notices (e.g., dialog boxes). Most browsers show such a warning when unencrypted information is submitted over the Internet; predictably, most users either ignore this message entirely or disable it.
2. General-purpose graphics: operating and windowing systems that allow general purpose graphics also permit spoofing. This has important implications for the design of spoof-resistant systems, as we must assume that the design can be easily copied.
3. Golden arches property: the marketing investment made by organizations in promoting their brand and visual identity is designed to invoke trust between the consumer and the organization. However, this trust can be abused: given principle number two, particular care must be taken to prevent the user from assigning trust exclusively based on graphics alone.
4. Unmotivated users: security is generally a secondary goal for a user conducting an online transaction; their focus will be on completing the primary goal (e.g., purchase a product online) rather than ensuring their security. In response to security warning like Fig. 11, most users just click "yes" without reading the warning message.

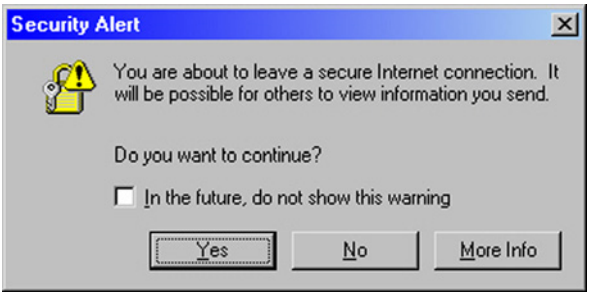


FIG. 11. Security warning pop-up message.

5. The 'barn door': once released, for whatever length of time, user information can be exploited. Secure systems should focus on protecting user information before they leave the user's control.

The authors argue that any complete phishing solution should fulfill all of these goals. In the following sections, we discuss available technical solutions for thwarting phishing attacks.

4.1 Detecting Phishing Attacks

Wenyin et al. [58] propose a system for detecting phishing website based on visual similarity. By examining the similarities between text, images, overall layout and overall style, an overall measure of similarity is produced. Experimental results indicate a low level of false positives based on a collection of 328 suspicious web pages. They intend the algorithm to be applied in a commercial setting by a monitoring company.

An automatic response to a phishing email can be used to detect the authenticity of the response [10]. It retrieves the embedded links in the email, visits the linked website, provides phantom user information, and analyzes the response from the fake website. If the visited website reacts differently from the expected behavior of a legitimate website, it determines that the site is a phishing site.

Some consideration should also be given to the structure of URLs over the entire website; simple URLs can be readily identified by users, and makes the identification of obfuscated URLs somewhat easier. Such updates to custom web applications can be done without interruption to users; however, secure application development requires skilled developers and thorough testing. The number of attack vectors available to the phisher can be substantially reduced through the use of these techniques,

and is relatively cost effective for the organization (when compared with the cost of an attack exploiting their web application).

Unicode attacks exploit the visual similarities between many Unicode characters (Section 2.3). Such attacks can be detected by character-character similarity and word-word similarity [26]. It has been demonstrated that this attack can be accomplished using the English alphabet, Chinese characters, or the Japanese alphabet.

4.2 Retaliation

Several anti-phishing companies offer retaliatory services [27]. They respond by sending phishing sites so much fake financial information that the sites cannot accept information from would-be victims. Most phishing sites run off of web servers installed on hijacked home computers and cannot handle much traffic. However, retaliatory services generally do not shut down phishing sites by overwhelming them with traffic, as occurs in a denial-of-service attack. They just send the sites as much traffic as they can handle and dilute their database with largely false information, a process known as *poisoning*.

Similar technique is proposed by [10]. Phantom user information is provided to the embedded website in the phishing email. By repeating this step rapidly, it can poison the phishing database.

4.3 Client-side Security Measures

The installation of generic security software on a user's local machine can circumvent a number of phishing attacks, in addition to protecting against a number of other security risks. Four key pieces of software should be installed on each user's machine:

- Anti-virus protection: removes malware and protects against the installation of new malware by phishers (and others). It should be regularly updated [22].
- Firewall/IDS: blocks unauthorized network connections that could indicate the installation of an unauthorized phishing program or use of a non-standard port for SSL traffic (which can indicate a phishing operation at work).
- Anti-spyware: removes spyware, which could potentially release sensitive user information to potentially malicious parties.
- Anti-spam: filters out unsolicited bulk email, including many phishing attacks.

Most consumers already recognize the value of anti-virus systems; it would be reasonable to assume they would be similarly interested in the Internet equivalents. While the purchase price for all four components can be substantial, well-regarded

freely available products are also available in each of the four categories. The combination of these services on a local machine can create some false positives; however, the net defense-in-depth effect gained positively impacts on a user's or an organization's security posture. Similar systems should also be deployed at the local network level [36].

Sophisticated email clients are widely used; however, only advanced corporate users require most of the functionality provided. The unnecessary functionality exposes the user to additional exploitable vulnerabilities that can be used by phishers [46]. The success of many phishing attacks can be attributed to the use of HTML email as it is particularly successful in obfuscating hyperlinks. By disabling HTML email in all email client applications, standard obfuscation and spoofing techniques can be rendered ineffective; however, this makes legitimate HTML emails difficult to read. The email client should also prevent the user from quickly executing dangerous content. At minimum, the user should be forced to save the attachment before opening it. This gives anti-virus software a better opportunity to consider the file, as well as preventing malicious code from compromising the rendering application (i.e., the email application). The use of simple clients, plain text email and automated attachment blocking can eliminate potential attack vectors for a phisher.

4.4 Web Browser Enhancement

Web browsers, when properly patched and configured, can be used as a defense mechanism against phishing attacks. In some respects they are similar to email clients: most browsers contain more functionality than the user will typically require [46]. The more functionality provided, the more security flaws are generally exposed. For example, in typical web browsing, a user will only use 5% of Microsoft Internet Explorer's functionality. Therefore, a browser that is appropriate to the user is important: simple web browsers are sufficient and more secure for most users who simply seek to browse the web.

Web browsers should also be properly configured to protect against phishing attacks. Popup windows should be disabled, along with native Java support, ActiveX support, and multimedia auto-playback and auto-execute extensions. In addition, non-secure cookies should not be stored, and new downloads should not be executable from inside the browser before being copied to a local directory.

The plug-in architecture provided by most browsers is being used to support an increasing number of anti-phishing systems. Security toolbars are widely in use. For example: Spoofstick [52] displays a website's real domain name; Netcraft Toolbar [45] displays a information about the site; Trustbar [33] displays a logos and the certificate authority of the website; eBay Account Guard [21] indicates the true eBay

site; SpoofGuard [11] calculates the 'spoof' score; and Web Wallet [63] tries to ensure users submit their data to the intended site. Typically these plug-ins are added to the browser toolbar, and confirm that the current URL is not part of a known phishing attack by contacting a centralized server. Ultimately, their effectiveness is dependent on the reporting mechanisms used by the system. Users often ignore toolbar messages and toolbars also make mistakes, so these toolbars must be used with caution [62]. The embedded links in phishing emails often contain a different link from that displayed in the text. For example,

```
<a href=http://123.132.234.87> http://www.goodsite.com</a>
```

An anti-phishing browser extension [40] would detect such discrepancy and warn the user. Other approaches to identifying phishing websites include [51]:

- Social networking: the toolbar informs the user if other people they know have viewed and trusted the site.
- 'Golden arches' property: the Trustbar takes advantage of the 'Golden arches' property described by Dhamija [16]. It displays the company's logo, as well as the logo of the company that signed the SSL/TLS server certificate (e.g., Verisign).

Dhamija and Tygar [17] propose the use of trusted security windows for the display and submission of credentials. The user would assign a unique security image as the background of the security window. The image would be stored locally, and could not be spoofed by a remote user. Therefore, the user would be aware when they were looking at legitimate security information or entering their username and password into an authentic form. The use of browser-generated random images or server-generated random images (also known as dynamic security skins) can also provide the user with a prominent visual indication of a secure connection. Z. Ye et al. [64] proposed trusted paths from the browser to the human user that might work under browser spoofing.

In the future, the anti-phish functions may become a built-in feature of web browsers. Netscape plans to release a web browser designed to resist phishing which will frequently update blacklists of suspected phishing websites and a whitelist of trusted sites. When a user follows an e-mail link and visits a trusted site, the browser will automatically render it, but block user access if the site is part of a known phishing attack. When a user visits a site not on a whitelist or blacklist, the browser renders it with enhanced security that disables ActiveX and JavaScript capabilities, which phishers could use to exploit vulnerabilities. Microsoft intends to provide anti-phishing functionality as a core part of its next browser, Internet Explorer 7.0. Deepnet Explorer 1.4 [15], a browser shell that uses the current version of Internet Explorer to render web pages, analyzes web addresses and warns users about those

on a blacklist of suspect sites. Users can then choose to either stop or continue trying to access a site.

4.5 Server-side Security Measures

Organizations should take a role in preparing users for an eventual phishing attack [24]. Most major online vendors, such as major banks, PayPal or eBay, already practice this to some extent (and to some effect) [46]. Communications from the organization should remind users not to release credentials to any other party, with an emphasis on prompting the user to consider the legitimacy of the motivation (e.g., email hyperlink) that drove them to the page. General phishing resources should also be made available to customers, detailing methods that can be used to ensure the validity of a site and how a customer can report a phishing scheme. Reported attacks should be responded to quickly, and users appropriately notified. Finally, all outgoing communications should be standardized; this reduces the likelihood legitimate communications could be confused with a phishing attack. All of these suggestions have a low cost to the organization, but must be delivered in a consistent manner where the customer is not overloaded with information.

Organizations can take a number of steps to validate their email communications with their customers, in order to make phishing attacks more obvious [46]. Emails can be personalized with some personal information known only to trusted organizations, such as greeting the customer by name, or including the last few digits of their credit card. A trail of trust can be established if subsequent emails precisely reference previous communications. Digital signatures can also be used to securely sign emails [1,55]; however, this relies on the user to validate the signature. Specialist web applications can also be made available to users to check the email was in fact sent from the organization. In order for these techniques to deter a phishing attack, the user must be aware of their existence and actively look for them.

Poor development techniques can expose custom web applications to some phishing techniques, such as cross-site scripting or the inline embedding of custom content (as discussed previously). Some of the key security requirements for a custom web application include [46]:

- all submitted content should be validated,
- session identifiers should be carefully monitored,
- tightly control URL redirection services provided,
- ensure safeguards are present in the authentication process (e.g., two-stage logins, anti-key logging processes, or personalized content), and
- use image cycling (regularly change the name of images on the site to render fake websites, that link to the legitimate website, out-of-date).

4.6 Alternative Authentication

Two-factor authentication (e.g., username/password and a secure token) has been suggested [46] as a possible solution to phishing attacks. By making the password time-dependent (i.e., it can only be used once), the phisher is limited in their ability to subsequently connect to the server. This system combats simple eavesdropping and password guessing; however, it is not a complete solution to phishing attacks [50,54]. Attack techniques such as man-in-the-middle or the use of Trojan horses will not be stopped: man-in-the-middle will still grant the phisher access, and Trojan horses will allow the phisher access to subsequent sessions from that machine. Two-channel authentication[6] is similarly vulnerable to active phishing attacks, but would eliminate some phishing attack vectors.

Delayed password disclosure [51] requires the server to continuously authenticate itself with the user. After a user enters each character of their password, a predefined image selected by the user is displayed. The pattern of images would be difficult for a phishing website to mimic. Mutual authentication is also achieved by using server and client side certificates. However, this requires users to have their certificate with them in order to connect to their bank; this inconvenience will limit the use of this technology [13].

Sophisticated browser password management can also be used to circumvent phishing attacks [51]. If the user allows the browser to manage all passwords, and a domain name is associated with each password, a user's credentials will only be automatically entered at legitimate websites [31].

Bellovin [6] believes that new authentication mechanisms will fail until prior relationships can be adequately captured. The use of certificates, both in email and on websites, merely guarantees the sender/website owns that particular domain name. It does not guarantee that this is the same party that the user gave money or sensitive data to. He proposes a simple solution to illustrate this point: if users were provided with the bank's certificate when opening an account, the certificate could be used to authenticate bank email and websites. The certificate is bound to a previous legitimate transaction, rather than simply being bound to a name.

4.7 Email Security

By modifying existing spam email filtering approaches, phishing emails can be detected and filtered by analyzing their content. According to [35], 54 out of 3,370 spam emails intercepted were phishing emails. Phishing emails typically contained

[6] Two-channel authentication requires the user to authenticate over two different mediums. For example, part of the authentication would involve the bank sending a challenge via SMS, and the user replying via SMS.

text related to banks and auction sites. By checking the text and other email characteristics such as sender, domain, and links, they formulated a scoring system to identify and block phishing mails.

Digital signatures can be used to make it easier to check the identity of the sender and the integrity of the message. However, it is still possible for a phisher to send a message using an anonymous public/private key pair. There are two popular standards for digitally signed email, S/MIME and PGP, which are supported by most Internet mail clients.

5. Comprehensive Anti-Phishing Efforts

Van der Merwe et al. [44] identify five key counter-attack categories for users and organizations to consider:

1. Education: users should be equipped with the skills to identify, and avoid, potential phishing attacks [36]. To a certain extent, this approach has failed: the vast majority of email correspondence reminds users that the organization will NEVER ask them for their password. Despite these regular warnings, phishing attacks continue to succeed in doing exactly that. Ironically, many phishing emails also include similar warnings.
2. Preparation: the danger of a phishing attack should be recognized, and policies put in place to manage or respond to such attacks. Different authentication technologies should be assessed for potential vulnerabilities. This particular category is of more relevance to organizations.
3. Avoidance: steps can be taken to avoid becoming the target of a phishing attack. For example, the use of anti-spam systems to filter out phishing messages or the use of Verisign verification on secure websites.
4. Intervention: when those behind phishing attacks step forward to influence the outcome of the attack, their success will be entirely dependent on the user. This relies on category one: the user should stop to think before submitting any personal information over the Internet.
5. Treatment: after a phishing attack, systems must be able to recover, identify the extent of the damage, and contact the appropriate organizations to prevent the misuse of sensitive information.

In other words, phishing cannot be prevented just by technical means alone; rather, a comprehensive response is necessary.

5.1 User Vigilance and Education

The behavior of users targeted by phishing attacks has been studied extensively in [18,19,48]. [18] observed the responses of 22 participants and analyzed the results by sex, age, education level, hours using the computer, etc. The study did not find any of these factors made a significant difference in the susceptibility of the user to the attack. Somewhat shockingly, even in the best case scenario, when users expected spoofs to be present and were motivated to discover them, many users could not distinguish a legitimate website from a spoofed website. In fact, the best phishing site was able to fool more than 90% of participants. In [19], 57 participants were tested and found that people use various strategies to distinguish phishing websites; however, these techniques were not necessarily effective. In [48], a user education course was offered and found that the user-awareness was greatly improved. Individual users are the most essential piece in an anti-phishing effort and they must take an active role to avoid becoming a victim of a phishing attack. Users can take several simple steps to protect themselves and their privacy:

- If a user gets an email warning that their account will be shut down unless they reconfirm billing information, they should not reply.
- Never respond to HTML email with embedded submission forms (i.e., never enter information directly into an email).
- Never click on hyperlinks within email even if they look legitimate; instead, directly type in the URL in the web browser.[7]
- Avoid emailing personal and financial information.
- Do not email back to confirm account information. Instead, call or log on to the company's website.
- For sites that indicate they are secure, review the SSL certificates by clicking the lock icon. Call the company if any certificate warning messages are displayed when you log into the website.
- Review credit card and bank account for unauthorized charges.
- Report suspicious activity.
- Ensure software updates are applied in a timely manner.

5.2 Proactive Detection of Phishing Activities

Various companies offer monitoring services, which are aimed at the early detection and elimination of phishing attacks. For example [27]:

[7] This would still leave the user vulnerable to a DNS poisoning attack; however, it would defeat a significant percentage of phishing attacks, which rely on malformed or disguised URLs.

- Corillian monitors and evaluates suspicious traffic on weekends, when most phishers conduct reconnaissance. By analyzing web logs, they are able to identify patterns of possible phishing behavior, such as downloading and saving images, or linking to images from a remote site. The process of verifying stolen accounts can also be detected.
- NameProtect identifies phishing attacks by monitoring spam from many sources (e.g., honey pot accounts [34]) and by checking domain name registration records for newly registered domains with names similar to that of their client.
- Cyota provides account information to phishers. The accounts themselves are set up in order to observe the phishing and fraud process. This allows the organizations involved to learn more about the nature of the attack.

5.3 Reporting and Response

Early reporting of phishing schemes allows them to be shut down as soon as possible and also allows users to be provided with some warning (e.g., by the organization involved or through anti-phishing software) [49]. Major banks and e-commerce businesses generally have reporting forms as part of their website; the US Bank provides an email address to forward suspect emails to, while Citibank also lists recent scams with a link to each one. Independent groups, such as the Anti-Phishing Working Group, also maintain information regarding known phishing attacks. Digital Phishnet is an organization formed to fight phishing attacks. It combines the forces of nine of the top ten US banks and financial services providers, four of the top five ISPs and five digital commerce and technology companies. They cooperate with the FBI, Federal Trade Commission (FTC), US Secret Service and the US Postal Inspection Service, under the aegis of the FBI's Internet Crime Complaint Center (IC3) [41].

Once reported, law enforcement officials are responsible for shutting the website down, tracing the source of the emails, tracking stolen funds and prosecuting those responsible. In Australia, the Australian High Tech Crime Centre and the Australian Computer Emergency Response Team are responsible for pursuing reported phishing attacks [49]. The URL contained within the phishing email will be used in a DNS search to find the ISP responsible for hosting the attack. This information usually allows the website to be quickly shutdown; however this may not be the case if the ISP is overseas, or in an unfriendly country. A G8 taskforce, consisting of 37 member countries, has recently been established to combat computer crime, including phishing. Of the phishing attacks recorded in May 2006 [3], 34.1% were conducted from inside the US, 15% from China, 8.17% from Korea, 3.94% from France, 3.38% from Germany.

Through effective reporting, historical conceptions about the spread of phishing attacks are changing [29]. Rather than spreading in a disorganized wildfire pattern, researchers now believe phishing attacks originate from specific IP blocks. CipherTrust [12] believes most phishing attacks are likely to originate from fewer than 5,000 networks. Messages sent from sources that do not typically send legitimate email are candidates for subsequent analysis. The IP addresses contained in such emails can then be followed to check for phishing attacks. More research is likely to allow researchers to better characterize phishing attacks.

5.4 Legal

In the United States, Democratic Senator Patrick Leahy introduced the *Anti-Phishing Act of 2005* on February 28, 2005 [30]. It allows prison time of up to five years and fines of up to US $250,000 for people who design fake websites for the purposes of stealing money or credit card numbers. California passed an anti-phishing law in September 2005, permitting victims to seek recovery of actual damages or up to $500,000 for each violation, whichever is greater [32]. Other US states, including Texas, New Mexico and Arizona, have also passed an anti-phishing law.

Although not common, some phishers get arrested. A 45-year-old California man, Jeffrey Brett Goodin, was arrested in January 2006 and charged with operating an online phishing scheme that targeted America Online customers [47]. He was charged with wire fraud and unauthorized use of a credit card. Goodin is alleged to have sent e-mail messages to thousands of AOL users to entice them to visit fraudulent websites he set up to collect personal information. Another phisher was arrested in August 2005 in Iowa [57]. Jayson Harris was charged with 75 counts of wire fraud for allegedly stealing credit card numbers and personal information in a phishing scheme targeting Microsoft's MSN customers. Other countries have followed the lead of the U.S. by tracing and arresting phishers.

Companies are taking proactive approaches in cracking down the phishers. On March 31, 2005, Microsoft filed 117 federal lawsuits in the US District Court for the Western District of Washington. The lawsuits accuse phishers of using various methods to obtain passwords and confidential information. AOL reinforced its efforts against phishing in early 2006 with three lawsuits seeking a total of $18 million USD under the 2005 amendments to the Virginia Computer Crimes Act.

6. Conclusion

Much of the Internet's malicious user population[8] has historically been motivated by challenge, curiosity, rebellion, vandalism, and the desire for respect and power. Modern trends in phishing reveal a very different situation: criminals have adopted the well-developed and well-known techniques of malicious users and are exploiting Internet users with sophisticated phishing attacks. The concept of phishing has mutated significantly since its creation almost ten years ago. Modern phishers are financially motivated and likely to pursue their attacks more aggressively than the average cracker [53]. The influence of organized crime further supports the changing nature of crime on the Internet. Phishing is also being used target individual users in an attempt to gain access to specific resources [27].

However, the outlook is not entirely bleak: anti-virus, anti-spyware and anti-spam systems are continuing to evolve, as are Internet browsers. If organizations prepare well, remain vigilant and follow attack trends carefully, they can respond quickly and effectively with a range of techniques to defend their customer's data. If individuals take a responsibility for their protection and adopt a defense-in-depth approach, consisting of a comprehensive and complementary toolkit of software and education, they can defend themselves against the most sophisticated attacks. There is no simple solution, but active and aware users and organizations have the ability to form a strangle-hold on this ever-growing threat. Consider yourself warned!

Acknowledgements

Fragments of this chapter have been taken from Berghel, "Phishing Mongers and Posers" [7] with the permission of the publisher.

References

[1] Anti-Phishing Working Group, "Proposed solutions to address the threat of email spoofing scams", 2003.
[2] Anti-Phishing Working Group, "Origins of the Word "Phishing" ", 2005.
[3] Anti-Phishing Working Group, "Phishing activity trends report", May 2006.
[4] Anti-Phishing Working Group, "Phishing archive", 2005.
[5] Anti-Phishing Working Group, "What are phishing and pharming?", http://www.antiphishing.org, 2006.
[6] Bellovin S.M., "Spamming, phishing, authentication, and privacy", *Commun. ACM* **47** (12) (2004) 144.

[8] Hackers, crackers and script kiddies.

[7] Berghel H., "Phishing mongers and posers", *Commun. ACM* **48** (4) (2006) 21–25.
[8] Berghel H., Brajkovska N., "Wading into alternate data streams", *Commun. ACM* **47** (4) (2004) 21–27.
[9] CACM Staff, "News track", *Commun. ACM* **48** (2) (2005) 9–10.
[10] Chandrasekaran M., et al., "PHONEY: Mimicking user response to detect phishing attacks", in: *Proc. 2006 Int. Symposium of World of Wireless, Mobile and Multimedia*.
[11] Chou N., Ledesma R., Teraguchi Y., Mitchell J.C., "Client-side defense against web-based identity theft", in: *11th Annual Network and Distributed System Security Symposium*, 2004.
[12] CipherTrust, http://www.ciphertrust.com.
[13] Clayton R., "Insecure real-world authentication protocols (or why phishing is so profitable)", in: *Thirteenth Cambridge Protocols Workshop*, Sidney, Sussex, UK, 2005.
[14] Cloudmark, http://www.cloudmark.com.
[15] Deepnet Explorer Browser, http://www.deepnetexplorer.com.
[16] Dhamija R., "Detecting phishing attacks: A user task analysis", in: *Authentication for Humans: Designing and Evaluating Usable Security Systems*, PhD dissertation, School of Management Information Systems, UC Berkeley, 2005.
[17] Dhamija R., Tygar J.D., "The battle against phishing: Dynamic security skins", in: *ACM Symposium on Usable Security and Privacy*, 2005.
[18] Dhamija R., Tygar J.D., Hearst M., "Why phishing works", in: *CHI 2006*, April 22–27, 2006.
[19] Downs J., Holbrook M., Cranor L.F., "Decision strategies and susceptibility to phishing", in: *Symposium on Usable Privacy and Security (SOUPS)*, July 12–14, 2006, pp. 79–90.
[20] Drake C.E., Oliver J.J., Koontz E.J., "Anatomy of a phishing email", in: *Conference on Email and Anti-Spam*, Mountain View, CA, USA, 2004.
[21] eBay Toolbar and Account Guard, http://pages.ebay.com/help/confidence/account-guard.html.
[22] Federal Trade Commission, "How not to get hooked by a 'phishing' scam", 2005.
[23] Fernandez J.D., et al., "Computer forensics: a critical need in computer science programs", *J. Comput. Small Coll.* **20** (4) (2005) 315–322.
[24] Flinn S., Stoyles S., "Omnivore: Risk management through bidirectional transparency", in: *Proceedings of the 2004 workshop on New security paradigms*, ACM Press, Nova Scotia, Canada, 2005, pp. 97–105.
[25] Friedman B., et al., "Users' conceptions of web security: A comparative study", in: *ACM CHI: Conference on Human Factors in Computer Systems*, 2002.
[26] Fu A., et al., "The methodology and an application to fight against Unicode attacks", in: *Proc. SOUPS*, 2006.
[27] Geer D., "Security technologies go phishing", *Computer* **38** (6) (2005) 18–21.
[28] Good N., et al., "Stopping spyware at the gate: A user study of privacy, notice and spyware", in: *Symposium on Usable Security and Privacy*, 2005.
[29] Goth G., "Phishing attacks rising, but dollar losses down", *Security & Privacy Magazine, IEEE* **3** (1) (2005) 8.
[30] Grant Gross, "Anti-Phishing Act pushes for 5 years and $250,000 fine", http://www.thestandard.com/internetnews/001048.php, March 5, 2005.

[31] Halderman J.A., Waters B., Felten E.W., "A convenient method for securely managing passwords", in: *Proceedings of the 14th International Conference on World Wide Web*, ACM Press, Chiba, Japan, 2005, pp. 471–479.
[32] Haskins W., "California passes nation's first antiphishing law", http://www.newsfactor.com/news/California-Passes-First-Antiphishing-Law/story.xhtml?story_id=010000Z2F774, October 4, 2005.
[33] Herzberg A., Gbara A., "TrustBar: Protecting (even naïve) web users from spoofing and phishing attacks", http://www.cs.biu.ac.il/~herzbea/Papers/ecommerce/spoofing.htm, 2004.
[34] The Honeynet Project & Research Alliance, *Know Your Enemy: Phishing*, The Honeynet Project & Research Alliance, 2005.
[35] Inomata A., Rahman S., Okamoto T., Okamoto E., "A novel mail filtering method against phishing", in: *PACRIM*, 2005.
[36] Internet Security Systems, "Protect your business from phishing", 2005.
[37] Jakobsson M., "Modeling and preventing phishing attacks", in: *Financial Cryptography '05*, 2005.
[38] Keizer G., "Phishing costs nearly $1 billion", InformationWeek, 2005.
[39] Kerstein P., "How can we stop phishing and pharming scams?", in: *CSO*, July 19, 2005.
[40] Kirda E., Kruegel C., "Protecting users against phishing attacks with antiphishing", in: *COMPSAC*, 2005.
[41] Kuchinskas S., "Phish fighters form alliance", http://www.internetnews.com/bus-news/article.php/3445511, December 8, 2004.
[42] Levy E., "Criminals become tech savvy", *Security & Privacy Magazine, IEEE* **2** (2) (2004) 65–68.
[43] Levy E., "Interface illusions", *Security & Privacy Magazine, IEEE* **2** (6) (2004) 66–69.
[44] Merwe A.v.d., Looc M., Dabrowski M., "Characteristics and responsibilities involved in a Phishing attack", in: *Proc. of the 4th International Symposium on Information and Communication Technologies*, Trinity College Dublin, Cape Town, South Africa, 2005, pp. 249–254.
[45] Netcraft Toolbar, http://toolbar.netcraft.com.
[46] Ollmann G., "The phishing guide", NGS Software Insight Security Research, 2005.
[47] Roberts P., "California man arrested in AOL phishing scheme", http://www.eweek.com/article2/0.1895.1916273.00.asp, January 27, 2006.
[48] Robia S., Ragucci J., "Don't be a phish: Steps in user education", in: *ITiCSE '06*, June 26–28, 2006.
[49] Rudd B., *An Analysis of Phishing and Possible Mitigation Strategies*, SANS Institute, 2004.
[50] Schneier B., "Two-factor authentication: Too little, too late", *Commun. ACM* **48** (4) (2005) 136.
[51] Sinclair S., Smith S.W., "The TIPPI point: Toward trustworthy interfaces", *Security & Privacy Magazine, IEEE* **3** (4) (2005) 68–71.
[52] Spoofstick, http://www.spoofstick.com.
[53] Treese W., "The state of security on the internet", *netWorker* **8** (3) (2004) 13–15.
[54] Tuliani J., *The Future of Phishing*, Cryptomathic Ltd., 2004.

[55] Tumbleweed, *Using Digital Signatures to Secure Email and Stop Phishing Attacks (White Paper)*, Tumbleweed Communications, 2005.
[56] Varghese S., "Korgo worm takes phishing to a new level", Sydney Morning Herald, Sydney, 2004.
[57] Wagner J., "MSN billing phisher arrested", August 24, 2005; http://www.internetnews.com/security/article.php/3529746.
[58] Wenyin L., et al., "Detection of phishing webpages based on visual similarity", in: *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, ACM Press, Chiba, Japan, 2005, pp. 1060–1061.
[59] Wikipedia, "Identity theft", Wikipedia, the free encyclopedia, 2005.
[60] Wikipedia, "Phishing", Wikipedia, the free encyclopedia, 2005.
[61] Wikipedia, "Social engineering (computer security)", Wikipedia, the free encyclopedia, 2005.
[62] Wu M., et al., "Do security toolbars actually prevent phishing attacks?", in: *Proc. CHI*, 2006.
[63] Wu M., et al., "Web wallet: Preventing phishing attacks by revealing user intentions", in: *SOUPS*, 2006.
[64] Ye Z., Smith S., Anthony D., "Trusted paths for browsers", *ACM Trans. Inform. System Security* **8** (2) (May 2005) 153–186.

Reflections on System Trustworthiness

PETER G. NEUMANN*

Computer Science Laboratory
SRI International
EL-243, 333 Ravenswood Ave
Menlo Park, CA 94025-3493
USA

Abstract

We examine here a range of concerns relating to computer systems and networks, with particular attention to difficulties in system development, and the resulting vulnerabilities, threats, and risks. We examine some approaches that might achieve dramatic improvements in the ability to develop, operate, and use trustworthy systems. The problems and their solutions typically require a combination of technology and social policy.

*Principal scientist.

ISSN: 0065-2458/DOI: 10.1016/S0065-2458(06)70006-7

1. A Total-System Perspective

Each of the following items presents pressing challenges relating to the constructive use of information technology. The totality of all the interrelated challenges requires concerted efforts that transcend the individual problems.

• *Trustworthiness*. Trustworthiness implies simply that something is worthy of being trusted to satisfy its expected requirements. Users often trust systems that are not worthy of being trusted, with respect to attributes such as system and network security, system reliability and survivability, human safety, interoperability, predictable system behavior, and other important attributes that are for the most part not receiving enough concerted attention. Computer-communication infrastructures are typically riddled with flaws. In the absence of more serious attacks, governments and system developers seem to have been lulled into a false sense of security. At present, neither proprietary nor source-available system developers are sufficiently militant in satisfying critical needs. In mass-market software, the patch mentality seems to have won out over well-designed and well-implemented systems.

• *Total system life-cycle issues*. Developing and operating trustworthy systems is inherently difficult today. Typically, a system is not likely to be trustworthy unless the relevant attributes were explicitly recognized from the beginning of system development, reflected in sound system architectures and software development, explicitly addressed in system procurements, and their fulfillment mandated throughout system operation.

• *System development practice*. Costly failures have occurred in developing large systems, such as the modernization efforts for the US Internal Revenue Service, US and UK air traffic control systems, the FBI Virtual Case File, and German TollCollect, to name just a few. Procurement and development of large-scale hardware/software systems remain high-risk activities, with cost overruns, delays, and even abandonment of entire projects.

• *The Internet*. Increasingly, many enterprises are heavily dependent on the Internet, despite its existing limitations. Internet governance, control, and coordination create many contentious international problems. The Internet infrastructure itself is susceptible to denial-of-service attacks and compromise, while the lack of security and dependability of most attached systems also creates problems (e.g., penetrations such as open relays being used to host zombies and "bots"). Worms, viruses, and other malware are often impediments, as are ubiquitous problems of spam e-mail and phishing attacks that may result in identity theft.

• *Critical national infrastructures*. Despite some past recognition of the pervasiveness of serious vulnerabilities, critical national infrastructures such as electrical

power, energy, telecommunications, transportation, finance, and government continuity are typically still vulnerable to attacks and accidental collapses. For example, massive power outages keep recurring, despite supposed improvements. Telecommunications outages can have severe consequences, as can transportation shutdowns and fuel shortages. Furthermore, these infrastructures have interdependencies that result in widespread system failures.

• *Accountability*. Oversight of computer activities is often as weak as oversight of corporate practices. On the other hand, audit mechanisms must also respect privacy needs. As one example that fails on both counts, today's unauditable all-electronic voting systems are lacking in accountability in a would-be effort to protect voter privacy. In fact, without the addition of some sort of voter-verified audit trail, they provide no meaningful assurances that votes are correctly recorded and processed. (For example, see the October 2004 special issue of the *Communications of the ACM*, devoted to the integrity of election systems.)

• *Privacy*. Privacy is something that many people do not value until after they have lost it. Personal privacy is relevant pervasively in our lives, especially in financial matters and health care. Some advocates of homeland security have postulated the need to sacrifice privacy in order to attain security, although the necessity of this tradeoff is highly debatable. Sacrificing privacy does not necessarily result in greater security. (Benjamin Franklin's quote is apt in this regard: "Those who would sacrifice liberty for security deserve neither.") Furthermore, serious inroads to privacy protection have occurred that may be difficult to reverse. Surveillance is becoming more widespread, but often without adequately respecting privacy. Legitimate needs for anonymity or at least pseudoanonymity (for example, to protect victims and legitimate whistle-blowers) must not be suppressed or dismissed as dangerous.

• *Education*. In many countries, university curricula in software engineering and trustworthiness inadequately reflect the needs of critical systems. Instruction is often aimed at programming in the small, while more or less ignoring systems in the large. This situation has potentially serious long-term implications worldwide.

As noted above, it is the totality of these problems that is of primary concern. Simplistic local approaches are not effective. Greater foresight and pervasive system-oriented thinking are urgently needed, along with greater private-public cooperation.

2. Anticipating Disasters

As Henry Petroski noted over twenty years ago [42], we generally learn less from successes than from failures. The ACM Risks Forum [32] and *Computer-Related*

Risks [34] include a startling number of failures and risks, and provide a goldmine of opportunity for anyone who wants to learn from past mistakes. Intriguingly, or perhaps ironically, most of the content of [34] is still as relevant today as it was in 1995. The same types of failures continue to recur, and the range of causes remains much the same. Indeed, the scope and extent of the risks has increased steadily. For example, the ACM Risks Forum continues to report computer system development fiascos and operational failures of aircraft, air-traffic control, defense systems, train crashes, electrical power, telecommunications, medical health systems, and financial problems. These difficulties include problems in reliability, system survivability, security, privacy, and human well-being. Some of these problems have been escalating dramatically, such as spam, phishing attacks, identity thefts, and financial losses.

In recent years, some unusual natural disasters have occurred, such as the 9.0-magnitude Indonesian earthquake that triggered a tsunami killing more than 200,000 people in 11 countries around the Indian Ocean, the exceptionally heavy 2006 hurricane season in the Caribbean area including the devastating effects of Katrina, and a major mudslide in La Conchita, California. Although failures of information technology obviously had no role in *triggering* these disasters, IT systems could play significant roles in *anticipating, detecting, monitoring, and responding to* such events, minimizing losses of life, injuries, and consequential damages. What have we learned from such events, especially with respect to the need for proactive contingency plans?

For example, a tsunami detection and early-warning system such as had already been deployed in the Pacific Ocean could also have been used in the Indian Ocean. Such a system could have given timely warnings to millions of people, and could have saved many lives *if* local authorities had citizen alerts and evacuation plans in place. Early warnings and preparedness for hurricanes and typhoons are improving as computer prediction of possible storm paths is becoming more accurate and as many authorities prepare disaster response plans and train for their deployment. However, preparedness tends to improve only after disasters have occurred (and then often only temporarily). In the case of the mudslide in the hills above La Conchita, which followed an awesome sequence of rainstorms, sensors in the hills were designed to trigger advance warnings, which evidently were not taken seriously enough. (A similar slide had occurred in an adjacent area nine years earlier, and insurance companies had already declined to provide future coverage.)

Several problems arise in connection with developing detection and warning systems.

- Institutions (especially governments, corporations, and defense departments) tend to fashion response plans for past situations rather than for potentially devastating future situations. Unless a similar disaster has recently occurred in a sim-

ilar venue under similar conditions, few people worry about low-probability high-impact events. A comparable tendency holds for trustworthy computing. A computer networking event not unlike a tsunami occurred in 1988—namely, the Internet Worm [47,54] that affected about 10% of the 60,000 Internet hosts active at the time. As a result, an emergency response team (now US-CERT) was formed to help coordinate responses and warn of vulnerabilities. Prior to the year 2000, a large upgrade effort to avoid the Y2K crisis was generally successful; the situation could have been much more serious without the intensive remediation efforts. Today, many new threats such as malware and terrorist attacks could easily disable critical infrastructures and the Internet. However, because the cybersecurity equivalent of a tsunami seems extremely unlikely to many people unfamiliar with the nature of the vulnerabilities, there is little interest in mounting efforts to increase system trustworthiness and engage in other preventive measures. The consequences of major meltdowns could be very dramatic, especially if accompanied by terrorist attacks.

- Institutions tend to optimize short-term costs and ignore long-term consequences. Also, farsighted analyses of what might happen are always subject to poor assumptions, faulty reasoning, and mandates to reach self-serving conclusions. This is discussed further in Section 8.

- People generally do not like to make unnecessary preparations, and often resent taking sensible precautions. Repeated false warnings tend to inure them, with a resulting loss of responsiveness. Even justifiable warnings that are heeded (such as the Y2K remediation or boarding up for an oncoming hurricane) are often denigrated if the resulting effects are only relatively minor.

It is clear that much greater attention needs to be devoted to predicting, detecting, and ameliorating both natural catastrophes and unnatural computer-related misuse, attacks, disasters, and outages. Efforts are needed to dramatically improve the trustworthiness of those systems on which many lives depend, and to make those systems more tolerant to human misbehavior as well as malfunctions and natural causes.

3. Trustworthiness

Estimates of system trustworthiness ultimately depend on having some sort of logical basis for confidence that a system will predictably satisfy its critical requirements. Measures of trustworthiness are particularly important for information security, system integrity and reliability, human safety, fault tolerance, and overall enterprise survivability in the face of wide ranges of adversities (including malfunctions, deliberate attacks, and natural causes).

Many lives increasingly depend on critical national infrastructures—all of which in turn depend in varying degrees on the predictable behavior of computer-communication resources. Indeed, these infrastructures often depend on the Internet for information and control and may be vulnerable to attacks from any attached computer systems.

Unless critical information system resources are sufficiently trustworthy, other systems are at risk from failures and subversions. Unfortunately, for many of the key application domains, the existing information infrastructures are lacking in trustworthiness. For example, power grids, air-traffic control, high-integrity electronic voting systems, the emerging US Department of Defense Secure Global Information Grid, national infrastructures, and many collaborative and competitive Internet-based applications all need systems that are more trustworthy than we have today or are likely to have in the foreseeable future.

Numerous steps are needed to develop trustworthy systems. Consider an analogy with the planet's natural environment—expectations for which are somewhat similar to expectations for trustworthy information systems. For example, pure air and uncontaminated water are vital, as are the social systems that ensure them.

Although poorly chosen analogies can be misleading, the analogy with the natural environment is appropriate. Each of the following items is applicable to both trustworthy information systems and natural environments.

- Their critical importance is generally underappreciated until something goes fundamentally wrong—after which undoing the damage can be very difficult if not impossible.
- Problems can result from natural circumstances, equipment failures, human errors, malicious activity, or a combination of these and other factors.
- Dangerous contaminants may emerge and propagate, often unobserved. Some of these may remain undetected for relatively long periods of time, whereas others can have immediately obvious consequences.
- Once something has gone recognizably wrong, palliative countermeasures are typically fruitless—too little, too late.
- Your own well-being may be dramatically impeded, but there is not much you as an individual can do about aspects that are pervasive—perhaps international or even global in scope.
- Detection, remediation, and prevention require cooperative social efforts, such as public health and sanitation activities, as well as technological means including increased trustworthiness.
- Up-front preventive measures can result in significant savings and increased human well-being, ameliorating major problems later on.

- As discussed further in Section 8, long-term thinking is relatively rare. There is frequently little governmental or institutional emphasis on proactive prevention of bad consequences. Many of the arguments against far-sighted planning and remediation are skewed, being based on faulty, narrowly scoped, or short-sighted reasoning—especially relating to short-term profits rather than long-term savings and other benefits.
- Commercial considerations tend to trump human well-being, with business models sometimes considering protection of public welfare to be detrimental to corporate and enterprise bottom lines.

In some contexts, pure water is becoming more expensive than oil. Fresh air is already a crucial commodity. Short- and long-term effects of inadequately trustworthy information systems can similarly be costly. Proactive measures are as urgently needed for system trustworthiness as they are for breathable air, clean water, and environmental protection. It is difficult to remediate computer-based systems that were not designed and implemented with trustworthiness in mind. It is also difficult to remediate serious environmental damage.

Anticipating and responding to compelling long-term needs does not require extraordinary foresight, whether for air, water, reversing global warming, or trustworthy systems upon which to build infrastructures. Our long-term well-being—perhaps even our survival—depends on our willingness to consider the future and commitment to taking appropriate actions.

4. Risks in Trusting Untrustworthiness

The Internet provides ample opportunity for proving the age-old truism, "There's a sucker born every minute." Carnival-style swindles and other confidence games once limited to in-person encounters are now proliferating electronically, worldwide, at low cost and effort. Blatantly obvious pre-Internet examples are the so-called Nigerian-style postal scams that requested use of one's bank account to help move money; hoping for a proffered generous commission, the suckers are then separated from their assets. These scams have been updated to today's e-mail phishing and e-mail scam attacks that efficiently harvest personal information from vastly more people, and are considerably more sophisticated—for example, replicating a legitimate website in every respect except for perhaps just one hard-to-detect bogus URL. Indeed, it is becoming increasingly difficult to distinguish the real from the bogus, and people continue to be victimized.

Many other kinds of scams, stings, and misrepresentations also exist. Deceptive unsolicited e-mail (spam) offering bogus goods and services opens up new avenues

for fraud and identity theft. Online activities are emerging with glaring opportunities for swindles, manipulations, and assorted malfeasance, such as online auctions (with irregularities such as nondelivery and secondary criminality), an alarming increase in highly sophisticated phishing attacks, Internet gambling, and fraudulent websites (e.g., with deceptive URLs creating the appearance of legitimacy). Any of these and other situations could result in inordinate risks, such as financial ruin, blackmail, compromised democracy, or even loss of life. But it is perhaps not surprising that people fall for such schemes, particularly when the technology superficially appears genuine.

Today's unauditable paperless all-electronic voting systems present significant risks (see Section 9). The risks are even greater for voting over the Internet. With independent accountability seriously lacking today, e-voting can be likened to using an off-shore gambling site not subject to any regulation and managed by unknown and unaccountable agents.

We tend to trust third-party relationships with banks, telephone companies, airlines, and other service providers whose employees have in some way earned our trust, collectively or individually. But what about untrustworthy third parties? Some computer-based applications rely critically on the putative integrity and noncompromisibility of automated trusted third parties, with little if any easily demonstrated human accountability. Examples include digital-certificate authorities, cryptographic servers, surveillance facilities, sensitive databases for law enforcement, and credit-information bureaus. With appealing short-term cost incentives for pervasive use of outsourcing, the need for demonstrably trustworthy third-party institutions becomes even more important. However, security, privacy, and accountability are often ignored in efforts to save money.

Is placing trust in offshore enterprises inherently riskier than using domestic services? Not necessarily. Corruption and inattention to detail are worldwide problems. The deciding factor here is the extent to which comprehensive oversight can be maintained.

Is domestic legislation enough? Of course not. Any legislation should not be overly simplistic; for example, it should avoid seeking solely technological fixes or purely legislative solutions to deeper problems. Besides, serious complexities arise from the fact that such problems are international in scope and demand international cooperation.

Is there a role for liability (for flagrant misbehavior or injurious neglect) and differential insurance rates—for example, based on how well a purveyor is living up to what is expected of it? Such measures have significant potential, although they will be strongly resisted in many quarters.

So, how can we provide some meaningful assurance that critical entities such as direct or third parties are sufficiently trustworthy? Ideally, institutions providing,

controlling, managing, and monitoring potentially riskful operations should be decoupled from other operations, avoid conflicts of interest, and be subject to rigorous independent oversight. Enronitis and collusion must be avoided, even if it means that the costs are greater. Furthermore, the people involved need altruism, sufficient foresight to anticipate the risks, and a commitment to effectively combat those risks. At the very least, their backgrounds should be free of criminal convictions and other activities that would cast serious suspicions on their trustworthiness. In addition, legislators need to be able to see beyond the simplistic and palliative, to approaches that address the real problems. Above all, the entire populace must become more aware of the risks and the concerns outlined above, especially to the inherent combination of technology and policy issues.

This conclusion should not be a surprise. Overall, there are many risks that must be addressed. The old Latin expression "Caveat emptor" (Let the buyer beware) is even more timely today.

5. Principles for Developing Trustworthy Systems

Everything should be made as simple as possible—but no simpler.
Albert Einstein

Developing trustworthy systems with complex requirements is inherently a complex challenge. In general, simple solutions are hopelessly inadequate in such cases. On the other hand, enormously complex systems—even if they purport to be trustworthy—are likely to be unmanageable, from the perspective of developers, system administrators, application operators, and end-users.

Ideally, there should be some middle ground. In particular, the recommended approach, considered in Section 6, is to develop trustworthy systems as conceptually sound predictable compositions of simpler components, perhaps even with provably sound combinations of provably sound components.

In anticipation of that approach, a relevant set of principles can be helpful in increasing trustworthiness—if the principles are used intelligently as guidelines for system development and operation.

5.1 Saltzer–Schroeder Security Principles

The ten basic security principles formulated by Jerry Saltzer and Mike Schroeder [51] in 1975 are all still relevant today, in a wide range of circumstances. They are actually of broader interest than just with respect to security. For example, each one is also relevant to reliability, survivability, and human safety. In essence, these principles are summarized as follows (overly tersely), for present purposes:

- *Economy of mechanism*: Seek design simplicity (wherever and to whatever extent it is effective).
- *Fail-safe defaults*: Deny accesses unless explicitly authorized (rather than permitting accesses unless explicitly denied).
- *Complete mediation*: Check every access, without exception.
- *Open design*: Do not assume that design secrecy will enhance security.
- *Separation of privileges*: Use separate privileges or even multiparty authorization (e.g., two keys held by different entities) to reduce misplaced trust.
- *Least privilege*: Allocate minimal (separate) privileges according to need-to-know, need-to-modify, need-to-delete, need-to-use, and so on. The existence of powerful mechanisms such as *superuser* is inherently dangerous.
- *Least common mechanism*: Minimize the amount of mechanism common to more than one user and depended on by all users.
- *Psychological acceptability*: Strive for ease of use and operation—for example, with easily understandable and forgiving interfaces.
- *Work factors*: Make cost-to-protect commensurate with threats and expected risks.
- *Recording of compromises*: Provide nonbypassable tamper-resistant and tamper-evident audit trails of evidence.

These are of course basic guidelines, not hard-and-fast rules—especially in light of some potential mutual contradictions. Two fundamental caveats must be recognized. First, each principle by itself may be useful in some cases and not in others. Second, when taken in combinations, groups of principles are not necessarily all reinforcing; indeed, they may in some cases conflict with one another. Consequently, development must consider appropriate use of each principle in the context of the overall effort. Examples of a principle having both positive and negative aspects are scattered through the following discussion.

The Saltzer–Schroeder principles grew directly out of the MIT/Honeywell/BellLabs Multics experience (e.g., [40]) begun in 1965 and discussed further later in this section. Each of these principles has taken on almost mythic proportions among the security elite, and to some extent buzzword cult status among many fringe parties. Therefore, we do not explain each principle in detail—although considerable depth of discussion is needed for successful application of each principle. Careful reading of the Saltzer–Schroeder paper [51] is recommended if it is not already a part of your library. Matt Bishop's security books [7,8] are also useful in this regard, placing the principles in a more general context.

Various caveats are considered in Section 12.

TABLE I
APPLICABILITY OF SALTZER–SCHROEDER PRINCIPLES

Principle	Trustworthiness	Assurance
Economy of mechanism	is a vital aid to sound design. Exceptions must be handled completely.	can simplify local analysis.
Fail-safe defaults	simplifies design, use, operation, maintenance.	can simplify analysis.
Complete mediation	is vital, but beware of compromise from below.	can simplify analysis locally.
Open design	Secret designs do not preclude compromise. Open design can inspire stronger system security.	Open designs do not preclude compromise. Open design encourages independent analysis.
Separation of privileges	avoids many types of common flaws.	focuses analysis more precisely.
Least privilege	limits effects of flaws; simplifies operation.	focuses analysis more precisely.
Least common mechanism	avoids certain common flaws.	modularizes analysis.
Psychological acceptability	is relevant to usability and operations.	Ease of use is helpful, must anticipate crises.
Work factors	are misleading if systems can be compromised from outside/within/below.	give a false sense of security if unexpected compromises are ignored.
Compromise recording	is an after-the-fact diagnostic and deterrent.	is only an indirect contributor.

Table I summarizes how each of the Saltzer–Schroeder principles can contribute to the goals of trustworthiness and assurance, particularly with respect to security, reliability, and other survivability-relevant requirements. Intriguingly, most of these principles can also be helpful in enhancing composability.

In particular, complete mediation, separation of privileges, and allocation of least privilege are beneficial to composability and trustworthiness. Open design can contribute significantly to composability, when subjected to internal review and external criticism. (See Section 6.) However, conflicts persist about the importance of open design with respect to trustworthiness, with some people still clinging tenaciously to the notion that security by obscurity is sensible—despite risks of many flaws being so obvious as to be easily detected externally, even without reverse engineering. Indeed, the recent emergence of good decompilers for C and Java, along with the likelihood of similar reverse engineering tools for other languages, suggests that such attacks are becoming steadily more practical. Overall, the pretense of keeping a design se-

cret and the supposed unavailability of source code are realistically not significant deterrents, especially with ever-increasing skills among black-box system analysts. However, there are cases in which reliance on security by obscurity is unavoidable—as in the hiding of private and secret cryptographic keys, although the cryptographic algorithms and implementations can be public.

Fundamental to trustworthiness is the extent to which systems and networks can avoid being compromised by malicious or accidental human behavior and by events such as hardware malfunctions and so-called acts of God. In [35], we consider *compromise from outside, compromise from within*, and *compromise from below*, with fairly intuitive meanings. These notions appear throughout this chapter.

In theory, there are various cases where certain weak links can be avoided (such as zero-knowledge protocols that can establish a shared key without any part of the protocol requiring secrecy, Byzantine algorithms, and k-out-of-n cryptography), although in practice they may be undermined by compromises from below (involving trusted and supposedly trustworthy insiders subverting the underlying operating systems) or from outside (involving penetrations of the operating systems and masquerading as legitimate users).

From its beginning, the Multics development was strongly motivated by a set of principles—some of which were originally stated by Ted Glaser and Neumann in the first section of the first edition of the Multics Programmers' Manual in 1965. (See http://multicians.org.) It was also driven by extremely disciplined development. For example, no coding effort was begun until a written specification had been approved by the Multics advisory board; also, with just a few exceptions such as low-level device drivers, all the code was written in a subset of PL/I just sufficient for the needs of Multics, for which the first compiler (early PL, or EPL) had been developed by Doug McIlroy and Bob Morris.

In addition to the Saltzer–Schroeder principles, further insights on principles and discipline relating to Multics can be found in a paper by Fernando Corbató, Jerry Saltzer, and Charlie Clingen [12] and in Corbató's Turing lecture [11].

5.2 Further Principles

An earlier view of principled system development was given by Neumann in 1969 [33], relating to what is often dismissed as merely "motherhood"—but which in reality is both profound and difficult to observe in practice. The principles under consideration in that paper included automatedness, availability, convenience, debuggability, documentedness, efficiency, evolvability, flexibility, forgivingness, generality, maintainability, modularity, monitorability, portability, reliability, simplicity, and uniformity. Some of those attributes indirectly affect security and trustworthiness, whereas others affect the acceptability, utility, and long-term future of systems.

Considerable discussion in [33] was also devoted to (1) the risks of local optimization and the need for a more global awareness of less obvious downstream costs of development (e.g., writing code for bad—or nonexistent—specifications, and having to debug really bad code), operation, and maintenance (see Section 8); and (2) the benefits of higher-level implementation languages (which prior to Multics were rarely used for the development of operating systems [11,12]).

In the context of developing predictably trustworthy systems, an expanded set of principles is listed below. Although most of them might seem more or less obvious to advanced developers, there are interpretations, hidden issues, and potential pitfalls for their successful implementation. As a result, a seemingly paradoxical situation arises: understanding and experience are required in order to make optimal use of the principles. Thus, the learning experience is essentially iterative.

- *Sound architecture*. Recognizing that it is better to avoid design errors early than to attempt to fix them later, composable architectures inherently capable of evolvable, maintainable, robust implementations are required. Furthermore, good interface design is as fundamental to good architectures as is their internal designs. Both the architectural structure and the architectural interfaces (particularly the visible interfaces, but also some of the internal interfaces that must be interoperable) can benefit from careful specification.
- *Abstraction*. The primitives at any given logical or physical layer should be relevant to the functions and properties of the objects at that layer, and should mask lower-layer detail where possible. Ideally, the specification of a given abstraction should be in terms of objects meaningful at that layer, rather than requiring lower-layer (e.g., machine-dependent) concepts. Abstractions at one layer can be related to the abstractions at other layers in a variety of ways, thus simplifying the abstractions at each layer rather than collapsing different abstractions into a more complex single layer. Particularly useful examples of abstraction include trustworthiness kernels, virtual machine monitors, and similar layered defenses.
- *Modularity*. Modularity relates to the characteristic of system structures in which different entities (modules) can be relatively loosely coupled and combined to satisfy overall system requirements, whereby a module could be modified or replaced as long as the new version satisfies the given interface specification. In general, modularity is most effective when the modules reflect specific abstractions and provide encapsulation within each module (see the next item).
- *Encapsulation*. Details that are relevant to a particular abstraction should be local to that abstraction and subsequently isolated within the implementation of that abstraction and the lower layers on which the implementation depends. One example of encapsulation involves information hiding—for example, keeping

internal state information inaccessible to the visible interfaces [41]. Another example involves masking the idiosyncrasies of physical devices from higher-layer system interfaces, and from the user interfaces as well.

- *Layered and distributed protection*. Protection (and generally defensive design for security, reliability, and so on) should be distributed to where it is most needed, and should reflect the semantics of the objects being protected. With respect to the reality of implementations that rely on—and perhaps pass through—entities of different trustworthiness, layers of protection are vastly preferable to flat concepts such as single sign-on (i.e., where only a single authentication is required). With respect to psychological acceptability, single sign-on has enormous appeal; however, it can leave enormous security vulnerabilities as a result of compromise from outside, from within, or from below, in both distributed and layered environments. Overall, psychological acceptability can conflict with other principles, such as complete mediation, separation of privileges, and least common privilege.
- *Constrained dependency for integrity*. Dependencies on less trustworthy entities should be avoided unless potential negative effects can be somehow confined or constrained. However, it is possible in some cases to surmount the relative untrustworthiness of mechanisms on which certain functionality depends—as in various types of trustworthiness-enhancing mechanisms (see [36]). In essence, do not trust anything on which you must depend—unless you are adequately satisfied with demonstrations of its trustworthiness or the ability to surmount its relative untrustworthiness. This intuitive extension of Biba's notion of multilevel integrity [6] is considered further in Section 6.
- *Architectural minimization of what must be trustworthy*. Appropriate trustworthiness should be situated where it is most needed, suitable to overall system requirements, rather than required uniformly across widely distributed components (with potentially many weak links) or totally centralized (with creation of a single weak link and forgetting other vulnerabilities). Trustworthiness is expensive to implement and to ensure. Thus, significant benefits can result from minimizing what has to be trustworthy. This principle can contribute notably to sound architectures. In combination with economy of mechanism, this provides avoidance of both bloatware and adverse dependence on less trustworthy components. For example, in some cases a simple end-to-end check can determine the presence of intermediate compromises and avoid the necessity of trusting everything else for integrity (apart from denial-of-service attacks).
- *Object orientation*. The OO paradigm bundles together abstraction, encapsulation, modularity of state information, inheritance (subclasses inheriting the attributes of their parent classes—e.g., for functionality and for protection), and

subtype polymorphism (subtype safety despite the possibility of application to objects of different types). This paradigm facilitates programming generality and software reusability, and if properly used can enhance software development. This is a contentious topic—for example, in that most of the OO methodologies and languages are sloppy with respect to inheritance.

- *Separation of policy and mechanism*. Statements of policy should avoid inclusion of implementation-specific details. Furthermore, mechanisms should be policy neutral where that can be advantageous in achieving functional generality. However, this principle must never be used in the absence of understanding about the range of policies that needs to be implemented. There is a temptation to avoid anticipating meaningful policies, deferring them until later in the development—and then discovering that the desired policies cannot be realized with the given mechanisms. This is a characteristic chicken-and-egg problem with abstraction.
- *Separation of duties*. In relation to separation of privileges, separate classes of duties of users and computational entities should be identified, so that distinct system roles can be assigned accordingly. Distinct duties should be treated distinctly, as in activities of system administrators, system programmers, and unprivileged users.
- *Separation of roles*. Concerning separation of privileges, the roles recognized by protection mechanisms should correspond in some readily understandable way to the various duties. For example, a single all-powerful superuser role intrinsically violates separation of duties, separation of roles, separation of privilege, and separation of domains. The separation of would-be superuser functions into separate roles (as in Trusted Xenix) is a good example of desirable separation. Once again (as with single sign-on), there is a potential conflict between principles: the monolithic superuser mechanism provides economy of mechanism, but violates other principles. In practice, all-powerful mechanisms are sometimes unavoidable, and sometimes even desirable despite the negative consequences (particularly if confined to a secure subenvironment). However, they should be avoided wherever reasonable.
- *Separation of domains*. Concerning separation of privileges, domains should be able to enforce separate roles. For example, a single all-powerful superuser mechanism is inherently unwise, and is in conflict with the notion of separation of privileges. However, separation of privileges is difficult to implement if there is inadequate separation of domains. Separation of domains can help enforce separation of privilege, but can also provide functional separation (as in the Multics ring structure, a kernelized operating system with a carefully designed kernel, a capability-based architecture, or a virtual-machine monitor). The prin-

ciple of least common mechanism is also somewhat related. It is desirable to avoid sharing of trusted multipurpose mechanisms, including executables and data, thereby minimizing the use of all-powerful mechanisms such as *superuser* and shared buffers (such as the historically seminal FORTRAN `common`). As one example of the flaunting of principles, exhaustion of shared resources provides a huge source of covert storage channels, whereas the natural use of a common calendar clock provides a source of covert timing channels.

- *Sound authentication*. Authentication is a pervasive problem. Nonbypassable authentication should be applicable to users, processes, procedures, and in general to any active entity or object. Authentication relates to evidence that the identity of an entity is genuine, that procedure arguments are legitimate, that types are properly matched when strong typing is to be invoked, and other similar aspects.
- *Sound authorization and access control*. Authorizations must be correctly and appropriately allocated, and nonsubvertible. Crude all-or-nothing authorizations are often riskful (particularly with respect to insider misuse and programming flaws). In applications for which user-group-world authorizations are inadequate, access-control lists and role-based authorizations may be preferable. Finer-grained access controls may be desirable in some cases, such as capability-based addressing and field-based database protection. However, knowing who has access to what at any given time should be relatively easy to determine.
- *Administrative controllability*. The facilities by which systems and networks are administered must be well designed, understandable, well documented, and sufficiently easy to use without inordinate risks.
- *Comprehensive accountability*. Well-designed and carefully implemented facilities are essential for comprehensive monitoring, auditing, interpretation, and automated response (as appropriate). Serious security and privacy issues must be addressed relating to the overall accountability processes and audit data.

Similar to the summary in Table I, the additional principles also tend to contribute to the goals of achieving composability, trustworthiness, and assurance.

At this point in the analysis, it should be no surprise that these and other principles can contribute in varying ways to security, reliability, survivability, and other -ilities. Furthermore, many of the principles and other "ilities" are linked. We cite just a few of the interdependencies that must be considered.

For example, authorization is of limited use without authentication, *whenever identity is important*. Similarly, authentication may be of questionable use without authorization. In some cases, authorization requires fine-grained access controls.

Least privilege requires some sort of separation of roles, duties, and domains. Separation of duties is difficult to achieve if there is no separation of roles. Separation of roles, duties, and domains each must rely on a supporting architecture.

The comprehensive accountability principle is particularly intricate, as it depends critically on many other principles being invoked. For example, accountability is inherently incomplete without authentication and authorization—without which traceback to the users or originating entities is doubtful. In many cases, monitoring may be in conflict with privacy requirements and other social considerations [16], unless extremely stringent controls are enforceable. Separation of duties and least privilege are particularly important here. All accountability procedures are subject to security attacks, and are typically prone to covert channels as well. Principles are becoming increasingly important, and are fundamental to Peter Denning's Great Principles project (http://cs.gmu.edu/cne/pjd/GP).

6. System Composition: Problems and Potentials

The challenge of developing systems with realistic trustworthiness requirements is inherently complex, despite persistent advice to keep it simple. However, consider the goal of building trustworthy systems using predictably sound compositions of well-designed components along with analysis of the properties that are preserved by, transformed by, or emerging from the compositions. Conceptually, that can greatly simplify and improve development. Indeed, composition is seemingly theoretically relatively straightforward to achieve—especially if we follow the guidance of David Parnas, Edsger Dijkstra, and others. Unfortunately, there is a huge gap between theory and common practice: system compositions at present are typically *ad hoc*, based on the intersection of potentially incompatible component properties, and dependent on untrustworthy components that were not designed for interoperability and whose behavior can undermine the compositions—often resulting in unexpected results and risks. In practice, it is particularly difficult to determine all potentially negative effects of compositions of arbitrary components that were not designed with composition explicitly in mind.

Composition is a concept that is meaningful with respect to many entities, including requirements, specifications, protocols, implemented components, and analytic results such as evaluations and formal proofs. In many cases, the composition of different entities may have unpleasant results.

Other problems may arise because of the order in which operations are carried out, even though the operations may be theoretically commutative or in some broader sense equivalent (perhaps producing different but nevertheless acceptable results).

For example, consider the combination of error-correcting coding (which adds redundancy), compression (which removes redundancy), and cryptography (which ideally makes meaningful content look essentially random). Compressing after encrypting makes little sense, because there is little apparent redundancy. Similarly, compressing after adding redundancy for error correction also makes little sense, because it vitiates the overall error correction. Thus, if such a combination were to be effective, compression should precede encryption, which then should be followed by error-correcting coding.

With regard to subsystem composition, the following are particular concerns.

- *Composability and compositionality*. A distinction is sometimes made between two concepts pertaining to composition. *Composability* relates to the predictability of the preservation or transformation of existing properties under composition. *Compositionality* refers to the predictability of properties that emerge as a result of compositions.
- *Inadequate requirements*. If stated requirements do not explicitly demand that subsystems and other components be developed in ways that encourage compatibility and interoperability, composability is likely to be difficult to achieve. Furthermore, poorly defined requirements are likely to hinder composability.
- *Nonexistent or inappropriate specifications*. If system and subsystem specifications do not adequately define the relationships among interfaces, inputs, internal state information and state transitions, outputs, and exception conditions, and if those specifications are oblivious to critical relationships with related functionality, determining to what extent composability is possible becomes much more difficult. Composition of underconstrained specifications is an inherent problem, because the extent to which the components compose is ill-defined; supposed demonstrations of composability may actually be meaningless. Overly constrained specifications (e.g., including unnecessarily low-level and possibly incompatible details) are also often an impediment to composability. Shared state information across components is a particular source of potential problems.
- *Properties that exist beyond what is defined by stated individual subsystem interface specifications*. Assuming the presence of meaningful specifications, inadequacies of the specifications and inconsistencies between specifications and implementations are characteristic problems. In general, specifications are always inherently incomplete with respect to defining what should *not* happen, even if they are fairly good at defining what should happen. (Abstraction is a very important technique for simplifying specifications, but it suppresses detail that may include undesirable aspects of behavior and may therefore negatively affect compositional properties.) In addition, programming languages and com-

pilers themselves provide very few if any guarantees that something beyond what is expected cannot occur. Examples include shared-buffer interactions and unanticipated information residues from one invocation of a subsystem to a subsequent or concurrent invocation of the same subsystem; buffer overflows and other cases of inadequate bounds checks and inadequate runtime validation; inadequate authentication; improper initialization and finalization; improper encapsulation, which can result in interference and other unexpected interactions; race conditions; covert channels; and intentionally planted Trojan horses. This list represents just the nose of the camel. All these problems can impair composability. As one example, various Windows operating systems are actually relatively modular (which is essential for orderly development), but the modules are not sufficiently encapsulated to prevent adverse effects resulting from composition.

- *Properties that manifest themselves only as a result of combinations of subsystems*. Examples include adverse *emergent* properties (i.e., disruptive or even constructive effects that are not evident in any of the individual subsystems but that arise only when the subsystems are combined); adverse feedback interactions between subsystems, such as infinite loops or dependence on functionality that is less trustworthy; emergent covert channels that do not exist in any of the subsystems in isolation; mutual incompatibilities in the interfaces—perhaps resulting from internal state interference; global failure modes resulting from local faults, as in the 1980 ARPANET collapse [48] and the 1990 AT&T long-distance collapse (e.g., see [34]); so-called "man-in-the-middle" attacks (which might alternatively be called untrustworthy interpositions), in which an interposer can simulate the actions of each component; and other failure modes that arise only in the overall system context. A fascinating noncomposability situation is noted in attempts to combine encryption with digital signatures [4]: signatures are composable with public-key cryptography, but *not* with symmetric cryptography, in which case security may break down. These impediments to composability can arise essentially everywhere throughout the development life cycle—for example, incompatibilities among different requirements and policies, undesirable interactions in specifications and implementations, and difficulties in reconfiguration and maintenance.
- *Multivendor and multiteam incompatibilities*. In the interest of having heterogeneous architectures that enable mixing and matching of alternative components, it may be desirable to use multiple system developers. However, incompatibilities among interface assumptions, the existence of proprietary internal and external interfaces, and extreme performance degradations resulting from the inability to optimize across components can all result in difficulties in composing components.

- *Scalability*. Composability typically creates many issues of scalability. For example, performance may degrade badly or nonpredictably as multiple subsystems are conjoined. Composability can lead to a wide range of expected performance implications—for example, linear, multiplicative, or exponential in the number of composed subsystems. In practice, even further degradations can result—for example, from design or implementation flaws or indirect effects of the composition, such as unrecognized dependence on substantively slow interactions. Obviously, infinite loops and standstill deadlocks ("deadly embraces") are limiting cases of degradation, and can arise as a result of subsystem compositions.
- *Human issues*. Above all, people are the ultimate source of many problems. The supposed "good guys" can accidentally have profoundly negative effects on composability, through poor system conception, inadequate requirements, lack of comprehensive and accurate specifications, bad software-engineering practice, misuse or bad choices of programming languages, badly managed development, and sloppy operational practice (for example). Insider "bad guys" can have various negative effects on the desired composability, such as installing Trojan horses during development, operation, and reconfiguration that impair interoperability and compromise security. Human activities can also directly impair enterprise interoperability [18]. Outsider "bad guys" are generally less likely to negatively affect composability externally, except as a result of penetrations (through which they effectively become bad insiders), subversion of the development process, tampering, and denials of service (often without any internal access required).

There are many desiderata for achieving predictably assured composition, relating to requirements, specifications, implementations, programming languages, configuration information, and analyses thereof. Several relevant issues are noted below.

- *Compatibility and interoperability*. Compatibility implies merely the ability to coexist within a common framework, whereas interoperability additionally implies the ability to work together without adverse side effects. Both are essential prerequisites for composability.
- *Web interoperability*. In recent years, considerable effort has been devoted toward establishing a common definition of a *Web portal* concept that would facilitate universal interoperability providing access to Web services. As one example, Michael Alan Smith [53] has proposed a hierarchical General Portal Model that attempts to unify seventeen different definitions from the literature. From the top, the layers address process interfaces (process identification, transformation), resource discovery (resource identification, resource location, resource binding), and network interfaces (security, network access). In this

context, a portal implies an "infrastructure providing secure, customizable, personalizable, integrated access to dynamic content from a variety of sources, in a variety of formats, *wherever it is needed*." Among other approaches is that of a service-oriented architecture (e.g., [24]).

- *Consistency and completeness of the interface specifications*. Externally discernible functional behavior should be precisely what is specified, implying bilateral consistency of behavior with respect to the functional specifications. That is, the subsystem must do what it is supposed to do, *and nothing else beyond what is specified*. However, because specifications are inherently incomplete, many system failures (in security, reliability, performance, and so on) can result from events that occur outside the scope of specifications and thus are undetectable by any analyses based on those specifications.
- *Independence of specification abstractions*. As noted above, abstraction can be an enormous aid to composability of specifications, as well as to assurance proofs. However, it is essential that the details not explicitly represented by each abstraction be independent of the details of other abstractions. Otherwise, composability will most likely be impaired. One elegant example of provable composability is seen in the orthogonality theorem of Chander, Dean, and Mitchell [9], which provides soundness and completeness proofs for a trust management kernel with a clean separation between authorization and structured distributed naming.
- *Timing and synchronization issues*. In general, Lamport-style safety properties (i.e., nothing bad happens) compose better than liveness properties (something good eventually happens with certainty) [25], but this boundary is blurred by the inclusion of timing constraints, which are technically safety properties, but generally not composable. It is also blurred by the existence of properties that are neither safety nor liveness—such as information flow. Furthermore, time (whether real time or relative time) is typically common to different abstractions, which is a reason that synchronization and timing constraints can present serious impediments to facile composition. For example, see Kopetz [23] on composability in the Time-Triggered Architecture.
- *Explicit state visibility and information hiding*. If a subsystem is stateless (i.e., it does not remember any of its own state information from one invocation to the next), then it is less likely to have adverse interactions when that subsystem is composed with other subsystems—although there are always issues such as noncommutativity of operations and interference during concurrent execution. In addition, nontrivial recovery, as in selective rollback, may be unnecessary. However, statelessness is often not a desirable goal—although stack disciplines effectively separate the internal state information from the subsystem itself and

simplify composability. Assuming that a subsystem is stateful (i.e., it retains at least some of its own state information from one incarnation to the next), there is a choice between the classical notion of information hiding and explicit external visibility of state information (which tends to make explicit any residues that might impair compositionality). On the other hand, because information hiding typically masks internal state information, it can hinder facile composability if there are any implicitly shared states. However, this should be detectable with sensible specifications and implementation. (For example, pointers, loosely bound aliases, and other indirect references tend to create problems.) Thus, the separation of common stateful entities can greatly facilitate composition. Information hiding is also very desirable for other reasons, including isolation, security, system integrity, and tamper resistance.

One interesting historical approach is found in the formal specifications of SRI's Provably Secure Operating System (PSOS [19,38,39]), in which certain state information is hidden but from which the state information that is explicitly visible at the module interface is derived. Because hidden state information could not be accessed outside of the module (information hiding), it could not be referenced in any other module specification. As a result, there can be no module state residues or other state information that can be accessible to other modules or subsequent invocations of the same module beyond what is explicitly declared as visible. This greatly increases the composability of modules and the analysis of potential interactions. It also rules out certain characteristic design flaws simply because it is impossible to represent them in the specifications, even accidentally! (Note that bad implementations can introduce bugs that are not definable in specifications.)

6.1 Other Manifestations of Composition

As noted at the beginning of this section, composition is not limited only to components. It has other manifestations as well.

- *Policy composition*. Serious problems can result when different policies are in conflict or otherwise do not compose properly—especially if that lack of composability is not discovered until much later in development. Furthermore, attempting to compose policies often results in emergent properties that are not evident from the constituent policies. For example, see work by Virgil Gligor et al. with respect to the composability of separation-of-duty policies [21] and application-specific security policies [20]. Gligor notes (among other things) that policy composability does not necessarily imply the usefulness of the resulting policies, and that existing compositionality criteria are not always realistic. Preventing denials of service is a particularly thorny policy; besides,

policies that do not address denials of service are inherently incomplete. Of considerable interest is work by Heiko Mantel relating to the general composability of secure system policies and components [28] (e.g., flow properties that are preserved under refinement [27]). Many past efforts are of particular interest to the research community, such as [2,29].

- *Protocol composition*. There is also ongoing work on protocol composability—for example, see [13]. An interesting research challenge might be to consider a particular collection of protocols (e.g., for authentication, encryption, and integrity preservation) and prove that they are mutually composable, subject to certain constraints; the proofs could also be extended to demonstrating that their modular implementations would be composable.
- *Proof composition*. A book on compositionality of proofs [15] is worth careful reading for anyone interested in formal verification and high assurance of systems.
- *Certification composition*. Rushby [49] has characterized some of the main issues relating to the modular certification of an aircraft that is derived from separate certification of its components, based on an extension of a formal verification approach. The crucial elements involve separation of assumptions and guarantees (based on "assume-guarantee reasoning") into normal and abnormal cases.

6.2 Approaches for Predictable Composition

The following approaches can enhance the likelihood of predictable compositions.

- *Dependency analysis*. In many systems, unrecognized interdependencies among different components can hinder composability. Similar comments are relevant to contradictory or otherwise incompatible interdependencies among policies, models, separately compiled software, and even proofs. Identifying such dependencies and removing them or otherwise neutralizing them would be a considerable aid to composability,
- *Constrained and guarded dependency strategies*. The principle of constrained dependency for integrity is introduced in Section 5. Deterministic linearization or other suitable prioritization of intersubsystem dependencies (such as a lattice ordering) can avoid many adverse dependency problems, such as often result from misguided locking strategies and search strategies, compatibility mismatches in system upgrades, and unanticipated distributed interactions. For example, in Dijkstra's THE system paper [17], the use of a linearly ordered hierarchical locking structure guaranteed that no deadly embraces could occur *between* two different layers of abstraction (although in subsequent years a

deadly embrace was occasionally discovered *within* a particular layer). As another example, Biba's multilevel integrity [6] (MLI) requires in essence that no computational entity (e.g., user, program, process, or data) may depend on any other entities that are deemed less trustworthy (i.e., that are potentially less highly trusted) with respect to integrity. In the broadened sense of dependence considered here, the strict lattice ordering of multilevel integrity attributes implied by Biba may be relaxed if any relative untrustworthiness can be masked by creative system architecture or otherwise transcended—as in the trustworthiness-enhancing mechanisms enumerated in [36] as well as other architectural approaches such as isolation kernels and virtual machine monitors. Also, see Abadi et al. [1] for a formalization of dependency.

- *Functional consistency among layers of abstraction*. The 1977 Robinson–Levitt paper [45] on hierarchical formal specifications introduced the concept of formal mappings between different layers of functional specifications that represent abstract implementations of each layer as a function of the lower layers. Formal proofs at one layer can be derived by using the mapping functions together with the formal specifications at appropriate layers. The relatively unsung Robinson–Levitt mapping analysis is actually quite far-reaching, and can be used directly to relate properties of a composed system to individual properties of its subsystems. As noted above with respect to correctness and completeness of interface specifications, this approach is of course limited by any incompleteness in the functional specifications and mapping functions. The Robinson–Levitt approach was part of the SRI Hierarchical Development Methodology (HDM) [46] used in the Provably Secure Operating System [19, 38,39] project in the 1970s. An extremely impressive new application of this approach in a modern setting has been developed by John Rushby and Rance DeLong [50], which uses the interpretation mechanism of SRI's current formal methods environment (PVS), and applies it to high-assurance separation kernels (which explicitly provide both isolation and controlled sharing) as well as virtual-machine architectures. An earlier informal application of explicit interlayer relationships is found in the analysis of the interlayer dependencies in the Honeywell/Secure Computing Corporation (SCC) LOgical Coprocessor Kernel (LOCK) [52]. (PSOS's type-enforcement was the precursor of several SCC systems, including the Sidewinder firewall.)
- *Operating system and programming language approaches*. Program modularity, recursive and nested procedure-call protocols, clean stack disciplines, and the absence of unintended residues can all greatly enhance composability. Virtualized multiprocessing and rigorously enforced virtual machine separation have considerable possibilities in enabling extremely efficient distributed processing by abstracting out many of the usual pitfalls, especially when distributed

across networked systems. There is an important role for sound programming languages that naturally enforce modular separation with abstraction and encapsulation, compilers that efficiently enforce the programming-language modularity and strong typing, systems that provide efficient interprocedure and interprocess control flow, and optimizing compilers that do not throw out the baby with the bathwater (e.g., by prematurely binding entities that need to remain separated until later, creating less easily analyzed object code, seriously impeding debugging, or compromising security separations provided by architectural encapsulations and programming languages). (However, well-implemented aggressive optimizers are less likely to violate security than programmers are.) As one example, SPARK (the SPADE Ada Kernel, based on the Southampton Program Analysis Development Environment) provides a language-based approach to improving security and safety. Correctness-preserving transformations that survive compilation and optimization are another approach with significant promise. In particular, optimizing compilers must be fairly farsighted not to compromise the integrity of source code in the context of its system execution, although careful modularity with abstraction and encapsulation can diminish some of those possible effects. An alternative approach to assuring the soundness of the optimization is the translation validation approach considered at NYU [55], in which a validation tool confirms that the object code produced by the optimizer is a correct translation of the source code.

- *Principled designs, implementations, and use*. As a summary of this section, the Saltzer–Schroeder principles and the further principles discussed above are potentially extremely beneficial to the attainment of security and—more generally—trustworthiness. Techniques particularly relevant to composability include abstraction, hierarchical layering, encapsulation, design diversity, composability, pervasive authentication, and access control, as well as administrative and operational controllability, pervasive accountability and recovery, separation of policy and mechanism, assignment of least privilege, separation of concerns, separation of roles, separation of duties, and separation of domains. The object-oriented paradigm also has some merit, especially strong typing. (However, the would-be inheritance of implementations without strict inheritance of specification subclasses tends to impede composability. Every subclass instance must meet the specifications of all its superclasses, or else all verifications of uses of the superclasses are unsupported.)

Several recent proceedings are worthy of consideration with regard to composable system architecture and software engineering [14,22,26,43].

7. A Crisis in Information System Security

Section 4 considers risks in trusting entities that might not actually be trustworthy. Nevertheless, flawed systems that can cause more security and reliability problems than they solve are in widespread use.

Untrustworthy mass-market software might be used so extensively for various reasons, even if the source code is proprietary and the vendor can arbitrarily download questionable software changes without user intervention. Sometimes this is a path of least resistance (with few perceived alternatives) or obliviousness. Or perhaps it has the appearance of saving money *in the short term*. In some cases it is mandated organizationally—ostensibly to simplify procurement, administration, and maintenance, or because of a desire to remain within the monolithic mainstream. Often security, reliability, and the risks of networking are considered less important, or there is a belief that the free market will provide a cure. But the simplest answer is probably "because it's there." However, irrespective of any reasons *why* people might be willing to use flawed software, in certain cases it might be wiser *not to use it* at all—especially where the risks are considerable.

In my fourth testimony (August 2001) in five years for committees of the US House of Representatives, I made the following statement—amplifying similar statements made in earlier years:

> "Although there have been advances in the research community on information security, trustworthiness, and dependability, the overall situation in practice appears to continually be getting worse, relative to the increasing threats and risks—for a variety of reasons. The information infrastructure is still fundamentally riddled with security vulnerabilities, affecting end-user systems, routers, servers, and communications; new software is typically flawed, and many old flaws still persist; worse yet, patches for residual flaws often introduce new vulnerabilities. There is much greater dependence on the Internet, for Governmental use as well as private and corporate use. Many more systems are being attached to the Internet all over the world, with ever increasing numbers of users—some of whom have decidedly ulterior motives. Because so many systems are so easily interconnectable, the opportunities for exploiting vulnerabilities and the ubiquity of the sources of threats are also increased. Furthermore, even supposedly standalone systems are often vulnerable. Consequently, the risks are increasing faster than the amelioration of those risks."

In many respects, the situation does not seem to be getting better. The continuing flurry of viruses, worms, and system crashes raises the level of disruption to users and institutions. The incessant flow of identified vulnerability reports and the further existence of flaws that are not widely known suggest serious problems. The continual needs for installing copious patches in mass-market software (and the iterative problems they sometimes cause) suggest that we are not converging. Putting

the blame on inadequate system administration seems fatuous. Various exploitations of flaws (such as worms and viruses) are further examples of endemic problems in vulnerable systems that can be exploited. Unfortunately, too many people seem to be oblivious to the underlying security problems.

Suggestions that we need to raise the bar may be countered with the argument that past attacks have not really been serious, and we have had few pervasive disasters of information system security, so why should we worry? Unfortunately, Murphy's Law suggests that if it can happen, it eventually will. Also, the general overemphasis on reducing short-term costs allows long-term concerns to be ignored. (See the next section.)

The Free Software/Open Source movements have been touted as possible alternatives to the inflexibilities of closed-source proprietary code. Indeed, GNU-Linux/BSD Unix variants are gaining considerable credibility, and are seemingly less susceptible to malware attacks. However, by itself, availability of source code is not a panacea, and sound software engineering is still essential. Even if an entire system has been subjected to extremely rigorous open evaluation and stringent operational controls, that may not be enough to ensure adequate behavior.

Many users and application developers have grown accustomed to flaky software, perhaps because they do not have to meet critical requirements (as in nuclear power control, power distribution, and flight and air-traffic control) and suffer no liability for disasters. Perhaps it is time to follow the adage of "Just Say No" to bad software, and to demand that software development be dramatically improved.

Many different approaches to software system development can be found in practice, such as object-oriented programming, aspect-oriented programming, agile software development, service-oriented architecture, design patterns, model-based design, event-driven architecture, clean-room development, extreme programming, formal methods, a long list of methodologies named after their progenitors, and so on. The discipline of these and other approaches can be very helpful, but trustworthiness demands much more than conventional software. Principled approaches are just one more step forward, and need to be coupled with sound development practices.

8. Optimistic Optimization

Many people (corporate executives, managers, developers, and so on) tend to ignore the long-term implications of decisions made for short-term gains, often based on overly optimistic or fallacious assumptions. In principle, much greater benefits can result from far-sighted vision based on realistic assumptions. For example, serious environmental effects (including global warming, water and air pollution, pesticide toxicity, and adverse genetic engineering) are largely ignored in pursuit

of short-term profits. However, conservation and environmental protection appear much more relevant when considered in the context of long-term costs and benefits. Furthermore, governments are besieged by intense short-sighted lobbying by special interests. Insider financial manipulations have serious long-term economic effects. Research funding has been increasingly focusing on short-term returns, to the detriment of the future.

Computer system development is a particularly frustrating example. Most system developers are unable or unwilling to confront life-cycle issues up front and in the large, although it is clear that up-front investments can yield enormous benefits later in the life cycle. In particular, defining requirements carefully and wisely at the beginning of a development effort can greatly enhance the entire subsequent life cycle and reduce its costs. This process should ideally anticipate all essential requirements explicitly, including (for example) security, reliability, scalability, and relevant application-specific needs such as evolvability, maintainability, usability, interoperability, and enterprise survivability. Many such requirements are typically extremely difficult to add once system development is well underway. Furthermore, certain types of requirements tend to change; thus, system architectures and interfaces should be relatively flaw-free and inherently adaptable without introducing further flaws. Insisting on principled software engineering (such as modular abstraction, encapsulation, and type safety), sensible use of sound programming languages, and use of appropriate support tools can significantly reduce the frequency of software bugs. All these up-front investments can also reduce the subsequent costs of debugging, integration, system administration, and long-term evolution—if sensibly invoked.

Consideration of the value of up-front efforts is a decades-old concept. However, it is often widely ignored or done badly, for a variety of reasons—such as short-term profitability, rush to market, lack of commitment to quality, lack of liability concerns, ability to shift late life-cycle costs to customers, inadequate education, experience and training, and unwillingness to pursue other than seemingly easy answers.

Overly optimistic development plans that ignore these issues tend to win out over more realistic plans, but can lead to difficulties later on—for developers, system users, and even innocent bystanders. The past is littered with systems that did not work properly and people who did not perform according to the assumptions embedded in the development and operational life cycles. (An example is seen in the mad rush to low-integrity paperless electronic voting systems with essentially no operational accountability, discussed in Section 9.) The lessons of past failures are widely ignored. Instead, we have a *caveat emptor* culture, with developers and vendors disclaiming all warranties and liability.

Many would-be solutions result in part from short-sighted approaches. Firewalls, virus checkers, and spam filters all have some benefits, but also some problems.

Firewalls would be more effective if they were not required to pass all sorts of executable content, such as ActiveX and JavaScript—but many users want those features enabled. (To date, viruses and worms have been rather benign, considering the full potential of really malicious code.) However, active content and malware would be much less harmful in a well-architected environment that could constrain executable content in some sort of "sandbox" that has rigidly limited effects.

Spammers seem to adapt very rapidly to whatever defenses they encounter. For example, they can test their current offerings against existing anti-spam products and adapt accordingly. Furthermore, domestic legislation may simply drive spammers offshore, without reducing the pain.

Better incentives are needed for far-sighted optimization, in larger contexts and over longer periods of time, with realistic assumptions and appropriate architectural flexibility to adapt to changing requirements. Achieving this will require many changes in research and development agendas, software and system development cultures, educational programs, laws, economy, commitment, and perhaps most important—in obtaining well-documented success stories to show the way for others. Particularly in critical applications, if it is not worth doing sensibly, perhaps it is not worth doing at all. But as David Parnas has said, let's not just preach motherhood; let's teach people how to be good mothers.

9. An Example: Risks in Electronic Voting Systems

The challenge of ensuring election system integrity provides a paradigmatic example of the considerations of the previous sections. The election process is an end-to-end phenomenon whose trustworthiness typically depends on the integrity of every step in the process. Unfortunately, each of those steps represents various potential weak links that can be compromised in many ways, accidentally and intentionally, technologically or otherwise; each step must be safeguarded from the outset and auditable throughout the entire process.

Irregularities reported in the 2000 and 2004 US national elections span the entire process, concerning voter registration, disenfranchisement and harassment of legitimate voters, huge delays in certain precincts, unbalanced distribution of voting equipment, absence of provisional ballots (required by the Help America Vote Act), mishandling of absentee ballots, and problems in casting and counting ballots for e-voting as well as other modes of casting and counting votes. Some machines could not be booted. Some machines lost votes because of programming problems, or recorded more votes than voters. Some touch-screen machines altered the intended vote from one candidate to another. The integrity of the voting technologies themselves is limited by weak evaluation standards, secret evaluations that are paid for

by the vendors, all-electronic systems that lack voter-verified audit trails and meaningful recountability, unaudited post-certification software changes, even runtime system or data alterations, and human error and misuse. (Gambling machines are held to much higher standards.) Other risks arise from partisan vendors and election officials. Furthermore, statistically significant divergences between exit polls and unaudited results created questions in certain states. All these concerns add to uncertainties about the integrity of the overall election processes.

With modern technology, the voting process could be more robust. Whether or not the potential weak links are mostly technological, the process can certainly be made significantly more trustworthy. Indeed, it seems to be better in many other countries than in the US; for example, Ireland, India, and the Netherlands seem to be taking integrity challenges seriously. As technologists, we should be helping to ensure that is the case—for example, by participating in the standards process or perhaps by aiding the cause of available source code and publicly accessible evaluations. However, the end-to-end nature of the problem includes many people whose accidental or intentional behavior can alter the integrity of the overall process, and thus creates many nontechnological risks.

With respect to computers used in elections, the principles outlined here would enable considerable improvements in trustworthiness if they were observed in practice. For example, architecturally minimizing the parts of the total system that must be trusted would by itself be a huge improvement, thereby reducing the extent of the weak links. The same is true of the principle of separating policy and mechanism.

The importance of understanding the idiosyncrasies of mechanisms and human interfaces, and indeed understanding the entire process, is illustrated by the 2000 Presidential election—with respect to hanging chad, dimpled chad, uncleaned chad slots, butterfly-ballot layouts, and the human procedures underlying voter registration and balloting. Clearly, the entire election process has vulnerabilities, including the technology and the surrounding administration. Looking into the future, a new educational problem will arise if preferential balloting becomes more widely adopted, whereby preferences for competing candidates are prioritized and the votes for the lowest-vote candidate are iteratively reallocated according to the specified priorities. This concept has many merits, although it certainly further complicates ballot layouts and voter awareness!

Alternative approaches have been proposed to existing voting systems (which have typically been lever machines, optically scanned paper, and paperless unauditable direct-recording computer systems). In approximate order of increasing conceptual complexity, these include (with examples of each) paper-based systems (Ben Adida [3], David Chaum [10], Ron Rivest [44]), cryptographic solutions (Andy

Neff [31], Josh Benaloh [5]), and voter-verified paper audit trails (VVPATs) (as an add-on for existing all-electronic systems, as proposed by Rebecca Mercuri [30]).

The VVPAT approach attempts to overcome the lack of integrity in existing direct-recording systems, but creates further complexity in the process. It is primarily a short-term fix to the current situation, in which proprietary software and proprietary evaluations against inherently incomplete voluntary standards provide relatively little system integrity. Cryptographic approaches require considerable care in design, analysis, implementation, and assurance, but also have the potential to avoid paper records—if the end-to-end systems could be made sufficiently trustworthy. On the

TABLE II
APPLICABILITY OF PRINCIPLES TO ELECTIONS

Principle	Computerization	Human Procedures
Economy of mechanism (+ sound architecture)	Simplistic mechanisms are dangerous. Complex systems need extensive analysis and predictable composability.	Operational simplicity is essential for poll workers. Perspicuous risk assessment is desirable throughout.
Fail-safe defaults	can simplify operation and improve trustworthiness.	can mitigate against insider misuse, fraud, and errors.
Complete mediation	can be useful in principled system architectures.	Weakness in depth requires end-to-end oversight.
Open design (+ openness generally)	Proprietary closed-source software and evaluations are inherently suspect.	Diverse oversight is essential throughout the entire process, especially over weak links.
Separation of privileges	can reduce insider misuse, human error, system failures.	can avoid centralized vested control throughout.
Least privilege (+ reduced needs for trust) (+ constrained dependency)	eschews root-privilege misuse, bootload subversion, trusting untrustworthiness. Avoid software built on subvertible underpinnings.	is important throughout the entire process, obviates allocation of excessive trust. Do not trust potentially untrustworthy people.
Least common mechanism	Beware of common flaws and common fault modes.	Separate roles may simplify assurance.
Psychological acceptability	Voter- and official-friendly systems can be helpful.	Ease of use and operation can help if it is not simplistic.
Work factors (+ objective risk analyses)	must encompass all systems, not just limited to strength of cryptography/authentication.	must encompass the entire process end-to-end, including developers and operators.
Compromise recording (+ pervasive monitoring)	Tamper-resistant audit trails are critical whenever results are suspect, and may help disincentivize fraud.	Manual procedures need oversight against compromise from outside/within/below, not just when suspicions arise.

other hand, the proposed paper-based systems have considerable conceptual simplicity and avoid many of the integrity problems of computer-based systems. However, these approaches address primarily only the vote recording and counting parts of the election process. End-to-end integrity must also include voter registration, voter and vote authentication, and postprocessing.

Table II tersely summarizes the potential relevance of principles (left column) for overall system architectures and development, for both computer-related systems (middle column) and operational procedures (right column) throughout the election process. The table represents a broadening of the Saltzer–Schroeder principles to address some additional aspects (suggested by what follows the *plus* sign in parentheses in the *principle* column). It thus generalizes the original principles somewhat to include related concepts discussed herein that reach farther than what was originally covered by Saltzer and Schroeder. It also reflects on the fact that these principles are relevant to trustworthiness overall—including (for example) many types of human errors and system failures that are not just limited to security issues. However, it does not remind the reader that this set of principles is only part of what is needed. Ultimately, expertise, experience, and good judgment are essential.

10. The Need for Risk Awareness

Around the world, our lives are increasingly dependent on technology. What should be the responsibilities of technologists regarding technological and nontechnological issues?

- Solving real-world problems often requires technological expertise as well as sufficient understanding of a range of economic, social, political, national, and international implications. Although it may be natural to want to decouple technology from the other issues, such problems typically cannot be solved by technology alone. They need to be considered in the broader context.
- Although experts in one area may not be qualified to evaluate detailed would-be solutions in other areas, their own experience may be sufficient to judge the conceptual merits of such solutions. For example, demonstrable practical impossibility or fundamental limitations of the concept, or the existence of serious conflicts of interest of the participants, or an obvious lack of personal and system-wide integrity are causes for concern.
- Ideally, we need more open, holistic, and interdisciplinary examinations of the underlying problems and their proposed solutions. (For example, see [37].)

Many concerns arise in important computer-related application areas, such as aviation, health care, defense, homeland security, law enforcement and intelligence—with similar conclusions. In each area, a relevant challenge is that of developing

and operating end-to-end trustworthy environments capable of satisfying stringent requirements for human safety, reliability, system integrity, information security, and privacy, in which many technological and nontechnological issues must be addressed throughout the computer systems and operational practices. Overall, technologists need to provide adequate trustworthiness in our socially important information systems, by technological and other means. Research and development communities internationally have much to offer in achieving trustworthy computer-communication systems. However, they also have the responsibility of being aware of the other implications of the use of these systems.

A deeper knowledge of fundamental principles of computer technology and their implications will be increasingly essential in the future, for a wide spectrum of individuals and groups, each with its own particular needs. Our lives are becoming ever more dependent on understanding computer-related systems and the risks involved. Although this may sound like a meta-motherhood statement, wise implementation of motherhood is decidedly nontrivial—especially with regard to risks.

Computer scientists who are active in creating the groundwork for the future need to better understand system issues in the large, especially the practical limitations of theoretical approaches. System designers and developers need broader and deeper knowledge—including those people responsible for the human interfaces used in inherently riskful operational environments; interface design is often critical. Particularly in those systems that are not wisely conceived and implemented, operators and users of the resulting systems also need an understanding of certain fundamentals. Corporation executives need an understanding of various risks and countermeasures. In each case, knowledge must increase dramatically over time, to reflect rapid evolution. Fortunately, the fundamentals do not change as quickly as the widget of the day, which suggests that pervasive emphasis on education and ongoing training is needed with respect to the concepts of this chapter.

An alternative view suggests that many technologies can be largely hidden from view, and that people need not understand (or indeed, might prefer not to know) the inner workings. For example, David Parnas's early papers on abstraction, encapsulation, and information hiding are important in this regard. Although masking complexity is certainly possible in theory, in practice we have seen too many occasions (for examples, see the ACM Risks Forum archives) in which the occurrence of inadequately anticipated exceptions resulted in disasters. The complexities arising in handling exceptions apply ubiquitously, to defense, medical systems, transportation systems, personal finance, security, to our dependence on critical infrastructures that can fail—and to anticipating the effects of such exceptions in design, implementation, and operation.

Thus, computer-related education is vital for everyone. The meaning of the Latin word "educere" (to educate) is literally "to lead forth." However, in general, many

people do not have an adequate perception of the risks and their potential implications. When, for example, the information media tell us that air travel is safer than automobile travel (on a passenger-mile basis, perhaps), the comparison may be less important than the concept that both could be significantly improved. When we are told that electronic commerce is secure and reliable, we need to recognize the cases in which it is not.

With considerable foresight and wisdom, Vint Cerf has repeatedly said that "The Internet is for Everyone." The Internet can provide a fertile medium for learning for anyone who wants to learn, but it also creates serious opportunities for the unchecked perpetuation of misinformation and counterproductive learning that will need to be unlearned.

In general, we learn what is most valuable to us from personal experience, not by being force-fed lowest-common-denominator details. In that spirit, it is important that education, training, and practical experiences provide motivations for true learning. For technologists, education needs to have a pervasive systems orientation that encompasses concepts of software and system engineering, security, and reliability, as well as stressing the importance of suitable human interfaces. For everyone else, there needs to be much better appreciation of the sociotechnical and economic implications—including the risks issues. Above all, a sense of vision of the bigger picture is perhaps what is most needed.

11. Risks of Misinformation

The problems of online misinformation are evidently worsening, because of the growth of the Internet and our ever increasing dependence on online systems. Information technology is a double-edged sword—perhaps even more so than many other technologies. In the hands of enlightened individuals, institutions, and governments, its use can be enormously beneficial. In other hands, it can be detrimental. Unfortunately, the dichotomy is often in the eye of the beholder, perhaps depending on one's objectives (e.g., personal financial gains, corporate profits, global economic well-being, politics, privacy, and environmental concerns).

Given a collection of online information, many people behave as if it is inherently authentic and accurate. This myth applies not only to websites, but also to many types of special-purpose databases, such as those found in law enforcement, motor vehicle departments, medicine, insurance, social security, credit information, and homeland security. We have seen many cases in which misinformation (e.g., false flight data, erroneous medical records, unentered acquittals, or tampered files) has resulted in serious consequences. The same is true of imprecise information (e.g., resulting in

false arrests, or affecting everyone with a particular name such as "David Nelson" who attempts to board an airplane).

Although an individual can occasionally observe that personal information about one's self is incorrect, more typically such erroneous information is hidden from the individual in question, possibly in diversely inaccurate versions. Overall, it is usually impossible for one to ensure that all such instances are correct—especially when mirrored in unknown sites all over the world. Furthermore, it is difficult to determine whether or not online information about anything else is authoritative. Worse yet, the volume of questionable information is growing at an extraordinary rate, and attempts to update substantive misinformation often have little effect—especially with the persistence of incorrect cached versions.

We increasingly rely on the Internet for many purposes, including education and enlightenment, irrespective of whether the sources are accurate. Oft-repeated overly simplistic sound-bite mantras seem to be popular. Furthermore, some people seem eager to waste time and energy that could be better spent elsewhere—or to drop out. There is a tendency for entrenched positions to remain fixed. Are we losing our ability to listen openly to other views and engage in constructive thought?

Another problem involves the inaccessibility of vital information. We seem to have evolved into a mentality of "If it is not on the Internet, it does not exist." Even though there are many more data bytes available today than ever before, search engines reportedly find only a small percentage of those pages, almost none of the database-driven dynamic Web pages, and very little of what is in most public libraries. Copyright restrictions and proprietary claims further limit what is available. For example, professional society digital libraries tend to be accessible only to those members who pay to subscribe. Furthermore, overzealous filtering blocks many authoritative sources of information. Are our education and information gathering suffering from a lowest-common-denominator process?

The propagation of misinformation has long been a problem in conventional print and broadcast media, but represents another problem that is exacerbated by the speed and bandwidth of the Internet. In general, widely held beliefs in supposedly valid information tend to take on lives of their own as urban myths; they tend to be trusted far beyond what is reasonable, even in the presence of well-based demonstrations of their invalidity.

In the face of such rampant misinformation, the truth can be difficult to accept, partly because it can be so difficult to ascertain, partly because it can seem so starkly inconsistent with popular misinformation, and partly because people want to believe in simple answers. Thus, we are revisiting classical problems that might now be considered as E-Epistemology, involving the nature and fundamentals of online knowledge—especially with reference to its limits and validity. However, there are some possible remedies, such as epistemic educational processes that teach us

how to evaluate information objectively. For websites, this might entail examining who are the sponsors, what affiliations are implied, where the information comes from, whether multiple seemingly reinforcing items all stem from the same incorrect source, whether purported website security and privacy claims are actually justified, and so on.

12. Boon or Bane?

Predicting the long-term effects of computers is both difficult and easy: it is easy to predict the future (often mistakenly), but very difficult to be correct. Here are some suggested possible visions of the future.

• Computers play an increasing role in enabling and mediating communication between people. They have great potential for improving communication, but there is a real risk that they will simply overload us, keeping us from really communicating. We already receive far more information than we can process. A lot of it is noise. Will computers help us to communicate or will they interfere?

• Computers play an ever-increasing role in our efforts to educate our young. In some countries, educators want to have computers in every school, or even one on every desk. Computers can help in certain kinds of learning, but it takes time to learn the arcane set of conventions that govern their use. Even worse, many children become so immersed in the cartoon world created by computers that they accept it as real, losing interest in other things. Will computers really improve our education, or will children be consumed by them?

• Computers play an ever increasing role in our war-fighting. Most modern weapon systems depend on computers. Computers also play a central role in military planning and exercises. Perhaps computers will eventually do the fighting and protect human beings. We might even hope that wars would be fought with simulators, not weapons. On the other hand, computers in weapon systems might simply make us more efficient at killing each other and impoverishing ourselves. Will computers result in more slaughter or a safer world?

• Information processing can help to create and preserve a healthy environment. Computers can help to reduce the energy and resources we expend on such things as transportation and manufacturing, as well as improve the efficiency of buildings and engines. However, they also use energy, and their production and disposal create pollution. They seem to inspire increased consumption, creating what some ancient Chinese philosophers called "artificial desires." Will computers eventually improve our environment or make it less healthy?

• By providing us with computational power and good information, computers have the potential to help us think more effectively. On the other hand, bad informa-

tion can mislead us, irrelevant information can distract us, and intellectual crutches can cripple our reasoning ability. We may find it easier to surf the Web than to think. Will computers ultimately enhance or reduce our ability to make good decisions?

• Throughout history, many people have tried to eliminate artificial and unneeded distinctions among people. We have begun to learn that everyone has much in common—men and women of all colors, races, and nationalities. Computers have the power to make borders irrelevant, to hide surface differences, and to help us overcome long-standing prejudices. However, they also facilitate the creation of isolated, antisocial groups that may spread hatred and false information. Will computers ultimately improve our understanding of other peoples or lead to more misunderstanding and hatred?

• Computers can help us to grow more food, build more houses, invent better medicines, and satisfy other basic human needs. They can also distract us from our real needs and make us hunger for more computers and more technology, which we then produce at the expense of more essential commodities. Will computers ultimately enrich us or leave us poorer?

• Computers can be used in potentially dangerous systems to make them safer. They can monitor motorists, nuclear plants, and aircraft. They can control medical devices and machinery. Because they do not fatigue and are usually vigilant, they can make our world safer. On the other hand, the software that controls these systems and the people involved may actually be untrustworthy. Bugs are not the exception; they are the norm. Will computers ultimately make us safer or increase our level of risk?

Much of the accumulated wisdom summarized in this chapter is not particularly new. But it is also not widely practiced. Many people are so busy advancing and applying technology that they do not look either back or forward. We should look back to recognize what we have learned about computer-related risks (e.g., [34]). We must look forward to anticipate the future effects of our efforts, including unanticipated combinations of seemingly harmless phenomena. Evidence over the past decades suggests we are not responding adequately to the challenges. Predilections for short-term optimization without regard for long-term costs abound. We must strive to make sure that we maximize the benefits and minimize the harm. Among other things, we must build stronger and more robust computer systems while remaining acutely aware of the risks associated with their use. Perhaps disciplined observance of the content of this chapter can help provide an impetus for the considerable culture change that is required for the development of trustworthy systems, networks, and enterprises in the future.

ACKNOWLEDGEMENTS

This chapter was written in part under National Science Foundation Grant Number 0524111. Sections 5 and 6 are based on a report sponsored by Douglas Maughan when he was the US Defense Advanced Research Projects Agency (DARPA) Program Manager for the Composable High-Assurance Trustworthy System (CHATS) program. Joshua Levy contributed some incisive comments on the final draft.

The author is very grateful to members of the ACM Committee on Computers and Public Policy, who over the years have been extraordinarily constructive in guiding the Inside Risks columns of the *Communications of the ACM*, which provided a springboard for some of the material here. In particular, Peter Denning, Jim Horning, David Parnas, Jerry Saltzer, and Lauren Weinstein have been especially helpful, as has Tom Lambert at *CACM*. In particular, Section 12 is based on a column that appeared in the March 2001 *CACM*, coauthored with David Parnas.

Section 3 was inspired by an article by Tim Batchelder, "An Anthropology of Air", *Townsend Letter for Doctors and Patients*, pp. 105–106, November 2005. "Because [air] is negative space, it is difficult to see the value in preserving it."

Lindsay Marshall at Newcastle has been extremely helpful in providing a nicely searchable archive facility for the ACM Risks Forum. (The official archives of all issues of the ACM Risks Forum since its inception in August 1985 can be found at http://www.risks.org, which indirects to http://catless.ncl.ac.uk/Risks/, and at ftp://www.sri.com/risks/.)

Will Tracz has consistently encouraged the author to contribute salient risks-related highlights to the *ACM SIGSOFT Software Engineering Notes*, ever since Will took over the editorship from Neumann's founding stint (1976–1993).

The citations given herein represent just the tip of an enormous literature iceberg. Additional references relevant to this chapter can easily be gleaned by searching the Web or by browsing my website PGN, March 2007.

REFERENCES

[1] Abadi M., Banerjee A., Heintze N., Riecke J.G., "A core calculus of dependency", in: *POPL '99, Proc. of the 26th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, Texas*, January 20–22, 1999, pp. 147–160.

[2] Abadi M., Lamport L., "Composing specifications", in: de Bakker J.W., de Roever W.-P., Rozenberg G. (Eds.), *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, *REX Workshop, Mook, The Netherlands, May–June 1989*, in: *Lecture Notes in Comput. Sci.*, vol. 230, Springer-Verlag, Berlin, 1989, pp. 1–41.

[3] Adida B., Neff C.A., "Ballot casting assurance", in: *Workshop on Electronic Voting Technology Workshop, Vancouver, BC, Canada, August 2006*, USENIX.

[4] An J.H., Dodis Y., Rabin T., "On the security of joint signature and encryption", in: *Advances in Cryptology*, *EUROCRYPT 2002, Amsterdam, The Netherlands*, in: *Lecture Notes in Comput. Sci.*, Springer-Verlag, Berlin, May 2002, pp. 83–107.
[5] Benaloh J., "Simple verifiable elections", in: *Workshop on Electronic Voting Technology Workshop, Vancouver, BC, Canada, August 2006*, USENIX.
[6] Biba K.J., "Integrity considerations for secure computer systems", Technical Report MTR 3153, The Mitre Corporation, Bedford, Massachusetts, June 1975. Also available from USAF Electronic Systems Division, Bedford, Massachusetts, as ESD-TR-76-372, April 1977.
[7] Bishop M., *Computer Security: Art and Science*, Addison–Wesley, Reading, MA, 2002.
[8] Bishop M., *Introduction to Computer Security*, Addison–Wesley, Reading, MA, 2004.
[9] Chander A., Dean D., Mitchell J.C., "Reconstructing trust management", *J. Computer Security* **12** (1) (January 2004) 131–164.
[10] Chaum D., "Secret-ballot receipts and transparent integrity: Improving voter confidence & electronic voting at polling places", Technical report, March 2002, http://www.chaum.org.
[11] Corbató F.J., "On building systems that will fail (1990 Turing Award Lecture, with a following interview by Karen Frenkel)", *Commun. ACM* **34** (9) (September 1991) 72–90.
[12] Corbató F.J., Saltzer J., Clingen C.T., "Multics: The first seven years", in: *Proc. of the Spring Joint Computer Conference, vol. 40*, *Montvale, New Jersey*, AFIPS Press, 1972.
[13] Datta A., Küsters R., Mitchell J.C., Ramanathan A., Shmatikov V., "Unifying equivalence-based definitions of protocol security", in: *Proc. of the ACM SIGPLAN and IFIP WG 1.7 Fourth Workshop on Issues in the Theory of Security, Oakland, California*, IEEE Computer Society, April 2004.
[14] de Boer F.S., et al. (Eds.), *Formal Methods for Components and Objects*, *4th International Symposium*, *Lecture Notes in Comput. Sci.*, vol. 4111, Springer-Verlag, Berlin, November 2005.
[15] de Roever W.-P., de Boer F., Hanneman U., Hooman J., Lakhnech Y., Poel M., Zwiers J., *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*, *Cambridge Tracts Theoret. Comput. Sci.*, vol. 54, Cambridge University Press, New York, NY, 2001.
[16] Denning D.E., Neumann P.G., Parker D.B., "Social aspects of computer security", in: *Proc. of the 10th National Computer Security Conference*, September 1987.
[17] Dijkstra E.W., "The structure of the THE multiprogramming system", *Commun. ACM* **11** (5) (May 1968).
[18] Faughn A.W., "Interoperability: Is it achievable?", Technical report, Harvard University PIRP report, 2001.
[19] Feiertag R.J., Neumann P.G., "The foundations of a Provably Secure Operating System (PSOS)", in: *Proc. of the National Computer Conference*, AFIPS Press, 1979, pp. 329–334, http://www.csl.sri.com/neumann/psos.pdf.
[20] Gligor V.D., Gavrila S.I., "Application-oriented security policies and their composition", in: *Proc. of the 1998 Workshop on Security Paradigms, Cambridge, England*, 1998.

[21] Gligor V.D., Gavrila S.I., Ferraiolo D., "On the formal definition of separation-of-duty policies and their composition", in: *Proc. of the 1998 Symposium on Security and Privacy*, *Oakland, California*, IEEE Computer Society, May 1998.
[22] Gorton I., et al. (Eds.), *Component-Based Software Engineering, 9th International Symposium*, *Lecture Notes in Comput. Sci.*, vol. 4063, Springer-Verlag, Berlin, June/July 2006.
[23] Kopetz H., "Composability in the time-triggered architecture", in: *Proc. of the SAE World Congress*, *Detroit, Michigan*, SAE Press, 2000, pp. 1–8.
[24] Krafzig D., Banke K., Slama D., *Enterprise SOA Service-Oriented Architecture Best Practices*, Prentice Hall, Upper Saddle River, NJ, 2004.
[25] Lamport L., "A simple approach to specifying concurrent program systems", *Commun. ACM* **32** (1) (January 1989) 32–45.
[26] Löwe W., et al. (Eds.), *Software Composition, 5th International Workshop*, *Lecture Notes in Comput. Sci.*, vol. 4089, Springer-Verlag, Berlin, March 2006.
[27] Mantel H., "Preserving information flow properties under refinement", in: *Proc. of the 2001 Symposium on Security and Privacy*, *Oakland, California*, IEEE Computer Society, May 2001, pp. 78–91.
[28] Mantel H., "On the composition of secure systems", in: *Proc. of the 2002 Symposium on Security and Privacy*, *Oakland, California*, IEEE Computer Society, May 2002, pp. 88–101.
[29] McCullough D., "A hookup theorem for multilevel security", *IEEE Trans. Software Engineering* **16** (6) (June 1990).
[30] Mercuri R., "Electronic vote tabulation checks and balances", PhD thesis, Department of Computer Science, University of Pennsylvania, 2001. http://www.notablesoftware.com/evote.html.
[31] Neff C.A., "A verifiable secret shuffle and its application to e-voting", in: *Proc. of the ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania*, November 2001, pp. 116–125.
[32] Neumann P.G., "Illustrative risks to the public in the use of computer systems and related technology, index to RISKS cases", Technical report, Computer Science Laboratory, SRI International, Menlo Park, California. Updated regularly at http://www.csl.sri.com/neumann/illustrative.html also in .ps and .pdf form for printing in a denser format.
[33] Neumann P.G., "The role of motherhood in the pop art of system programming", in: *Proc. of the ACM Second Symposium on Operating Systems Principles*, *Princeton, New Jersey*, ACM, October 1969, pp. 13–18, http://www.multicians.org/pgn-motherhood.html.
[34] Neumann P.G., *Computer-Related Risks*, ACM Press, New York, and Addison–Wesley, Reading, MA, 1995.
[35] Neumann P.G., "Practical architectures for survivable systems and networks", Technical report, Final Report, Phase Two, Project 1688, SRI International, Menlo Park, California, June 2000. http://www.csl.sri.com/neumann/survivability.html.
[36] Neumann P.G., "Principled assuredly trustworthy composable architectures", Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, December 2004. http://www.csl.sri.com/neumann/chats4.html, .pdf and .ps.

[37] Neumann P.G., "Holistic systems", *ACM Software Engineering Notes* **31** (6) (November 2006) 4–5, http://www.csl.sri.com//neumann/holistic.pdf.

[38] Neumann P.G., Boyer R.S., Feiertag R.J., Levitt K.N., Robinson L. , "A provably secure operating system: The system, its applications, and proofs", Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, May 1980, 2nd edition, Report CSL-116.

[39] Neumann P.G., Feiertag R.J., "PSOS revisited", in: *Proc. of the 19th Annual Computer Security Applications Conference (ACSAC 2003), Classic Papers section*, *Las Vegas, Nevada*, IEEE Computer Society, December 2003, pp. 208–216, http://www.acsac.org/ and http://www.csl.sri.com/neumann/psos03.pdf.

[40] Organick E.I., *The Multics System: An Examination of Its Structure*, MIT Press, Cambridge, MA, 1972.

[41] Parnas D.L., "On the criteria to be used in decomposing systems into modules", *Commun. ACM* **15** (12) (December 1972).

[42] Petroski H., *To Engineer Is Human: The Role of Failure in Successful Design*, St. Martin's Press, New York, 1985.

[43] Reussner R.H., et al. (Eds.), *Architecting Systems with Trustworthy Components*, *International Seminar, Dagstuhl, Germany*, *Lecture Notes in Comput. Sci.*, vol. 3938, Springer-Verlag, Berlin, December 2004.

[44] Rivest R.L., "The threeballot voting system", Technical report, MIT, Cambridge, MA, October 2006.

[45] Robinson L., Levitt K.N., "Proof techniques for hierarchically structured programs", *Commun. ACM* **20** (4) (April 1977) 271–283.

[46] Robinson L., Levitt K.N., Silverberg B.A., *The HDM Handbook*, Computer Science Laboratory, SRI International, Menlo Park, California, June 1979 (three volumes).

[47] Rochlis J.A., Eichin M.W., "With microscope and tweezers: The Worm from MIT's perspective", *Commun. ACM* **32** (6) (June 1989) 689–698.

[48] Rosen E., "Vulnerabilities of network control protocols", *ACM SIGSOFT Software Engineering Notes* **6** (1) (January 1981) 6–8.

[49] Rushby J., "Modular certification", Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, June 2002.

[50] Rushby J.M., DeLong R., "Compositional assurance for security: A MILS integration protection profile", Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, December 2006.

[51] Saltzer J.H., Schroeder M.D., "The protection of information in computer systems", *Proc. IEEE* **63** (9) (September 1975) 1278–1308.

[52] Saydjari O.S., Beckman J.M., Leaman J.R., "LOCKing computers securely", in: *10th National Computer Security Conference, Baltimore, Maryland*, 21–24 September 1987, pp. 129–141;
Reprinted in Turn R. (Ed.), *Advances in Computer System Security*, vol. 3, Artech House, Dedham, Massachusetts, 1988.

[53] Smith M.A., "Portals: Toward an application framework for interoperability", *Commun. ACM* **47** (10) (October 2004) 93–97.

[54] Spafford E.H., "The Internet Worm: Crisis and aftermath", *Commun. ACM* **32** (6) (June 1989) 678–687.
[55] Zuck L., Pnueli A., Fang Y., Goldberg B., "VOC: A translation validator for optimizing compilers", in: *Electronic Notes in Theoretical Computer Science*, 2002.

Author Index

Numbers in *italics* indicate the pages on which complete references are given.

Subject Index

B

C

F

M

P

This page intentionally left blank

Contents of Volumes in This Series

Volume 45

Volume 46

Volume 47

Volume 52

Volume 53

Volume 54

Volume 55

Volume 56

Volume 57

This page intentionally left blank