Knowledge Re-use and Dissemination for Resource Elicitation in Software Engineering

Max H. Garzon, Chris Simmons and Jason Knisley

Computer Science Department, The University of Memphis, Memphis, TN 38152, U.S.A.

Abstract. Software Engineering requires intense knowledge-driven practices and procedures that require team work in collaborative environments. Despite recent improvement in collaboration technology and the fact that over 80% of software project delays and overruns can be traced back to failures in proper requirement identification, requirement gathering remains an elusive art. This study reports on a survey of industry practices run to capture both experiential and procedural knowledge into an ontology that identifies and prioritizes it. The ontology is used as the foundation of a knowledge building tool, ***R.E.M.,*** that can assist junior project managers (PMs) through the software process, including mitigating communication skills, negotiation skills, and time management facilities while eliciting requirements and managing small to medium software projects. Systematic use of R.E.M. will afford organizations with a principled and systematic repository of "institutional memories" of development experience, less dependent on personalities and more reflective of the culture of the organization.

1 Introduction

Effective Software Engineering (SE) requires intense knowledge- and experience-driven practices and procedures that emerge in human brains as a result of interaction and team thinking in collaborative environments. Despite enormous improvement in collaboration technology in the last few decades, there still remains a large gap in our ability to systematically store and organize the experience and knowledge garnered by team participants in the practice of SE, from coders, developers and software managers, to entire organizations, throughout their lifetimes. The result is that, for example, after a decades long career as a software project manager PM with an organization X, during which PM has accumulated a great deal of experience and knowledge about the process, the clients and the culture prevalent in the organization, PM leaves or retires and a young PMJr takes his place. PMJr usually begins yet again the entire cycle of creating anew that experience and knowledge, not to mention the less tangible but more important issues of understanding, and re-using the data and knowledge contained in closed and ongoing projects. While the process may have a positive side in the sense of bringing in fresh ideas and new methods to bear on the job, it is also clear that over 50% of knowledge and experience (part of what might be termed "institutional memory"), will be lost in the turnover, not to mention the cost due to delays or parallel training that might be required to make the loss bearable, or the analo-

H. Garzon M., Simmons C. and Knisley J..
Knowledge Re-use and Dissemination for Resource Elicitation in Software Engineering.
DOI: 10.5220/0004638500140023
In *Proceedings of the 4th International Workshop on Software Knowledge* (SKY-2013), pages 14-23
ISBN: 978-989-8565-76-1

gous problem for developers and other roles in the software process. Turnover of PMs usually results in total loss of experiential knowledge to the organization. It is thus desirable to develop methods to build "institutional memories" of this knowledge and individual experience to make it less dependent on personalities and more reflective of organization's culture. How could the problem be addressed in a more principled and systematic way to provide more effective solutions, both cost and timewise?

There is clear overlap between ontology engineering and requirement engineering. Dobson [18] explored the synergy between the two fields and developed a dependability ontology to address requirement engineering. The field of ontology engineering provides techniques to effectively develop concepts and relationships for a given domain [6]. An ontology provides an explicit specification of concepts related to a domain and the relationships amongst them to create an organized knowledge base. Furthermore, ontologies provide a mechanism for shared vocabularies, allow an improvement of information retrieval, and help data integration [14]. There are several methodologies for ontology development. For example, Methonology [7] is an ontology building methodology that focuses on the reuse or reengineering ontologies. The KACTUS project involves a methodology that builds upon itself as knowledge grows and each application is implemented [8]. The On-To-Knowledge methodology involves gathering project objectives through four steps (kickoff, refinement, evaluation, and ontology maintenance) that is used within a knowledge management tool [8]. Cheah [14] described the need of multi-site project management (MSPM), where a need for an ontology that best suites this genre of knowledge as software development is becoming a multi-site initiative. OWL was used to conceptualize the knowledge specifications into logical data models. Jarrar et al. [16] proposed using conceptual data modeling techniques for building ontologies. This provided an ontology-engineering framework that enables reusing conceptual models for modeling and representing ontologies. However, there has been a perceived lack of research focusing on the art of requirement elicitation and tools in the large that will help facilitate this process to aid in requirement development in an experiential and situated context.

In this article, we present our findings from a study conducted to address this problem. The problem is approached from the point of view of knowledge engineering [19], a discipline that involves integrating knowledge into computer systems in order to solve complex problems, normally requiring a high level of human expertise. While the solution offered can be extended to other phases of the software process, such as design and testing, we focus here primarily on illustrating the methodology and results for one of the most difficult and important phases of the process, namely *requirement elicitation* and *task scheduling*. Poor requirement gathering is associated with 85% of errors in software projects [1]. Requirement engineering (RE) has remained one of the most unpredictable phases within the software development lifecycle. RE involves requirement elicitation, analysis (including negotiation), specification, and validation [11]. RE research has continued to overlook the *elicitation* factor, as it provides a key aspect in developing customer and user requirements [10]. Research has mainly focused on developing various methods to gather requirements, while research has been stagnant in developing methods and tools for requirement elicitation that is not oblivious to PM turnover, for example. A notable advance is the work by Newell and Huang [3] describing the importance of constructing, articulating, and redefining shared information with organizational members in a social interaction environment. It is thus essential to provide a knowledge management tool that

captures and disseminates pertinent information to all involved parties in a working group context, as part of the usual interactions required among the team members.

In Section 2, we describe and show the findings of a field study carried out to articulate a proper ontology for such a tool. In Section 3, we describe the structure and development of our proposed tool ***r.E.M***., or "Requirement Elicitation Manager." We also present a summary of current work to extend it to a full-fledged ***R.E.M.***, a Resource Elicitation Manager that includes other stages in the software cycle, and present the results of the evaluation of an alpha prototype**.** Finally, Section 4 concludes with a discussion of the philosophy or R.E.M., as well as some future research.

2 An Ontology for Requirement Elicitation

In this Section we describe a field study conducted to inform the design of r.E.M and show the resulting ontology developed as the architecture for r.E.M. and R.E.M. r.E.M enables PMs with the ability to capture, embody and capitalize on experiential knowledge to assist them in the development of software requirements.

In this work, we make a careful distinction between 'requirement gathering" (the typical act of collecting requirements) and "Requirement elicitation", the nontrivial process that currently includes interviews, questionnaires, user observations, workshops, brain storming, use cases, role playing and prototyping, in order to arrive at a set of requirements of high-quality. Requirement elicitation is recognized as one of the most critical activities of software development. Poor requirement elicitation increases the percentage of a software project failure [9]. Dalkir [4] proposed techniques to elicit tacit knowledge within an organization using cognitive maps, decision trees, and knowledge taxonomies for a systematic approach. Carrizo et al. [10] proposed a framework that enables an elicitation process depending on the type of project at hand. Enabling a seamless elicitation process within a knowledge repository can provide great insight towards developing a complete set of requirements. Hoffman and Lehman [15] developed a set of requirement engineering activities from surveying the developed requirements engineering (RE) cycle of a selected number of organizations to determine the success in producing requirements. Their RE activity success was measured using three factors: *knowledge, resources, and process*. We used their technique to determine key elements in producing quality requirements. What should then be the structure of an ontology for effectively capturing and managing requirement elicitation so that a junior PM can inherit knowledge, work flow and information of the entire organization while minimizing turnover costs?

2.1 Methodology and Tools

The great majority of time and effort in the software process is spent in structuring communication with the client and among the team members, followed by structuring the process of definition, assignment, logistics and delivery of tasks to/from the software team members. Requirement elicitation is really an art, and the best one can really do is to ask experienced software managers what worked for them and what they have come to believe are the key factors and steps in the process flow to make it

happen. After a review of the literature as described above, in order to develop an ontology consistent with actual practices that would make r.E.M**.** truly useful, we scheduled and conducted interviews with a number of seasoned PMs. We developed a qualitative questionnaire to capture senior project managers' experiential and situational knowledge in requirement elicitation for software development within any small to medium sized organizations. *Embodying that knowledge in the design and work flow of r.E.M. will enable it to capture and capitalize on experiential knowledge described above*.

We conducted interviews with five (5) seasoned senior project managers (sPM) from three Fortune 500 businesses, 1 small-to-medium businesses, and 1 consultant company. The interviews were structured as a complete set of 13 open-ended questions, focusing on outsourcing, soft and behavioral skills, and duties of project managers that were deemed by them as being an important need for junior project managers to know and follow when eliciting requirements. These interview sessions were recorded by the authors with the permission of each participant and IRB approval.

Below are the questions that were asked each participant. Each question is labeled by the order which it was asked, and followed by a synopsis of the answers received that emphasizes the consensus, while leaving out a few divergent points.

1. *What information does a good project manager provide a customer when defining requirements?*

There was a consensus that any PM should provide a customer with templates or tools necessary to capture the "as is" and "to be" scenarios. Templates used consisted of Business Product Document, Visio document, Gap analysis document, and/or presentations that help drive the development of requirements.

2. *How can a PM effectively capture customer goals within requirements?*

Effectively capturing customer goals requires *meeting of the minds* through the creation of a requirements document. This document references business goals that are in line with the product being developed, and which enable a mapping of requirements to goals. Some of the techniques currently involve business process map, system questionnaires, workshops, and application development sessions. All 100% of interviewees believe some form of communication and collaboration session is required to effectively capture customer goals within requirements.

3. *How do we know when a requirement has been properly defined/captured? Or, that the requirement is at the appropriate level of detail?*

Participants suggested that knowing when a requirement is at the appropriate level of detail boils down to knowing when the stakeholders involved are willing to sign off on the requirement. One interviewee suggested bringing in a representative from the technical team to help with analyzing and deciding when a requirement is properly defined. Mainly, a *governance process* is needed to facilitate the process of properly defining a requirement, where negotiation skills can play heavily with new project managers, thus making requirement elicitation an art.

4. *How do you make customers aware that once approved, requirements should not be changed?*

Our interviewees discussed the importance of having the customer aware of change as an important aspect to minimizing change. Providing the customer with upfront in-

formation regarding how a change impacts various aspects of the project including cost and time, aids the customer and PM with the ability to come to a consensus on requirements. Some of the techniques to accomplish this objective involve workshops, requirements document, education, and enabling sign off on everything that will help reduce requirement change, but may not eliminate it at all.

5. *What are the most common fatal mistakes/errors that a manager can make when gathering requirements?*

There was a consensus amongst interviewees that talking to the wrong person is a common fatal mistake when attempting to gather requirements. PMs need to ensure the customer understands that equal and joint ownership is vital to the success of the project, where the customer makes available the important people to help effectively elicit requirements. Equal ownership helps in ensuring the time schedule is appropriate, system users are available, and that management commitment and signoff are there. Results suggest that more universities allow companies to come in to explain their needs. Others suggest the following training opportunities to assist in outsourced development: database knowledge, shell scripting, swing, configuration of management, process improvement, unit testing, documentation, project management, time management, cultural differences, outsource management, information assurance, and tools and technologies that will help improve the software development lifecycle.

6. *How can a PM tell that the set of requirements is complete?*

Results suggest that an integration of teams is required to help determine if the set of requirements is complete. This aids in tracking details and obtaining signoff to determine if requirements are satisfactory with the customer and the next phase of the project. Tools to accomplish cohesiveness include workshops, assumption and decision logs, and previous project requirements to help determine if a set of project requirements is complete.

7. *How effective are similar project requirements used in developing new project requirements (common patterns)?*

Interview participants highlight the ability to save an average 20% of time during the requirement gathering phase. Previous project requirements provide great leads when a project manager elicits customer requirements for new projects. Participants recommended the need for a highly documented repository that embodies the reuse of project requirements, which provides good leverage for knowledge reuse.

8. *How to effectively elicit and manage risks within a project?*

Participants recommend the need for risk management and being able to manage those risks on a weekly basis. Some suggest having a risk management team identify risks that directly impact the critical path to developing the product. Mitigating risks involves continuously tracking and communicating risks to appropriate personnel.

9. *How to effectively elicit from a customer an assessment of the impact of the business rules on the project requirements?*

The interviewees highlighted the ability of being able to document the ‘as is’ process. Decomposing the ‘as is’ process will assist in facilitating tasks that are important to the business and provide an operational definition list to clarify business rules. This process is completed in a couple of notable ways including workshops and functional specification document to aid in identifying gaps.

10. How to effectively prioritize requirements?
Interview responses varied in how to effectively prioritize requirements. Suggestions range from forcing the business into a joint effort, to documenting the end goal, and developing use case scenarios. These various suggestions enable the business to seamlessly prioritize requirements based on goals, cost, and time.

11. What other factors to get from the customer in gathering requirements?
Interviewees ranged broadly in additional factors, such as providing a prototype, application flexibility, potential environment, and identifying potential reports. Also, identifying training needs, volume of transactions, system factors, people factors, hardware/equipment used to run applications, recursive processes, business conditions, customer responsibilities, and providing a solid project charter would help ensure that requirements are at the appropriate level of detail.

12. How do available resources impact requirement development and formulation?
Interview participants recommended identifying the budgetary constraints early in order to provide the customer knowledge of how the resource constraints directly impact requirement development within a project. This process requires obtaining executive level buy-in to provide an organizational understanding of available resources needed to effectively gather requirements.

13. Is the ontology/organization in these questions above a comprehensive account for all concepts and relationships for successful requirement gathering and advice?
Participants suggest the ontology captures the major functions influencing project performance when gathering requirements. Noted was the ability to facilitate the change review process in a tool that is able to capture "scope creep" when decisions await approval.

2.2 The Ontology

The responses from the questionnaire were analyzed from the recordings to gain thorough understanding to questions that may have been missed during recording. We then went through iterative design to produce an ontology for the design of r.E.M.. As shown in Fig. 1, this ontology identifies and prioritizes knowledge to provide steps for successful implementation into a usable knowledge management tool for effective requirement gathering. It was used as data template in the design of r.E.M and R.E.M.

2.2 The Process Flow in r.E.M.

In addition to the ontology, an appropriate flow of the software process can be identified from the interview responses. Our analyses led us to the conclusion that building a software tool is feasible that would embody this ontology, organize relevant data, suggest requirement elicitation strategies, remind them of resource constraints and strategies, facilitate the negotiation process, and reduce the logistic cost of communication among software team members. This tool has the potential to provide a practical and usable solution to the problem (as 100% of questionnaire respondents emphasized.) Negotiation skills, time management, cultural differences, outsource manage-

ment, and information assurance trainings were identified as being the most notable aspects to capture in the process flow. Above all, the ability to enhance effective communication was identified as ***vital***. The tool described below addressed most of these constraints and is likely to allow even PMjrs to hone in on the art of requirement elicitation and project management in communication through r.E.M.

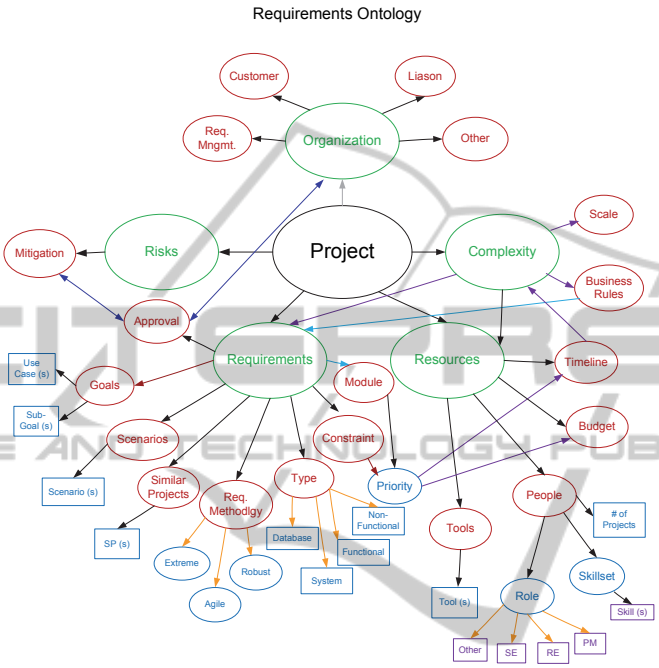

Fig. 1. An ontology for requirements elicitation in r.E.M. for small-to-medium sized software projects. The **ovals** refer to major concepts necessary to successfully articulate requirements for a particular project. The **boxes** refer to the terminal entities that provide specifics of the super-class concepts. The **arrows** refer to the relationships between concepts relevant to requirement gathering. The ontology is the backbone of the structure of a software repository tool R.E.M.

The ***r.E.M.*** (requirement Elicitation Manager) provides three major functionalities for managing projects/requirements, their proper storage in a centralized file system and enabling communication among the software team. r.E.M. assumes participants communicate in *pull-mode*, including clients, PMs and members of the software team. This design eliminates the logistic problem of massive use of email to team members and ad-hoc file personal archives. Although the design of r.E.M. was done using a Structured Development Process (aka, Generic Lifecycle Model) consisting of five distinct phases: requirements definition, software design, software implementation, software testing and evaluation, and although it only concerns the first phase (the other phases will be used in future extensions to R.E.M.), it is important to note that users of r.E.M. can also use it in iterative approaches and switch a project to an earlier stage (more in Section 4 below.) A formal evaluation of an alpha version of r.E.M. was conducted as part of the evaluation of its extension R.E.M. The results can be seen below in Section 3.2.

3 R.E.M.: Resource Elicitation Manager

In this Section, we present a summary of a second module that extends r.E.M. to a full-fledged tool R.E.M. by adding facilities to handle resource allocation (the Scheduling Elicitation Manager, S.E.M.), and the first part of the process flow to move a project along the stages of the software development process. Although these extensions seem to impose a specific methodology (e.g., Structured Development), the end product R.E.M. is really methodology independent, as will be discussed in Section 4.

3.1 Scheduling Elicitation Manager (S.E.M.)

Both r.E.M. and S.E.M. (Schedule Elicitation Manager) are distinct, but coherent, modular knowledge-based application components of R.E.M. Both modules are intended to serve as knowledge engineering tools aimed at facilitating end-to-end projects for PMs, helping in allocating resources, and providing an at-a-glance view of project status. S.E.M. focuses solely on the partitioning, scheduling, and monitoring of development/programming tasks associated with requirements derived during the requirements elicitation phase. Specifically, S.E.M.'s user interface (UI) provides PMs and developers with the following capabilities:

- Allow PMs to break each project down into requirements and each requirement down into tasks (r.E.M.);
- capture task complexity and dependencies and schedule tasks accordingly for execution using the Critical Path Method, while allowing PMs to make changes;
- allows PMs to plan appropriately for time logistics, including assigning tasks to developers and monitoring their progress, and follow up through completion;
- provide PMs with alerts and warnings when a task is about to exceed its deadline, or upgrade it to a warning if the task is on the critical path.

3.2 Evaluation of r.E.M and S.E.M. Prototypes

A formal evaluation of an alpha version of r.E.M. was conducted using a survey taken by 42 students enrolled in an undergraduate Software Engineering course at The University of Memphis, with IRB approval, ideal users given their minimal experience in software management and major need in terms of guidance. The 10 questions were each posed as statement of achievement of the intended requirements of the tool (e.g., "The tool r.E.M. allows for easy handling of projects, requirements and users." and "The strategy used to display the projects Dashboard is useful to plan and move the projects along.") Responses were captured on a categorical Likert scale (ranging from 0="Strongly Agree" to 4="Strongly Disagree"). The evaluation gave was deemed satisfactory and very encouraging, since r.E.M. received an average score of **0.92** (slightly lower than "Strongly Agree".) Likewise, a separate evaluation of S.E.M. received a score of **1.34** (slightly above "Agree"). Numerous comments in the open-ended question were very encouraging that the combined package R.E.M. might be very useful in their professional lives and that they would actually be seeking and willing to adopt it as software managers and developers.

Fig. 2. Typical snapshot of the PM's user interface for *r.E.M* and *S.E.M.*, enabling them to coordinate and move along requirements and tasks from definition to completion effectively, while minimizing the team's efforts spent on logistics and communication overhead, according to an evaluation with PMJrs..

4 Discussion and Future Work

The basic premise of this work is that the type of knowledge transacted in successful software Engineering (SE) projects is not reducible to a number of abstract and universal principles, but rather requires anchoring them in an experiential practice. An ontology has been developed from a field study that involved qualitative interviews from senior project managers. The ontology was embodied into two modules (r.E.M. for resource elicitation and S.E.M. for scheduling elicitation) of an integrated tool R.E.M. that captures factual and procedural knowledge engineering required to support project managers in developing small to medium size projects. This is a first step in producing a comprehensive tool R.E.M. that will benefit all project managers with mitigating lack of experience in communication skills, negotiation skills, outsource management, and time management, as well as other logistics while eliciting and developing software solutions. This tool is expected to evolve as it accumulates institutional memories in the form of project data, developers and PMs performance records and practices in the organization. They can be used as valuable data in projected built-in enhancements concerning advising systems for PMs in estimating completion time for task assignments, best actors for a given task and process flow that will make full use of that data knowledge to make the software process smoother and effective.

Although R.E.M. was designed using a specific methodology (Structured Development), the end product is really methodology independent. R.E.M can be used by any manager using any software development methodologies (including Agile) which involves breaking requirements down into tasks, assigning actors and timelines, switching tasks back or forth to prior stages to completion. R.E.M. also offers a unified platform for file storage that enables seamless and effective communication among team members in pull-mode. Further, R.E.M.'s design is flexible and allows addition of other modules, e.g., for design (D.E.M.) and Testing (T.E.M.) that will be

described elsewhere. The real test of the advantages of the tool remains the eventual wide acceptance by PM/developers in the practice of software engineering.

References

1. J. Liebowitz, J. and I. Megbolugbe (2003). “A set of frame- works to aid the project manager in conceptualizing and implementing knowledge management initiatives,” Int. J. of Project Management. 21:189-98.
2. J. Liebowitz (2005). ‘Conceptualizing and Implementing Knowledge Management’. Editors: Love P., Fong P.S.W. and Irani Z. Management of Knowledge in Project Environments. Elsevier. Burlington, UK.
3. J. C. Huang, J. C. & S. Newell (2003) “Knowledge integration process and dynamics within the context or cross-functional projects,” Int.l J.l of Project Management, 21, 167–176.
4. K. Dalkir (2005). “Knowledge Management in Theory and Practice - Knowledge Capture and Codification,” Elsevier. Butterworth-Heinemann, Burlington, MA.
5. Dieter Fensel, Ian Horrocks, Frank van Harmelen, Deborah L. McGuinness, and Peter F. Pate;-Schneider (2001). “OIL: An Ontology Infrastructure for the Semantic Web,” IEEE Intelligent Systems (Special Issue on Semantic Web), 16(2).
6. Q.N.N. Tran and Graham Low (2008). "MOBMAS: A methodology for ontology-based multi-agent systems development." Information & Software Technology 50, no. 7/8: 697-722.
7. M. Fernández-López, A. Gómez-Pérez, A. Pazos-Sierra, J. Pazos-Sierra (1999). “Building a chemical ontology using METHONTOLOGY and the ontology design environment,” IEEE Intelligent Systems & their applications 4 (1) 37–46.
8. O. Corcho, M. Fernández-López, and A. Gómez-Pérez (2003). "Methodologies, tools and languages for building ontologies. Where is their meeting point?." Data & Knowledge Engineering 46:1, 41.
9. Hickey, A. M., and Davis, A. M. “A unified model of requirements elicitation,” Journal of Management Information Systems, 20, 4 (2004), 65-84.
10. D. Carrizo, O. Dieste, N. Juristo (2008). “Study of Elicitation Techniques Adequacy” 11th Workshop of Requirements Engineering, Barcelona, Spain, 104-144.
11. G. Koyonya, G., and I. Sommerville (1998). “Requirements Engineering: Processes and Techniques,” John Wiley and Sons.
12. H. Andrade, H. and J. Saltz. “Towards a knowledge base management system (KBMS): An ontology-aware database management system (DBMS)”Proceedings of the 14th Brazilian Symposium on Databases, Florianopolis, Brazil.
13. D. R. R. Young, "Effective Requirements Practices," Boston: Addison- Wesley, 2001.
14. C. Cheah (2007). “Ontological Methodologies – From Open Standards Software Development to Open Standards Organizational Project Governance” IJCSNS International Journal of Computer Science and Network Security, 7:3.
15. H. Hofmann and F. Lehner (2001). “Requirements Engineering as a Success Factor in Software Projects” IEEE Software, July.
16. M. Jarrar, J. Demy J., R. Meersman (2003). “On Using Conceptual Data Modeling for Ontology Engineering,” In: Aberer K., March S., and Spaccapietra S., (eds.): Journal on Data Semantics, LNCS Vol. 2800, Springer. ISBN: 3-540-20407-5., 185–207.
17. N. Guarino (1997). “Understanding, building and using ontologies,” International Journal of Human Computer Studies 46, 293-310.
18. G. Dobson and P. Sawyer (2006). “Revisiting ontology-based requirements engineering in the age of the semantic web,” in Proc. of the Int. Seminar on Dependable RequirementsEngineering of Computerised Systems at NPPs. Halden, Norway.
19. S. Kendal, M Creen (2007), An Introduction to Knowledge Engineering, Springer.