Specifying a Knowledge Management System

Graham Durant-Law

graham@durantlaw.info

Abstract*: This paper proposes a methodology to decompose organisation or enterprise complexity so that all requirements for a knowledge management system are considered. The technique builds upon soft system methodologies and 'hard' system engineering approaches. It begins with the assumption that objectives or outcomes are achieved in an environment that produces or has constraints. These constraints either restrict the availability of solutions or restrict how solutions can be employed. The solutions are real things in the real world, and perform tasks to achieve objectives.*

Keywords*: Knowledge System Design Methodology, user requirements, system requirements.*

Introduction

Aristotle can be credited with the idea of '*the whole is greater than the sum of its parts*,' however contemporary systems thinking is credited to Ludwig von Bertalanffy (Blauberg *et al.*, 1973). A system is an organised complexity of components that together form a unitary whole and whose interactions reduce local entropy (Hitchins, 2003). Systems can be hard or soft, and open or closed. A hard system is clearly defined with tangible boundaries and evident purpose. An example is a computer. A soft system is complex, poorly defined, with intangible boundaries, and often without singular purpose (Hitchins, 2003). The Australian Health System is an example of a soft system. A closed system is unable to interchange with any other system. On the other hand an open system is free to interchange with other systems (Hitchins, 2003).

A knowledge management system is inherently a soft open system with open boundaries. This means that boundaries are permeable and difficult to position. What may be useful to one person in one part of an organisation may be useless to someone else in another department. Any knowledge management initiative must therefore establish clear achievable goals that deliver benefits to the organisation, or a sub-set of the organisation, and take into account user and stakeholder requirements. Furthermore it must bring together the three essential components described in the literature – people, process and technology[1].

[1] See for example, (Cho *et al.*, 2000); (Delong & Fahey, 2000); (Davenport & Prusak, 1998); (Kannegieter, 2001); (Soo *et al.*, 2001); and (Tiwana, 2002).

The key to success is that a knowledge management system must be useful and solve a problem, or problems. This means that the road to success lies in careful planning, using a systems of systems[2] approach, and the generation of sound requirements for each sub-system (Stevens *et al.*, 1998). Indeed Standish Group studies have found that 21% of all project failures can be attributed to incomplete or changing requirements, followed by lack of user involvement, lack of resources, unrealistic expectations and lack of executive support (Standish Group, 2000). All this points to the need to have a sound methodology that allows the thorough development of system and user requirements, because requirements generate a clear understanding of what actually has to be delivered. This is common sense, but common sense is often circumvented when the process is difficult or complex. It is further compounded because *'there is no standard and generally agreed requirements engineering process'* (Damain *et al.*, 2003).

This paper proposes a methodology to decompose organisation or enterprise complexity so that all requirements for a knowledge management system are considered. The paper begins with a short discussion on requirements, then suggests a model, and finally uses a hypothetical example to illustrate how requirements might be specified for a knowledge management system.

Requirements

Requirements detail the attributes or specifications of a product or service, and are usually divided into user requirements and system requirements. User requirements describe what the user wants from a product or service, and are the start point for the system requirements. For example a requirement from a user perspective for a software interface might be something like *'the user shall be able to create graphic links to another module even if the other module is not visible'* (Stevens et al., 1998). One the other hand system requirements describe in detail the features the product or service must provide, and allow each part of the design to be tested. System requirements provide an abstract model of what the future system must do. They differ from user requirements in that they talk about the end product, and not just the results that are needed. Furthermore they must define how the system interfaces with the other systems around it (Stevens, 2002).

The characteristics of a good requirement are as follows (Stevens, 2002):

- it expresses a single, or atomic, stipulation in a complete sentence that is clear and unambiguous;
- conjunctions that make multiple requirements and let out clauses are avoided;
- the to be verb 'shall' is used consistently as the link between the subject and predicate;
- where the condition is less absolute, verbs such as 'should' are used;

[2] The systems of systems approach recognises that systems come together to form meta-systems, which fulfil functions greater than the coupled systems. It also recognises that systems rarely stand alone.

- where possible either the user or the result is clearly stated;
- it is modular and able to stand alone;
- it is not in conflict with other requirements; and
- it is verifiable, traceable, and feasible.

All this sounds simple, but in practice writing requirements for a soft system is difficult (Vencel, 1999). This arises for at least two reasons. Firstly, soft systems have boundaries that are difficult to define and involve uncertainty (Checkland, 1993b). Furthermore requirements analysis uses a reductionist approach with the depth of analysis correlating with the complexity of the problem. This approach assumes that a complex system can be decomposed into small components, and that each of these components has a complete set of specifications. It further assumes that by reassembling the components the complex system is reformed (Vencel, 1999). However, the dynamic nature of a soft system means that the relationships between different components and factors often change, which results in increasing complexity and a changing end-state (Shehata & Bowen, 1999).

Secondly, reality often differs from intent, or intent may result in unforseen outcomes. For example, a company may implement a knowledge management system. The intent may be that it provides a forum to pose and respond to questions, and to browse and seek knowledge. However, the reality might be that it is not used, because it is perceived as adding a step in an already under-resourced area. Often these human factors are difficult to predict, and consequently are completely overlooked in the requirements development process. This human dimension is particularly important because people have different perceptions of the world, meaning that a problem for one person is not seen as a problem to another (Shehata & Bowen, 1999). This means that in developing the requirements for a knowledge management system the views of all of the people in the organisation must be considered.

Given that 21% of all project failures can be attributed to incomplete or changing requirements, and that in practice writing requirements for a soft system is difficult, it follows that a disciplined approach and a sound methodology is required. The next section introduces one model that might be used to develop the requirements for a knowledge system initiative.

The Knowledge System Design Model

The Knowledge System Design Model (KSDM) is a hybrid of HolisTech's ™ Holistic Systems and Development Model[3], and is currently being used by the author to define the requirements for, and build, a knowledge management system for the Australian Defence Force Capability Systems Division. It is a hybrid model that builds upon soft system methodologies and 'hard' system engineering approaches[4].

[3] Patrick Byrne holds the copyright on the Holistic Systems and Development Model, which is the subject of a partially complete systems engineering PhD thesis. The Holistic Systems and Development Model was most recently used by Patrick and the author to analyse and propose changes to the RAN regulatory framework in 2002.

[4] See for example (Braithwaite *et al.*, 2002; Checkland, 1993a, 1993b; Rosenhead, 1993; Shehata & Bowen, 1999; Sparks, 1997; Staker, 2000, 2003; Stevens et al., 1998)

The KSDM is a disciplined methodology that utilises a very robust requirements development process, which at its completion defines a comprehensive solution set. Importantly it brings together people, process and technology, and facilitates the development of an architecture from a conceptual system statement through objectives, into the real cost drivers of any system, the solution space. It also provides auditable traceability from organisational goals and objectives to be achieved in selected environments, to the actual objects or artefacts in the real world that produce the required outputs.

The KSDM builds on the assumption that objectives or outcomes are achieved in an environment that produces or has constraints. These constraints either restrict the availability of solutions or restrict how solutions can be employed. The solutions are real thing in the real world, and perform tasks to achieve objectives. By dealing with real things in the real world complexity is immediately reduced. The KSDM is illustrated at Figure 1 and discussed below. Whilst the KSDM is shown as a cycle with a start and end point in practice it is an iterative process, with each space being examined many times as requirements in other spaces become apparent.

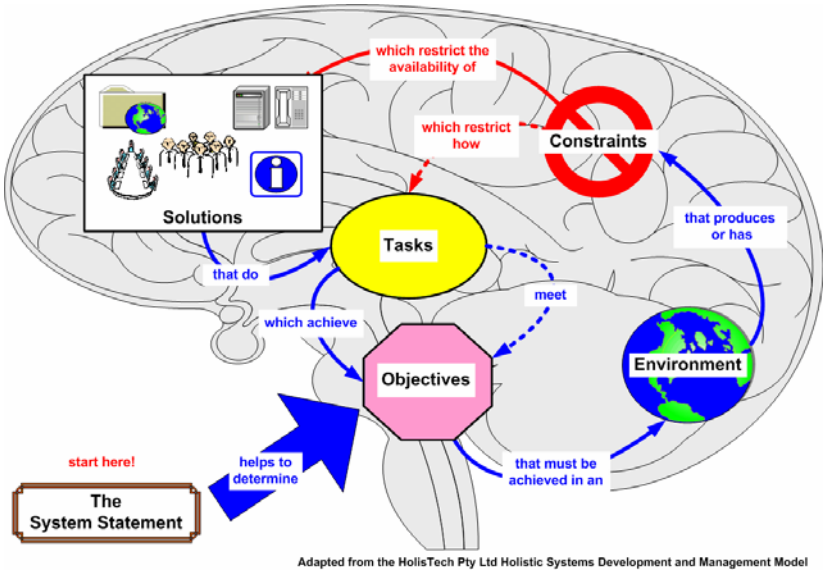

Figure 1: Knowledge System Design Methodology

System Statement

The system statement comprises top-level user requirements that provide the initial guidance to the development process, and set the scene for the development of processes within the system, and between other systems. It has three mandatory parts and an option fourth component. The first part is a broad statement of intent on what the system should achieve. The second part outlines some of the broad characteristics expected in that system. The third part is a statement of achievement, which provides some broad guidance on potential system components and processes, and the end state for the system. The optional component is a benchmark statement that directs the system architect to a tried and proven system in a comparable organisation.

Objectives

The objectives space provides a list of the outcomes expected from the implementation of the system. These may be articulated in the first instance by a series of identified problems, in which case the objectives are to solve those problems. It is also preferable that measures of effectiveness, the metric associated with objectives, are included for each objective. A measure of effectiveness is a metric that answers the question *'at what point will the objective be considered to be accomplished or satisfied?'* Objectives start to define the processes that are likely to be internal to the system.

Soft systems methodology tools such as mind-maps are useful to determine the top level objectives, and to provide a framework for future decomposition. Figure 2 shows an example of the first-cut of objectives captured using mind-mapping. Note that these objectives have not yet been expressed as requirements.

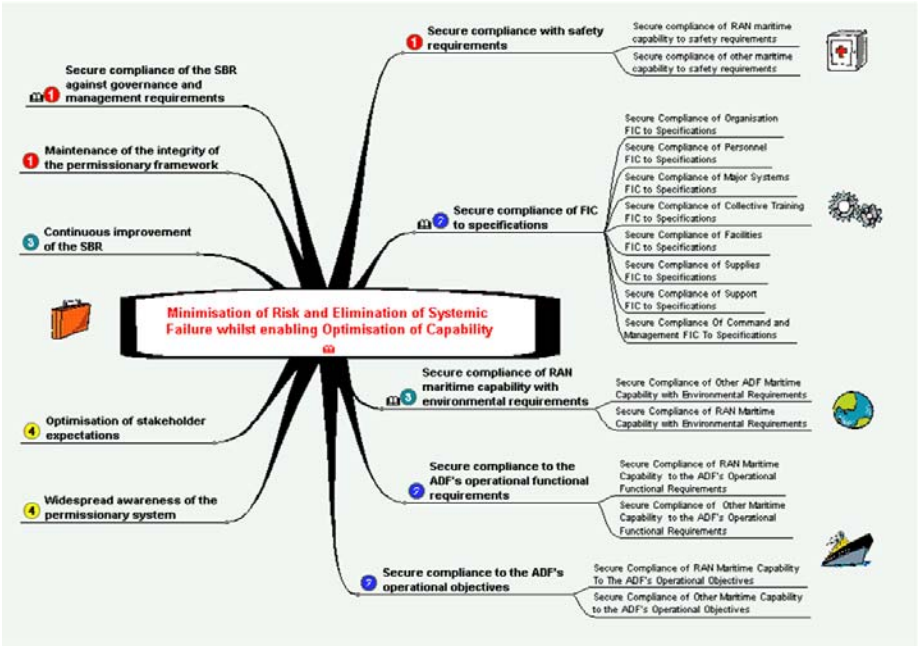

Figure 2: Mind-mapping Objectives[5]

Environment

No real-world system ever operates independently (Stevens et al., 1998). The environment space identifies the environment within which the system must operate, including the interfaces with other systems, and the positioning of the system boundary. Too often this step is done poorly. The product from this analysis is the interface control document. This document defines all the interactions between the systems, and provides a description of the organisational culture within which the system has to operate. For example it will specify whether or not the system is expected to operate in a dispersed or fragmented environment. Simple diagrams such as the one shown Figure 3 at enhance clarity.

[5] This diagram was developed by the author in a workshop that examined the objectives for a future RAN regulatory system.

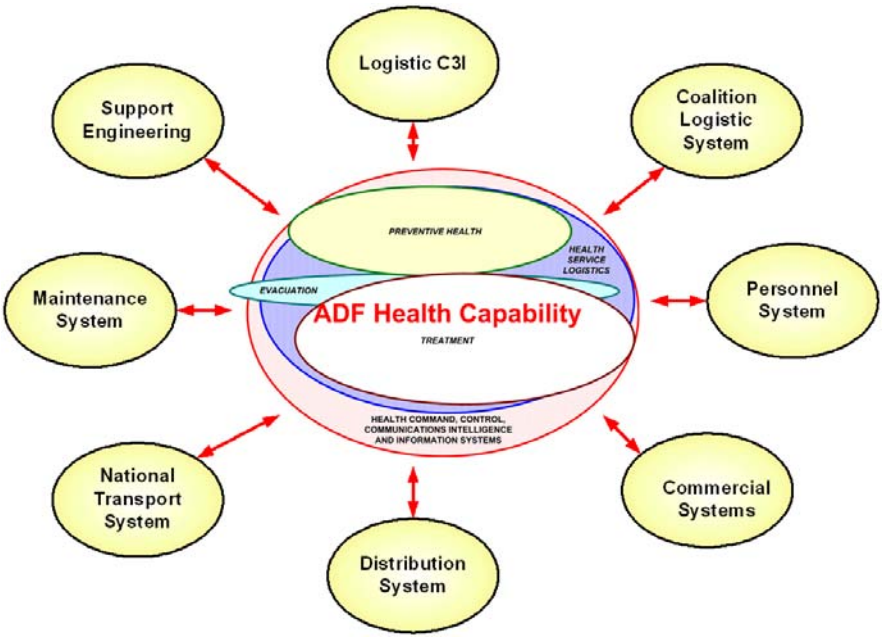

Figure 3: An Environment Diagram[6]

Constraints

The constraints space details any restrictions on the system that can be identified, including how it should satisfy objectives. These restrictions may include statutory occupational health and safety requirements, corporate policy requirements, security issues, or environmental hazards that may influence the final design of the system. Constraints are normally, but not exclusively, expressed as non-functional requirements.

Solutions

The solutions space is made up of components from the mnemonic PISHI[7], which stands for people, infrastructure, software, hardware, and information. This classification represents real thing in the real world, and each of these things is linked together by processes so that an output is realised to meet an objective. By dealing with real things in the real world complexity is immediately reduced. Each of these things does tasks that meet or achieve the objectives. Clearly this space captures all the necessary components of a knowledge management system – people, process and technology.

The ability to write solid atomic requirements for solutions depends upon which component is being addressed. For example writing requirements for hardware components is conceptually easy because conventional systems engineering methodologies can be applied. Similarly infrastructure requirements are relatively easy to define. However, the people, information and software components are much more difficult. Again this where soft systems methodologies should be applied to elicit as many requirements up front as possible. In

[6] This diagram was developed by the author in a workshop that examined the operating environment for a future ADF health capability.

[7] The construct of PISHI has been derived from an incomplete systems engineering PhD thesis by Patrick Byrne, which I have refined and adapted to fit into the Knowledge System Design Methodology.

practice the solution space will be visited many times as requirements in other spaces become apparent.

Tasks

The task space identifies all those functional options of achieving each objective or output. In other words, this space is populated with the answer to the question '*what does the system have to do to achieve the objectives or outputs?'* There may be more than one functional option to achieve any particular objective.' Tasks also help to define the processes that are likely to be internal to the system.

A Hypothetical Example

This section uses the methodology outlined above to elicit the high-level requirements for a knowledge management system. In so doing it gives a snapshot of how the KSDM might be used. The example uses real world organisations, but is hypothetical in that a knowledge management system has not yet been built. Furthermore whilst the methodology is discussed in a linear way in practice the process is iterative, and produces many thousands of requirements.

Situation

The Defence Health Capability Directorate (DHCD) desires a knowledge management system so that future equipment acquisitions, doctrine developments and organisational change, are better informed. DHCD aims to '*achieve a world class military health service for the ADF by fully harnessing the intellectual capital of the Defence Health Service and its customers*[8].' In particular it wants to harness the collective ideas, wisdom, and intellectual capacity of all health specialists in the Reserve Force Consultative Groups (RFCGs). The knowledge in these consultative groups is particularly important because these groups house a number of pre-eminent medical specialists, with significant day to day clinical experience.

System Statement

The process begins with management defining in broad terms a systems requirement. The statement must have a broad statement of intent, an outline of some of the expected characteristics, and a statement of achievement. It is desirable that it has a benchmark statement. The system statement requirements would read something like those shown in Table 1.

[8] The vision statement and the business objective were accessed from the Defence Health Service Branch intranet webpage. (http://defweb2.cbr.defence.gov.au/dpehs)

Serial	*Defence Health Capability Directorate Knowledge Management System - System Statement Requirements*	*Type*
1	The system shall link the Defence Health Capability Directorate and the Reserve Force Consultative Groups.	Intent
2	The system shall allow users to capture data so that future equipment acquisitions are better informed.	Achievement
3	The system shall allow users to capture data so that future doctrine developments are better informed.	Achievement
4	The system shall allow users to capture data so that future organisational changes are better informed.	Achievement
5	The system shall allow users to exchange data by providing a common information environment.	Characteristic
6	If existing software cannot be used commercial off the shelf software should be used in preference to purpose built products.	Characteristic
7	The system should have many of the characteristics of the RAAF Aircraft Research and Development Unit[9].	Benchmark

Table 1: System Statement Requirements

Objectives

The next step is to map the high-level objectives. This step will require involvement from throughout the organisation if it is to be successful. Rich pictures and mind-maps are useful tools to capture multiple perspectives during this analysis. Typically the analysis will produce requirements like those shown in Table 2.

Serial	*Defence Health Capability Directorate Knowledge Management System - Objective Requirements*	*Type*
1	The system shall provide central repositories of information for all organisations involved in the business process.	Objective
2	The system shall facilitate the use of digitised explicit knowledge.	Objective
3	The system should facilitate the use of non-digital explicit knowledge.	Objective
4	The system should provide on-line collaboration tools.	Objective
5	The system should provide agents to search academic databases like *Medline.*	Objective

Table 2: Objective Requirements

[9] See (Crompton & Murchland, 2002)

Environment

The next step is to consider the environment, including the organisation culture that the system has to operate in. In this example the analysis might produce a diagram like the one shown at Figure 4, and requirements like those shown in Table 3.

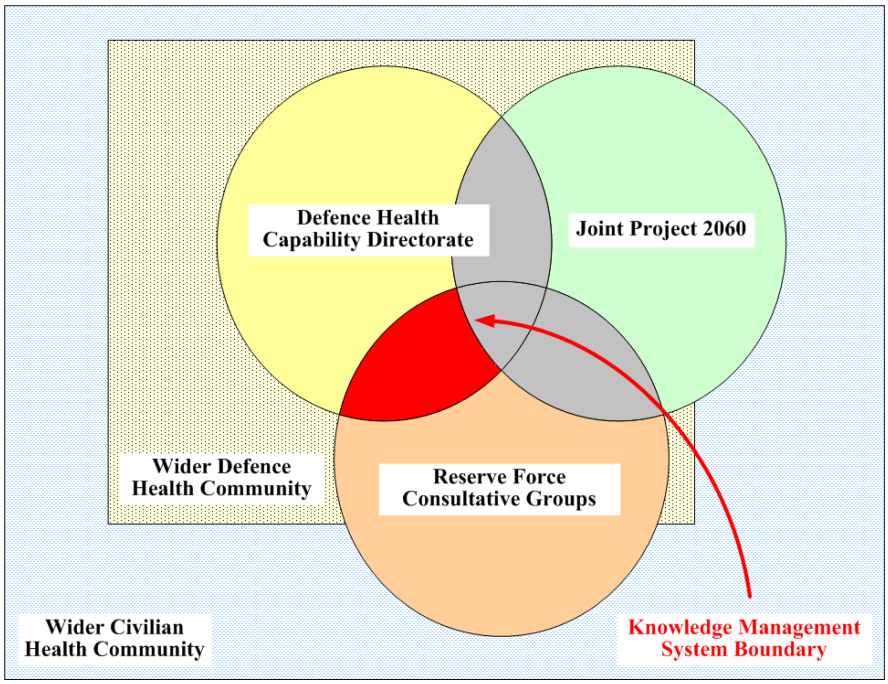

Figure 4: Environmental Boundary and Systems Interfaces

Serial	*Defence Health Capability Directorate Knowledge Management System – Environment Requirements*	*Type*
1	The system shall access information sources internal to the Defence Information Environment.	Environment
2	The system shall access information sources external to the Defence Information Environment.	Environment
3	The system should interface with Joint Project 2060.	Environment
4	The system should interface with national tele-health initiatives.	Environment
5	The system should interface with allied armies knowledge networks.	Environment

Table 3: Environment Requirements

Constraints

The next step is to consider the constraints that arise from the environment, objective and system statement analysis. Typically it will produce requirements like those shown in Table 4.

Serial	*Defence Health Capability Directorate Knowledge Management System – Constraint Requirements*	*Type*
1	The system shall prevent unauthorised users accessing the network.	People Constraint
2	The system shall operate on the Defence Restricted Network.	Infrastructure Constraint
3	The system should separate day to day business from the knowledge system activities.	Software Constraint
4	The system shall conform to Defence military standards for hardware.	Hardware Constraint
5	The system shall prevent information that should not leave the defence network from permeating the system boundary.	Information Constraint

Table 4: Constraint Requirements

Solutions

In practice the solutions space will be visited many times as requirements in other spaces become apparent. This will inevitably produce many thousands of requirements; hence diagrams should be used liberally to provide clarity. In this example a diagram like the one shown at Table 5 might be used. Note that all the elements of the mnemonic PISHI are represented. This analysis will typically produce requirements like those shown in Table 5.

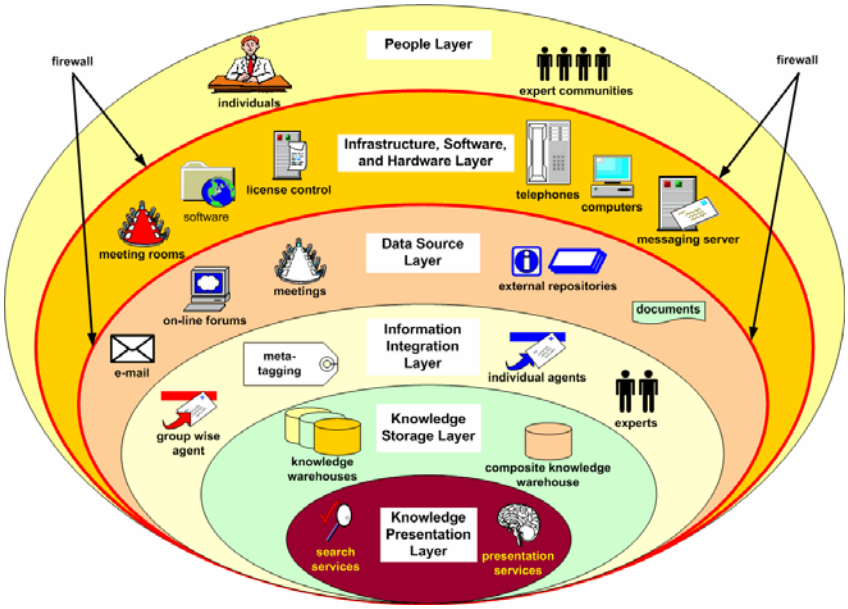

Figure 5: High-level Solution Diagram

Serial	*Defence Health Capability Directorate Knowledge Management System - Solution Requirements*	*Type*
1	The system shall provide a human moderator to ensure that the aggregation of unclassified data does not become classified information.	People Solution
2	The system shall provide meeting rooms equipped with teleconference facilities in each state.	Infrastructure Solution
3	The system shall provide a knowledge warehouse using the *Autonomy*[10] software suite.	Software Solution
4	The system shall use networked personal computers.	Hardware Solution
5	The system shall point to non-digital explicit information.	Information Solution

Table 5: Solution Requirements

Tasks

The final step is to consider all the tasks that arise from the system statement, objective, environment, constraint and solution analysis. In practice this step is done concurrently with the others, and is constantly revisited. Typically it will produce requirements like those shown in Table 6.

Serial	*Defence Health Capability Directorate Knowledge Management System - Task Requirements*	*Type*
1	The system shall filter data to confirm it is relevant.	Task
2	The system shall present information in a form that can be used by all users.	Task
3	The system shall retrieve data from the knowledge warehouse.	Task
4	The system shall index digital data.	Task
5	The system shall correlate data to establish relationships.	Task

Table 6: Task Requirements

[10] *Autonomy* operates from a series of portals or windows that contain an application to do a variety of functions, including automated retrieval of information, real-time analysis of the ideas involved in the content of any opened application, alert capabilities, and automatic indexing. It has been extensively tested by the Defence Science and Technology Organisation. ((Alvino, 2003))

Conclusion

Knowledge management systems are inherently soft open systems with permeable boundaries. Their open soft nature makes them susceptible to failure because of unrealistic expectations resulting from poorly expressed, incomplete, or changing requirements. The foregoing discussion has presented a model that allows the development of requirements for a knowledge management system to be developed in a holistic and disciplined way, thereby reducing the need to develop or modify requirements on the fly.

The model builds upon soft system methodologies and 'hard' system engineering approaches, and is based on the assumption that objectives or outcomes are achieved in an environment that produces or has constraints. These constraints either restrict the availability of solutions or restrict how solutions can be employed. The solutions are real things in the real world, and include people, infrastructure, software, hardware and information. All of these solutions perform tasks to achieve or meet defined objectives.

The model examines the system from a conceptual system statement through objectives, into the real cost drivers of any system, the solution space. The ability to write solid atomic requirements for solutions depends upon which component is being addressed. For example writing requirements for hardware components is conceptually easy because conventional systems engineering methodologies can be applied. Similarly infrastructure requirements are relatively easy to define. However, the people, information and software components are much more difficult. Again this where soft systems methodologies should be applied to elicit as many requirements up front as possible. That said by following the model in a disciplined way one is more likely to consider all of the components of a knowledge system and write precise atomic requirements. Research shows that the likelihood of success will increase by at least 21% simply by examining and preparing requirements more thoroughly (Standish Group, 2000). Clear requirements will allow management to understand and cost the initiative, thereby reducing unrealistic expectations and increasing the chances of success.

References

Alvino, L. (2003). *An overview of autonomy*. Adelaide: Defence Science and Technology Organisation.

Blauberg, I., Sadovsky, V., & Yudin, E. (1973). Some problems of general systems development. In W. Grey & N. Rizzo (Eds.), *Unity through diversity: A festschrift for ludwig von bertalanffy*. New York: Gordon and Breach.

Braithwaite, J., Hindle, D., Ledema, R., & Westbrook, J. (2002). Introducing soft systems methodology plus (ssm+): Why we need it and what it can contribute. *Australian Health Review, 25*(2), 191-199.

Checkland, P. (1993a). An application of soft systems methodology. In J. Rosenhead (Ed.), *Rational analysis for a problematic world: Problem structuring methods for complexity, uncertainty and conflict* (pp. 101-120). Chichester: John Wiley and Sons.

Checkland, P. (1993b). Soft systems methodology. In J. Rosenhead (Ed.), *Rational analysis for a problematic world: Problem structuring methods for complexity, uncertainty and conflict* (pp. 71-120). Chichester: John Wiley and Sons.

Cho, G., Jerrell, H., & Landay, W. (2000). *Know the way: How knowledge management can improve dod acquisition*. Fort Belvoir: Defense Systems Management College.

Crompton, R., & Murchland, P. (2002). Best practice in community building and information discovery. Retrieved 5 March 2003, 2003

Damain, D., Jonker, C., Treur, J., & Wijngaards, N. (2003). *A formal knowledge level process model of requirements engineering.* University of Calgary, Calgary.

Davenport, T. H., & Prusak, L. (1998). *Working knowledge: How organisations manage what they know*. Boston: Harvard Business School Press.

Delong, D. W., & Fahey, L. (2000). Diagnosing cultural barriers to knowledge management. *Academy of Management Executive, 14*(4), 113-127.

Hitchins, D. (2003). Putting systems to work. 2003

Kannegieter, T. (2001). *Knowledge management: A framework for succeeding in the knowledge era*. Sydney: Standards Australia International Limited.

Rosenhead, J. (Ed.). (1993). *Rational analysis for a problematic world: Problem structuring methods for complexity, uncertainty and conflict.* Chichester: John Wiley and Sons.

Shehata, M., & Bowen, S. (1999). *Soft systems methodology.* University of Calgary, Calgary.

Soo, C., Devinney, T., Midgley, D., Deering, A., & Kearney, A. (2001). Knowledge management: Philosophy, process, pitfalls, and performance. 2003

Sparks, J. N. (1997). *Soft operational research techniques for the acquisition and management of logistics*. Canberra: Department of Defence.

Staker, J. (2000). *Knowledge based soft systems engineering for military systems of systems.* Paper presented at the SETE 2000.

Staker, J. (2003). *Towards a knowledge based soft systems engineering method for systems of systems.* Paper presented at the Joint Systems Branch Conference, Salisbury.

Standish Group. (2000). *What are your requirements?*

Stevens, R. (2002). *Get it right the first time: Writing better requirements*. Oxford: Telelogic DOORS UK.

Stevens, R., Jackson, K., Brook, P., & Arnold, S. (1998). *Systems engineering: Coping with complexity*. London: Prentice Hall.

Tiwana, A. (2002). *The knowledge management toolkit: Orchestrating it, strategy, and knowledge platforms*. Upper Saddle River: Prentice Hall.

Vencel, L. (1999). *Why is defining requirements so hard?* Paper presented at the SETE 99.

