Lars Mathiassen – Timo Saarinen – Tuure Tuunanen – Matti Rossi

MANAGING REQUIREMENTS ENGINEERING RISKS: AN ANALYSIS AND SYNTHESIS OF THE LITERATURE

HELSINKI SCHOOL OF ECONOMICS
WORKING PAPERS
W-379

Lars Mathiassen* – Timo Saarinen** – Tuure Tuunanen** – Matti Rossi**

MANAGING REQUIREMENTS ENGINEERING RISKS: AN ANALYSIS AND SYNTHESIS OF THE LITERATURE

*Center for Process Innovation,
Georgia State University

**Helsinki School Economics,
Department of Management, Information Systems Science

November
2004

HELSINGIN KAUPPAKORKEAKOULU
HELSINKI SCHOOL OF ECONOMICS
WORKING PAPERS
W-379

HELSINGIN KAUPPAKORKEAKOULU
HELSINKI SCHOOL OF ECONOMICS
PL 1210
FIN-00101 HELSINKI
FINLAND

© Lars Mathiassen, Timo Saarinen, Tuure Tuunanen, Matti Rossi and
Helsinki School of Economics

ISSN 1795-1828
ISBN 951-791-895-X (Electronic working paper)

Helsinki School of Economics -
HeSE print 2004

Managing Requirements Engineering Risks:

An Analysis and Synthesis of the Literature

Lars Mathiassen (lmathiassen@gsu.edu)

Center for Process Innovation, Georgia State University

P. O. Box 4015, Atlanta, GA 30303-4015, USA

Phone: +1-404-651-0933, Fax: +1-404-463-9292

Timo Saarinen (timo.saarinen@hkkk.fi)

Helsinki School Economics, Department of Management, Information Systems Science

P. O. Box 1210, FIN-00101 Helsinki, Finland

Phone: +358-9-431-38272, Fax: Fax: +358-9-431-38700

Tuure Tuunanen (tuure.tuunanen@hkkk.fi)

Helsinki School Economics, Department of Management, Information Systems Science

P. O. Box 1210, FIN-00101 Helsinki, Finland

Phone: +358-40-544-5591, Fax: +358-9-431-38700

http://www.tuunanen.fi

Matti Rossi (matti.rossi@hkkk.fi)

Helsinki School Economics, Department of Management, Information Systems Science

P. O. Box 1210, FIN-00101 Helsinki, Finland

Phone: +358-9-431-38997, Fax: Fax: +358-9-431-38700

Managing Requirements Engineering Risks:

An Analysis and Synthesis of the Literature

Abstract

Requirements engineering is recognized as a key discipline in developing business software. Practitioners are, however, facing a steady stream of new techniques and an increasingly differentiated portfolio of requirements engineering risks. The purpose of this paper is to propose a model that links the available repertoire of techniques to the situations in which practitioners find themselves. To this end, the paper reviews the software development and requirements engineering literature to understand the risks that characterize requirement engineering situations, to classify available techniques to resolve these risks, and to identify key principles by which tactics can be applied to resolve requirements risks. The paper synthesizes the findings from the analysis into a contingency model for managing requirements engineering risks. The model sets the scene for future research and practitioners can use it to navigate the requirements engineering landscape.

Keywords: Business software; requirements engineering; risk management; contingency model.

Managing Requirements Engineering Risks:

An Analysis and Synthesis of the Literature

1. INTRODUCTION

The requirements for the first software applications were often easy to identify since most applications were developed by scientists to support their own needs and purposes. However, as programmers began to develop business software for end-users different from themselves, it soon became important to systematically gather, explicate, and understand user needs. This has resulted in a considerable variety of techniques (Byrd et al. 1992; Davis 1982; Keil et al. 1995; Nuseibeh et al. 2000) to support requirements engineering in business contexts. Some would argue that the constant stream of techniques has developed into a methodology jungle (Jayaratna 1994).

Researchers have responded by developing frameworks that practitioners can use to navigate the requirements engineering landscape. The idea is to help practitioners design approaches that fit the situations they face. Such contingency frameworks offer three elements: an understanding of the situations involved, an understanding of the portfolio of available techniques, and a set of heuristics that link available techniques to types of situations (Iivari 1992; Kickert 1983). Many contingency frameworks are based on risk management ideas: the profile of the situation is analyzed in terms of risks, approaches are seen as risk resolution tactics, and these tactics are linked to situations based on their capacity to resolve certain types of risks (Lyytinen et al. 1998). As a first attempt to

systematically apply requirements engineering techniques, Alter et al. (1978) introduced a contingency framework to help develop software for decision support. McFarlan (1981; 1982) made an effort to help organize development of business software by achieving appropriate integration internally amongst developers and externally between developers and end-users. Davis (1982) focused on the challenges in determining requirements for business software and developed a contingency framework to reduce the uncertainty of the development task.

Many changes have, however, occurred in requirements engineering practices and techniques since the early 1980s. Ubiquitous computing, increased emphasis on inter-organizational applications, and demand for shorter project life-cycles have introduced new techniques and changed the risk profile of requirements engineering. Today, developers often face end-users who are not within organizational reach and development teams are therefore challenged to establish effective interaction with would-be users to inform the design process (Duggan et al. 2004; Frolick et al. 1995; Peffers et al. 2003a). This challenge increases when developers face users who do not know how to describe their needs (Walz et al. 1993).

The literature provides a rich understanding of the risks related to development of software in business contexts (Barki et al. 1993; Lyytinen et al. 1998) and it offers an extensive portfolio of techniques for requirements engineering (Byrd et al. 1992; Keil et al. 1995; Nuseibeh et al. 2000). There is, however, no up-to-date contingency framework that links requirements engineering risks to appropriate tactics (Hickey et al. 2004). As a consequence, it is difficult for practitioners to find guidance in the vast literature on requirements engineering and design approaches tailored to the situations they face.

Questions like how to link the current portfolio of techniques to requirements risks, how to prioritize techniques over the project life-cycle, or how to combine different techniques remain open. Classical contingency frameworks for designing requirements engineering tactics (Alter et al. 1978; Davis 1982; McFarlan 1981; McFarlan 1982) are still useful, but they provide limited support to answer these questions. Moreover, they do not address the risks involved in connecting effectively to end-users that are outside organizational reach. Also, they do not take into account new techniques for requirements engineering that have been developed since the early 1980's.

This paper attempts to fill this gap by providing an up-to-date analysis and synthesis of what we know about requirements engineering risks and techniques. Based on the literature, we analyze why, when, and how requirement engineering techniques should be used in development projects and we synthesize the findings by proposing a contingency model that sets the scene for future research. Practitioners can use the model to navigate the requirements engineering landscape in business contexts.

The paper is structured as follows. Initially, we present our method for reviewing the software development and requirements engineering literature (Webster et al. 2002). We then analyze the literature to understand the risk profile of requirements engineering situations, to classify available requirements engineering techniques, and to identify key principles by which techniques apply to resolve requirements engineering risks. Subsequently, we synthesize insights from this body of knowledge into a contingency model for managing requirements engineering risks. We present the resulting model and show how it can be used to manage requirements risks as a project evolves. We conclude

by discussing implications of the proposed model for requirements engineering research and practice.

2. REVIEW METHODOLOGY

A quality review is complete and focuses on concepts. Two of the key issues in designing rigorous reviews of the literature are therefore how to identify the relevant literature and how to structure the analysis and presentation of the included literature (Webster and Watson, 2002).

2.1. Identifying the Literature

Our methodology for identifying literature seeks to include a clearly defined, complete, and relevant set of research articles. Webster & Watson (2002) emphasize the importance of a rigorous approach to identification of relevant literature recommending to: 1) identify relevant articles in leading journals, 2) go backward by reviewing the citations used by the articles in step 1, and 3) go forward by identifying articles citing the key articles identified in the previous steps. Our six step method implements this recommendation and is summarized in Table 1.

In the first step, we used the Web of Science–service with access to scientific literature from 1990 and onwards to identify software development and software engineering research that would help us understand the profile of risks and the portfolio of techniques in requirements engineering. In this process, we used broad key words to include as many potentially relevant papers as possible. On that basis, Web of Science helped us identify the 500 most relevant articles within software development as well as the 500 most relevant within requirements engineering. The keyword search was done May 15th 2004.

In the second step, we selected those of the papers from step one that were published in leading software engineering and information systems journals. Several papers identify leading journals (Gillenson et al. 1991; Hardgrave et al. 1997; Holsapple et al. 1994; Mylonopoulos et al. 2001; Whitman et al. 1999). We chose two recent lists published in 2003. One focuses on information systems journals (Peffers et al. 2003b), and the other on computer science and software engineering journals (Katerattanakul et al. 2003). By combining these lists, we arrived at leading journals that are relevant for our study, see Appendix 1. We then used the aggregate list to select articles from leading journals.

The sets of papers generated by the two first steps still contained a total of 135 articles. Many of these turned out to be of little or peripheral relevance to our study because of the broad key word search adopted in the first step. We therefore conducted a third step in which we manually filtered each of the two sets of articles based on specific criteria of relevance, see Table 1. The criteria were decided through rounds of discussions between the authors until a consensus was reached.

The first three steps do not include articles written before 1990 because of the Web of Science indexing limitations. As a fourth step, we therefore followed the advice of Webster and Watson (2002) and went backward through the reference lists of all articles included by step three. Within both steams of literature, we compiled an aggregate reference list sorted according to first author and included those articles that had two or more citations in the newer articles in leading journals, i.e. we included those older papers that had most impact in the newer literature.

The two lists of older literature were then in a fifth step filtered manually according to the rules of step three. In the final sixth step, we combined the lists of steps three and five to

generate the total lists of relevant papers to be included in the review. The resulting selection of literature for the review is listed in Appendix 2 with information about which journals the sample is drawn from.

Table 1 Literature selection

Step	Software Development	Requirements Engineering
Step 1: Broad search in Web of Science (May 15th 2004)	- Keywords: 'software development methods', 'software engineering management', 'software process management', 'software life cycle'. - **4,320** of 18,684,867. - Search limited to 500 most relevant**.**	- Keywords: 'requirements and determination' or 'requirements and elicitation'. - **2,633** of 18,860,525. - Search limited to 500 most relevant.
Step 2: Selecting articles in ranked journals[1]	- Result: **97** articles.	- Result: **40** articles.
Step 3: Selecting most relevant articles	- Criteria: 1) should theorize about either software development process or product over the whole life-cycle or 2) Should take a holistic approach to understanding and addressing software development problems and their solutions. - Result: **24** articles.	- Criteria: 1) Should evaluate tactics and techniques for requirements engineering in software and systems development. - Result: **32** articles.
Step 4: Identifying pre-1990 papers	- Result: list containing 62 articles with two or more citations.	- Result: list containing 56 articles with two or more citations.
Step 5: Selecting most relevant articles	- Result: **21** new articles out of the 62 with two or more citations.	- Result: **14** new articles out of the 56 articles with two or more citations.
Step 6: Combining results from step 3 and 5	- Result: **45** articles.	- Result **46**: articles.
Number of reviewed articles: **91** articles (see Appendix 2 for details).		

[1] See Appendix 1 for the list of journals

2.2. Structuring the Review

The objective of our literature review is to analyze why, when, and how requirements engineering techniques should be used in development projects and to synthesize the findings into a model for tailoring available techniques to the situations in which practitioners find themselves. We have consequently chosen contingency thinking (Iivari 1992; Kickert 1983) to help make sense of the selected literature within software development and requirements engineering. This choice is supported by Hickey and Davis's unified model of requirements elicitation (2004) in which they suggest to use situational characteristics as a basis for selection of elicitation techniques. Hickey and Davis argue that their model leads to important new research directions including (2004):

1. Taxonomy of situational characteristics in requirements elicitation.
2. Taxonomy of requirements elicitation techniques.
3. Development of ways to select appropriate techniques.

Compared to Hickey and Davis, our focus is more broadly on requirements engineering. In addition to elicitation of requirements, i.e. learning, uncovering, extracting, surfacing, or discovering needs of customers, users, and other potential stakeholders (Hickey et al. 2004), we include other requirements engineering activities such as selection, analysis, specification and validation of the requirements to be addressed in a specific release of business software. Also, as our goal is to develop a risk management model for navigating the requirements engineering landscape, we have chosen to analyze the literature in three specific themes: ad 1) the risk profile of requirements engineering situations , ad 2) the portfolio of requirements engineering techniques with a risk

resolution focus, and ad 3) the principles by which techniques apply to resolving requirements engineering risks.

Risks denote incidents that endanger a successful development process leading to wrong or inadequate software solutions, rework, implementation difficulty, delay or uncertainty (Boehm 1991; Lyytinen et al. 1996). Requirements risks, and software risks in general, involve the concept of consequence in the form of loss or uncertainty and they require managerial intervention (Barki et al. 1993; Lyytinen et al. 1996). We use the term techniques following Hickey and Davis (2004). Techniques must include a description of what to do, and they can include description of how to do it, including tools and notations to use while doing it.

Mathiassen & Stage (1992) use contingency thinking to link the profile of situations to the portfolio of techniques when developing business software. First, to characterize a given situation they distinguish between complexity, i.e. the amount and structure of the information available to support development, and uncertainty, i.e. the availability and reliability of the information needed for development. Second, they distinguish between techniques that specify requirements and techniques that experiment with requirements. Techniques based on specification are based on abstraction and textual or graphic representation of requirements. Experimental techniques are based on prototyping and iterative process models to gradually evolve software (Boehm 1988) and they involve end-users to help improve the quality of the resulting software (Davis 1982; Keil et al. 1995; Watson et al. 1993). The user base for requirements engineering has, however, widened and so has the gap between developers and users (Grudin 1991; Peffers et al. 2003a; Salaway 1987). This trend has created increased concerns for how to make

relevant information available to a software development team. If the team cannot effectively connect to and interact with would-be users it is difficult to discover relevant information about the software and its practical use. For these reasons, we have refined Mathiassen & Stage's original framework (1992) to reflect the increased importance of effectively connecting to and interacting with would-be-users. We do that by explicitly distinguishing between two different types of uncertainties, those related to the availability and those related to the reliability of the information needed to develop the new software. In this way, we arrive at a general conceptual framework for analyzing requirements engineering risks and tactics as illustrated in Table 2.

Table 2 Framework for literature analysis

Requirements Engineering Risks	Requirements Engineering Tactics
Requirements complexity	Requirements specification
Requirements reliability	Requirements experimentation
Requirements availability	Requirements discovery

3. ANALYSIS OF LITERATURE

In the following, we review the selected literature guided by the conceptual framework in Table 2 and addressing the following questions:

1. How can we understand and analyze requirements engineering risks?

2. How can we understand and identify available requirements engineering techniques?

3. What are the key principles by which techniques can be applied to resolve requirements engineering risks?

3.1. Understanding Risks

The reviewed literature emphasizes requirements complexity as a key risk in software development and requirements engineering. Requirements complexity refers to the amount and structure of the information that is available to design the new software. The more information that is available and the more unstructured it is, the higher the complexity (Mathiassen et al. 1995). Brooks (1987) argue that software is inherently complex. Digital computers are themselves more complex than most other human artifacts, and software has order-of-magnitude more states than computers. Technical issues have therefore been identified by Lyytinen (1988; 1987) as a major reason for development failure. Additional sources of complexity are emphasized by Boehm et al. (1989) who focus on the varying views implied by different stakeholders in the development process, and by Mills (1999) who reminds us that software evolves over time. Glass et al. (1992) summarize that software development 'is the most complex activity the human mind has ever undertaken'. The classical response to complex requirements is specification tactics that uses abstraction to document requirements based on combinations of textual and graphical representations (Mathiassen et al. 1995).

The reviewed literature also emphasizes requirements reliability as a key risk in business software development. Requirements reliability refers to the dynamics of information about the new software. Such dynamics occur as the involved stakeholders change perceptions because they learn during the development process or as the internal or external conditions for using the software change. An additional source of reliability risks is that end-user needs are seldom evident to developers (Houston et al. 2001; Kraut et al. 1995; Nidumolu 1995; Willcocks et al. 1994). Boehm (1988) argues that iterative

approaches can increase requirements reliability by combining learning with systematic documentation. Experimenting has generally been suggested as the tactic that addresses requirements reliability (Boehm 1988; Brooks 1987; Lyytinen 1987; Mathiassen et al. 1995; Ramamoorthy et al. 1996; Zmud 1980). Davis' (1982) contingency framework, which has been slightly modified by Fazlollahi and Tanniru (1991), helps practitioners select appropriate experimental techniques when the uncertainty of the development task is high.

The literature finally emphasizes risks related to requirements availability. The communication gap between developers and end-users has increased as more business applications target users that are external to the organization (Barki et al. 1993; Dennis et al. 1988; Nunamaker et al. 1991). Requirements availability depends on the physical, conceptual, and cultural distance between the developers and the would-be users. There is currently a shift from internal end-users towards customers and end-users that are external to the business. This shift occurs as business software is increasingly produced to markets and used by customers and business partners. The voice of the customers and other external users has, consequently, become an important factor in requirements engineering (Pai 2002; Ravichandran et al. 1999; Ravichandran et al. 2000; Zultner 1993). Questions have been raised on how to identify and reach external users (Hirschheim et al. 1991; Keil et al. 1995) and Salaway argues (1987) that it is more problematic to communicate with external users than with internal ones. Also, end-users in general rarely understand the requirements of business software applications (Walz et al. 1993; Watson et al. 1993). These factors increase the risks related to making information about requirements readily available for a development team.

Communication between stakeholders (Curtis et al. 1992; Curtis et al. 1988; Davidson 2002; Keil et al. 1995) and involvement of different groups of users (Bostrom 1977; Bostrom 1989; Elboushi et al. 1997) are classical examples of discovery tactics that can help development teams access relevant information about requirements.

This analysis of the literature confirms that requirements complexity, reliability, and availability represent important risks in requirements engineering. To further understand how well this conception of requirements risks covers the important sources of risks and how well it provides a balanced view of requirements risk profiles, we examined key sources on software risks and requirement risks. Barki et al. (1993) has reviewed the literature and provide on that basis a comprehensive list of the different sources of risk in development of business software. The only available source that examines in detail the specific risks involved in requirements engineering is Davis (1982). Table 3 maps these two accounts of risk sources to requirements complexity, reliability, and availability. The result suggests that the proposed conception of requirements risks is both comprehensive and well balanced.

Table 3 Mapping requirements engineering risks to measures

Risks	Proposed Measures of Risks	
	Software Development (Barki et al. 1993)	**Requirements Engineering (Davis 1982)**
Requirements Complexity	• Technical complexity • Relative project size • Number of links to existing systems • Number of links to	• A complex system • Lack of well-understood model of the utilizing system. • Lack of structure for activity or decision being supported

	future systems • Number of hardware suppliers • Number of software suppliers • Need for new hardware • Need for new software	
Requirements Reliability	• Task complexity • Extent of changes brought • Lack of development expertise in team • Team's lack of expertise with application • Team's lack of general expertise • Resource insufficiency • Magnitude of potential loss • Intensity of conflicts	• Lack of stability in use of the information system • Change in the utilizing system • Lack of stability in structure and operation of the utilizing system • Changes in the use of information • Lack of user experience in utilizing system and lack of experience in type of application being proposed
Requirements Availability	• Number of users outside the organization • Number of users in the organization • Lack of user experience and support • Number of hierarchical levels occupied by users • Team's lack of	• A large number of users affect the existence and stability of requirements • A large number of users which will affect level of participation and users' feeling of responsibility in specifying requirements • Type of users doing the specifications

	expertise with task • Number of people on team • Lack of clarity of role definitions • Team diversity	

3.2. Understanding Techniques

The literature suggests requirements specification as the tactic that resolves complexity risks in software development and requirements engineering. Three types of specification techniques are represented in the reviewed literature. First, *formal* techniques that are based on rigorously defined concepts and notation schemes are promoted as the exemplary technique to resolve complexity risks (Hausler et al. 1994; Hevner et al. 1993; Jenkins et al. 1984; van Lamsweerde et al. 2000). Formalization of requirements is established as a comprehensive and all-encompassing technique (Hevner et al. 1995; van Lamsweerde et al. 2000) that involves goal-oriented modeling to explicate and include viewpoints of all stakeholders (Darke et al. 1997; Leite et al. 1991; Nuseibeh et al. 1994). Box structures offer one such formal approach to represent requirements with execution semantics that allow for simulation of the specifications (Hevner et al. 1995). Other techniques are CREWS (Haumer et al. 1998), KAOS (van Lamsweerde et al. 2000) and Z (Liu et al. 1998). Second, *combined* techniques have been promoted to facilitate end-user involvement in requirements engineering. Scenario-based requirements elicitation (Haumer et al. 1998) was, for example, found to be helpful in engaging end-users. In a similar vein, Petri net modeling was successfully integrated with adoption of use cases (Lee et al. 1998).While these combined techniques facilitate end-user involvement, the basic form of representation is still formalized to avoid fuzziness and ambiguity (Rolland

et al. 2003; Rolland et al. 1998). Quite a variety of *pragmatic* specification techniques are also presented, for example in the available surveys of requirements engineering techniques (Byrd et al. 1992; Keil et al. 1995). These specification techniques focus either on acquiring information from end-users, on studying existing systems, or on developing graphical representations of requirements, and they adopt natural language as the basic means for defining semantics. Prominent examples of these techniques are entity-relationship modeling (Haumer et al. 1998; Pedersen et al. 2001) and data flow diagraming (Larsen et al. 1992; Marakas et al. 1998; Ramesh et al. 1999).

Two types of requirements experimentation techniques were found in the literature. First, there are *iteration* techniques that facilitate learning based on specifications, prototypes, and preliminary versions of software modules. Prototyping of business software and user interfaces help developers receive direct feedback from users (Davis 1982; Keil et al. 1995; Lyytinen 1987; Watson et al. 1993). Boehm argues that iterations should continue until requirements have stabilized at which point the process can adopt a pure specification approach to support construction of the final version of the software (Boehm 1988; Mathiassen et al. 1995). Second, there are *collaboration* techniques that involve end-users in the development process (Kujala 2003). The objective of these techniques is to have end-user knowledge and experience directly influence requirements engineering activities (Duggan et al. 2004; Kujala 2003). Joint Application Design (Andrews 1991; Wetherbe 1991) exemplifies this technique and it has provided the basis for more sophisticated ways of collaboration (Vessey et al. 1994). Other examples are participatory design (Kujala 2003) and ETHICS (Duggan et al. 2004). These techniques help users and developers solve problems collaboratively and debate requirements

through various forms of structured workshops and they have been widely used by practitioners (Baskerville et al. 2001; Blackburn et al. 1996).

Finally, the literature offers three types of techniques for connecting internal as well as external end-users to the development team to help discover requirements. First, *cognitive* techniques focus on listening to and understanding the voice of the customer or other user groups inspired by approaches in marketing science, like quality function deployment (Pai 2002; Ravichandran et al. 1999; Ravichandran et al. 2000; Zultner 1993), Delphi (Davis 1982), and laddering (Browne et al. 2002; Browne et al. 2001; Davidson 2002). Second, *group* techniques, like focus group interviews (Leifera et al. 1994; Telem 1988) and Group Support Systems (Chen et al. 1991; Duggan 2003; Duggan et al. 2004; Liou et al. 1993), are suggested to take advantage of group dynamics in discovering requirements. Third, *observation* techniques help discover requirements by having end-users explain or demonstrate their work process in context. Contextual Design (Holtzblatt 1995; Jones et al. 1993) is a prime example of discovering requirements by observing end-users while they work on a day-to-day basis. This technique simultaneously addresses the problem of reaching individual users and understanding the context of use. Discovery techniques generally focus on understanding the software and its use, for example with protocol analysis or behavior analysis (Byrd et al. 1992) and through rich information about the context in which it will be adopted (Fazlollahi et al. 1991). To facilitate this process, techniques are proposed to ensure effective communication, for example using multimedia to represent requirements (Ramesh et al. 1995), multidimensional data models (Pedersen et al. 2001), semantic maps (Marakas et al. 1998), and the use of cognitive mapping (Montazemi et al. 1986).

This analysis of the literature confirms that requirements specification, experimentation, and discovery characterize important tactics in requirements engineering. In addition, the current literature suggests a more refined understanding of the techniques (i.e. formal, combined, pragmatic, iterative, collaboration, cognitive, group, and observation techniques) that are available. To further understand how well this classification of requirements techniques covers available techniques and provides a balanced view of the overall portfolio of techniques, we compared and contrasted it with other conceptions of requirements engineering techniques. Byrd et al. (1992) provide a review of requirements engineering techniques and categorize them according to their approach to research information; Keil et al. (1995) categorize techniques based on their support for development of custom or package business software. Table 4 maps our conception against these two conceptions of requirements engineering techniques. Also, we used our classification scheme to categorize the techniques that are presented in the reviewed literature as summarized in Table 5. These mappings suggest that the proposed conception of requirements engineering techniques covers the available techniques well and provides a balanced view of the overall portfolio of techniques.

Table 4 Mapping classifications of requirements engineering techniques

Tactics	Techniques	Byrd et a. (1992)	Keil et al. (1995)
Requirements Specification	Formal techniques	• Formal analysis techniques • Mapping techniques	
	Combined techniques	• Formal analysis techniques	
	Pragmatic techniques	• Unstructured Elicitation Techniques • Mapping	

		techniques	
Requirements Experimentation	Iteration techniques	• Observation Techniques	• User-interface prototyping • Requirements prototyping • Trade show • Testing
	Collaboration techniques	• Structured elicitation techniques	• Facilitated team
Requirements Discovery	Cognitive techniques	• Mapping techniques • Structured elicitation techniques	• Survey • Interview
	Group techniques	• Unstructured Elicitation Techniques • Structured elicitation techniques	• Facilitated team • Email/bulletin board • User group • Focus group
	Observation techniques	• Observation Techniques	• MIS intermediary • Support line • Usability lab • Marketing and sales • Observational study

Table 5 Categorization of requirements engineering techniques in the literature

Tactics	Techniques
Requirements Specification	***Formal techniques*** • Box structure specification and design (Hausler et al. 1994; Hevner et al. 1993; Hevner et al. 1995) • CREV (Hickey et al. 2004) • CREWS (Haumer et al. 1998) • Goal modeling oriented requirements elicitation (Darke et al. 1997; Hevner et al. 1995; Leite et al. 1991; Nuseibeh et al. 1994; van Lamsweerde et al. 2000) • KAOS (van Lamsweerde et al. 2000)

	• Lyee (Rolland et al. 2003) • Machine rule induction (Byrd et al. 1992) • Multidimensional scaling (Byrd et al. 1992) • Object oriented Z (Liu et al. 1998) • Petri nets (Lee et al. 1998) • Prime-CREWS (Haumer et al. 1998) • State charts (Haumer et al. 1998) • VDM-SL (Liu et al. 1998) • VDM ++ (Liu et al. 1998) • Z (Liu et al. 1998)
	Combined techniques • Unified modeling language (Cysneiros et al. 2004; Haumer et al. 1998) • Scenario-based requirements elicitation (Haumer et al. 1998; Rolland et al. 2003; Rolland et al. 1998) • Petri nets combined with use cases (Lee et al. 1998) • SCRAM (Hickey et al. 2004)
	Pragmatic techniques • Booch's object oriented design method (OODA) (Hevner et al. 1993) • Business information analysis and integration technique (Davis 1982) • Business process planning (BSP) (Davis 1982) • Coad and Yourdon's object oriented method (OOAD) (Hevner et al. 1993) • Data flow diagrams (Larsen et al. 1992; Marakas et al. 1998; Ramesh et al. 1999) • Decision analysis (Watson et al. 1993) • Deriving requirements from an existing system (Davis 1982) • Ends/Means analysis (Wetherbe 1991) • Entity-Relationship modeling (Haumer et al. 1998; Pedersen et al. 2001) • Goal oriented approach (Byrd et al. 1992; Darke et al. 1997) • Information systems work and analysis of changes (Davis 1982) • ISAC (Haumer et al. 1998) • Jackson system development (JSD) (Vessey et al. 1994)

	• Meyer's object oriented approach (Hevner et al. 1993) • Multidimensional data models (Pedersen et al. 2001) • Normative analysis (Watson et al. 1993) • Object oriented analysis and design (Hevner et al. 1993; Vessey et al. 1994) • OOSE (Haumer et al. 1998) • Process analysis (Watson et al. 1993) • Repertoire Grids (Byrd et al. 1992) • Rich pictures (Darke et al. 1997) • Socio-technical analysis (Davis 1982) • Seidewitz and Stark's object oriented method (Hevner et al. 1993) • Strategy set analysis (Watson et al. 1993) • Text analysis (Byrd et al. 1992) • Use cases (Lee et al. 1998) • Variance analysis (Byrd et al. 1992) • Warren-Orr diagrams (Fazlollahi et al. 1991)
Requirements Experimentation	***Iteration techniques*** • Prototyping (Byrd et al. 1992; Davis 1982; Watson et al. 1993) • Requirements prototyping (Keil et al. 1995) • Testing (Keil et al. 1995) • Trade show (Keil et al. 1995) • User-interface prototyping (Keil et al. 1995)
	Collaboration techniques • Cooperative prototyping (Leifera et al. 1994) • Clean room (Salaway 1987; Trammell et al. 1996) • ETHICS (Duggan 2003) • Facilitated team (Keil et al. 1995) • Joint application design (Andrews 1991; Kujala 2003; Wetherbe 1991) • Participatory design (Duggan 2003; Kujala 2003) • Rapid application development (Salaway 1987) • Soft systems methodology (Kujala 2003) • Structured walkthroughs (Salaway 1987)
Requirements Discovery	***Cognitive techniques*** • Affinity techique (Duggan 2003) • Card sorting (Byrd et al. 1992; Maiden et al. 1998) • Cognitive mapping (Byrd et al. 1992; Montazemi et al.

	1986) • Critical success factors (Byrd et al. 1992) • Delphi method (Davis 1982) • Laddering (Browne et al. 2002; Browne et al. 2001; Byrd et al. 1992) • Open interview (Byrd et al. 1992) • Precision model (Bostrom 1989) • Quality function deployment (Duggan 2003; Elboushi et al. 1997; Pai 2002; Ravichandran et al. 1999; Ravichandran et al. 2000; Zultner 1993) • Semantic maps (Marakas et al. 1998) • Strategic Business Objectives (Frolick et al. 1995) • Structured Interview (Byrd et al. 1992) • Surveys (Keil et al. 1995) • Teach-back interview (Byrd et al. 1992)
	Group techniques • Brainstorming (Byrd et al. 1992) • EasyWinWin (Stallinger et al. 2001) • Email/bulletin board (Keil et al. 1995) • Facilitated team (Keil et al. 1995) • Focus groups (Keil et al. 1995; Leifera et al. 1994; Telem 1988) • Future Analysis (Byrd et al. 1992) • Group Support Systems and Joint Application Design (Duggan 2003; Duggan et al. 2004; Liou et al. 1993) • Group Support Systems and Strategic Business Objectives (Frolick et al. 1995) • Guided Brainstorming (Davis 1982) • Nominal group technique (Duggan 2003) • Requirements workshops (Hickey et al. 2004) • Structured Group Elicitation Method (Bryant 1997) • User group (Keil et al. 1995)
	Observation techniques • Behavior analysis (Byrd et al. 1992) • Contextual design (Holtzblatt 1995; Jones et al. 1993; Kujala 2003) • Marketing and sales (Byrd et al. 1992) • MIS intermediary (Keil et al. 1995) • Open systems task analysis (Jones et al. 1993)

	• Protocol analysis (Byrd et al. 1992) • Support line (Keil et al. 1995) • Usability lab (Keil et al. 1995)

3.3. Understanding Principles

The above analysis of requirements engineering risks and techniques in the literature can be summarized in the following fundamental principle for managing requirements engineering risks:

Resolution Principle. Tactics for requirements engineering resolve risks as follows:

1) Requirements complexity is resolved by specification tactics including formal, combined, and pragmatic techniques.

2) Requirements reliability is resolved by experimentation tactics including iteration and collaboration techniques.

3) Requirements availability is resolved by discovery tactics that connect relevant stakeholders through cognitive, group, and observation techniques.

This Resolution Principle links individual requirements risks to individual resolution tactics. It does not, however, shed light on how to combine techniques in response to the overall risk profile or on how to adjust tactics during requirements engineering practices.

Prioritizing during requirements engineering to respond effectively to different risks is an important issue (Ramamoorthy et al. 1996). The literature offers several suggestions for how to priorities risks and tactics. Some focus on the software to be developed while others focus on the development process. Prioritizing software issues, Fitzgerald (1996) suggests to distinguish between *what* business software is expected to do, and *how* it does it. This fundamental distinction applies to how requirements are best captured and

documented. Trammel et al.(1996) note in their review that projects should be incremental to ensure continuous customer feedback from each new version of the software. Other researchers recommend repetitive refinement of the software from the what-level towards the how-level (Drehmer et al. 2001; Hausler et al. 1994).

Our focus is on the process, i.e. on how different tactics should be adopted and prioritized during the project life-cycle. Many writers cite Boehm's (1988) spiral development model for the way it combines discovery, experimentation, and specification tactics through a sequence of iterative learning cycles in which requirements are incrementally specified (Apte et al. 1990; Bersoff et al. 1991; Lyytinen 1987; Lyytinen et al. 1998; Mathiassen et al. 1995; Ropponen et al. 1997). Mathiassen et al. (1995) provides similar, but more abstract guidance in their principle of limited reduction. Their model explains how specification and experimentation can be used and combined to reduce complexity and uncertainty (Mathiassen et al. 1995). There is also agreement in the literature that projects seldom rely on one single technique (Chatzoglou et al. 1996; Davis 1982). Instead, projects adopt a mixture of techniques in response to the organizational needs and executive contingencies they face (Watson et al. 1993). Moreover, the use of each technique should be tailored to the particular context of development (Basili et al. 1988; Ropponen et al. 1997; Ropponen et al. 2000). Boehm's spiral model exemplifies, in this way, important principles for how to prioritize requirements risks and tactics during the project life-cycle. First, the model combines several tactics that are used both in parallel and sequence. Second, priority is given to certain issues over others as the life-cycle evolves (e.g. first focus on reducing risks; then focus on constructing software). Third,

the model is generic and must be adapted to the specific development context (e.g. the number of iteration cycles depend on the context).

This suggests the following principles for prioritizing requirements risks and tactics. Initially, we should attempt to identify and connect to the end-users in order to discover requirements (Duggan et al. 2004; Elboushi et al. 1997; Frolick et al. 1995) and possibly involve them in the development effort as suggested by Kujala (2003). In this way, we bridge the communication gap and make it possible to listen to the voice of customers and other end-users (Curtis et al. 1992; Curtis et al. 1988; Davidson 2002; Keil et al. 1995; Pai 2002; Ravichandran et al. 1999; Ravichandran et al. 2000; Zultner 1993). From a strong initial position in which users are connected and the context of use is appreciated, it becomes feasible to increasingly focus on explicating and validating requirements through various forms of experimentation. Finally, as requirements stabilize, it becomes feasible to increasingly focus on detailing and specifying requirements as a basis for constructing the software. The literature supports initial emphasis on requirements availability and discovery (Browne et al. 2002; Browne et al. 2001; Duggan et al. 2004; Holtzblatt 1995; Jones et al. 1993; Nunamaker et al. 1991; Ravichandran et al. 1999; Ravichandran et al. 2000; Stallinger et al. 2001) and the subsequent priority between experimentation and specification is well understood (Apte et al. 1990; Bersoff et al. 1991; Boehm 1988; Lyytinen 1987; Lyytinen et al. 1998; Mathiassen et al. 1995). We summarize these insights for prioritizing risks and tactics during requirements engineering as follows:

Prioritizing Principle. The primary focus on requirements engineering risks and tactics should gradually change as follows:

1) Requirements availability through discovery.

2) Requirements reliability through experimentation.

3) Requirements complexity through specification.

The literature finally emphasizes the importance of understanding and managing the interaction between different requirements tactics (Lyytinen et al. 1998; Mathiassen et al. 1995). Interaction occurs when adoption of a tactic influences other types of risks than it was intended to reduce. A simple example illustrates this phenomenon. If a project manager is concerned with resource risks and team risks, he might add new members to the team to reduce resource risks. Such a tactic will, however, invariably impact team risks by introducing new persons into an established team. Tactics for reducing resource risks are, therefore, intrinsically related to tactics for team risks.

The fundamental building blocks in requirements risk management are expressed in the Resolution Principle above. It suggests that projects should understand their risk profile and respond by using tactics that target each identified risk (Lyytinen et al. 1998). To do this, risk management models contain lists of risk factors to help analyze the risk profile and identify tactics to resolve identified risks. A typical approach is to determine the risks and categorize them into either high or low risks (Davis 1982; Fazlollahi et al. 1991; McFarlan 1982). The models then provide suggestions for how to address different levels of risks by using specific resolution tactics. The literature also recommends that the risk profiles should be continuously assessed to monitor how different risks interact as they are addressed and a project evolves (Chen et al. 1999; Lyytinen et al. 1996; McFarlan 1982; Quintas 1994). Risk management, if practiced in this way, therefore involves

continuous sense-and-respond activities in which risk profiles are updated and the portfolio of adopted techniques is modified or changed (Lyytinen et al. 1996).

Mathiassen et al. (1995) provides a general understanding of why this is important. They argue that we often cannot reduce one source of risk without affecting other sources. Their Principle of Limited Reduction describes how tactics to reduce uncertainty risks through experimentation generate additional information and hence increase complexity risks (and visa versa with respect to specification tactics for reducing complexity risks). The consequence of this principle is that risks should be addressed systemically because adoption of certain tactics might require adoption of complementary tactics to address adverse effects. These insights are summarized in the following principle for addressing requirement engineering risks:

Interaction Principle. Adoption of a requirements engineering tactic can require adoption of compensating tactics to reduce the adverse effect on other risks than the ones targeted by the tactic.

This analysis of principles for linking requirements engineering tactics and risks is more broadly supported by the literature than indicated above. Table 6 summarizes the selected literature that addresses issues related to each of the identified principles.

Table 6 Sources addressing principles for linking requirements tactics and risks

Principle	Sources
Resolution Principle	(Andrews 1991; Apte et al. 1990; Barki et al. 1993; Baskerville et al. 2001; Blackburn et al. 1996; Boehm et al. 1989; Bostrom 1977; Bostrom 1989; Bowen et al. 1995; Brooks 1987; Browne et al. 2002; Browne et al. 2001; Bryant 1997; Byrd et al. 1992; Chen et al. 1991; Curtis et al. 1992; Curtis et al. 1988; Cysneiros et al. 2004; Darke et al. 1997; Davidson 2002; Davis 1982; Dennis et al. 1988; Duggan 2003; Duggan et al. 2004; Elboushi et al. 1997; Fazlollahi et al. 1991; Frolick et al. 1995; Glass et al. 1992;

	Haumer et al. 1998; Hausler et al. 1994; Hevner et al. 1993; Hevner et al. 1995; Hickey et al. 2004; Hirschheim et al. 1991; Holtzblatt 1995; Houston et al. 2001; Jenkins et al. 1984; Jones et al. 1993; Keil et al. 1995; Kraut et al. 1995; Kujala 2003; Larsen et al. 1992; Lee et al. 1998; Leifera et al. 1994; Leite et al. 1991; Liou et al. 1993; Liu et al. 1998; Lyytinen 1987; Lyytinen 1988; Maiden et al. 1998; Marakas et al. 1998; Mathiassen et al. 1995; Mills 1999; Montazemi et al. 1986; Nidumolu 1995; Nunamaker et al. 1991; Nuseibeh et al. 1994; Pai 2002; Rai et al. 2000; Ramamoorthy et al. 1996; Ramesh et al. 1995; Ramesh et al. 1999; Ravichandran et al. 1999; Ravichandran et al. 2000; Rolland et al. 2003; Rolland et al. 1998; Salaway 1987; Sawyer et al. 1998; Stallinger et al. 2001; Telem 1988; Walz et al. 1993; van Lamsweerde et al. 2000; Watson et al. 1993; Vessey et al. 1994; Wetherbe 1991; Willcocks et al. 1994; Zmud 1980; Zultner 1993)
Prioritizing Principle	(Apte et al. 1990; Basili et al. 1988; Bersoff et al. 1991; Boehm 1988; Chatzoglou et al. 1996; Davis 1982; Drehmer et al. 2001; Fitzgerald 1996; Hausler et al. 1994; Lyytinen 1987; Lyytinen et al. 1998; Mathiassen et al. 1995; Ramamoorthy et al. 1996; Ropponen et al. 1997; Ropponen et al. 2000; Watson et al. 1993)
Interaction Principle	(Boehm 1988; Chen et al. 1999; Davis 1982; Fazlollahi et al. 1991; Lyytinen et al. 1996; Lyytinen et al. 1998; Mathiassen et al. 1995; McFarlan 1982; Quintas 1994)

Having analyzed the existing literature on software development and requirements engineering to understand requirements risks, requirements techniques, and principles for linking the two, we proceed to synthesize the findings by proposing a model for managing requirements engineering risks.

4. SYNTHESIZING THE FINDINGS

Webster and Watson (2002) argue that reviews should extend current theories or develop new theories. In fact, they consider this the most important part of a literature review and the part that needs careful planning and the most elaboration. For that reason, we designed our analysis of the software development and requirements engineering literature with the explicit objective of developing an up-to-date contingency model that

could set directions for future research and inform practice. The literature base, the analytical framework (see Table 2), and the questions that guided the analysis were carefully designed to help synthesize the analysis into a model. In the following, we first review available knowledge about contingency models and models for managing software risks. These insights provide the foundation for synthesizing the literature analysis into a model. We then proceed to present the rationale for and structure of a model for managing requirements engineering risks in business contexts.

4.1. Building Contingency Models

Iivari (1992) discusses the issues involved in building contingency models based on insights from organization theory (Kickert 1983; Van de Ven et al. 1985). Iivari suggests a generic framework as follows:

1) Contextual factors considered,
2) Resolution options considered,
3) Methodology used,
4) Type of fit
    a) Selection approach,
    b) Interaction approach,
    c) Systems approach,
5) Effectiveness criteria used.

We have identified requirements complexity, requirements reliability, and requirements availability as the considered contextual factors. Similarly, we have identified

requirements specification, requirements experimentation, and requirements discovery as the considered resolution options. These factors and options are further elaborated in Tables 3, 4 and 5. The methodology adopted to arrive at this understanding of situational factors and resolution options is our analysis of published journal articles within software development and requirements engineering.

Iivari (1992) offers three types of fit between contextual factors and resolution options. The selection approach suggests that requirements engineering risks determine which tactic to adopt. A situation is considered as given and tactics are adopted through managerial selection. The interaction approach suggests that fit is achieved through design of appropriate relationships between the specific situation and appropriate tactics. A design influences not only which tactics to adopt but also the way in which tactics interact with and shape the situation. The focus is, however, still on optimizing the fit between pairs of risks and tactics. The systems approach suggests that fit represents the overall consistency between multiple requirement engineering risks, requirement engineering tactics, and the resulting performance characteristics.

The unidirectional causality implied by the selection approach is simplistic (Iivari 1992) and it contradicts the dynamics implied by the identified Prioritizing and Interaction Principles. The interaction and systems approaches offer more comprehensive views of the relationship between risks and tactics that are consistent with the findings from the literature. While the interaction approach offers dialectic conception of causalities, its focus on specific pairs of factors and options can lead to unintended sub-optimizations and it is not consistent with the insights underlying the Interaction Principle. For these

reasons, we choose the systems approach as the basis for building a synthesizing model to help manage requirements engineering risks.

Finally, which effectiveness criteria to use to link factors to options, is largely determined through our choice of the systems approach. We assume, as a consequence, that that there is no one best way to approach requirements engineering in a given situation. Instead, the individual elements of a project's approach to requirements engineering should be selected and combined to achieve an internal consistency or harmony, as well as a basic consistency with the risks that a project faces (Minzberg 1983, pp. 2-3).

Existing models for managing software risks provide additional support for synthesizing the findings from the literature analysis. Iversen et al. (2004) have identified four types of such models. First, there are risk lists (e.g. Barki et al. 1993). These models contain generic risk items (often prioritized) to help managers focus on possible sources of risk; they do not offer appropriate resolution techniques. Second, there are risk-action lists (e.g. Boehm 1991). These models contain generic risk items (often prioritized), each with one or more related risk resolution technique. Third, there are risk-strategy models (e.g. McFarlan 1982). These models relate a project's risk profile to an overall strategy for addressing it. They combine comprehensive lists of risks and resolution techniques with abstract categories of risks (to arrive at a risk profile) and abstract techniques (to arrive at an overall risk management strategy). The risk profile is assessed along the risk categories (e.g., into high or low), making it possible to classify the project as being in one of a few possible situations. For each situation, the model offers a dedicated risk strategy that combines several abstract techniques. Finally, there are risk-strategy analysis approaches (e.g. Davis 1982). These approaches are similar to risk-strategy models in

offering both detailed as well as aggregate risks and resolution techniques, but they apply different heuristics. There is no model linking aggregate risks to aggregate resolution techniques. Instead, these approaches offer a stepwise process in which risks are identified and linked to techniques to form an overall risk management strategy.

Iversen et al. (2004) suggest that risk-strategy models have the most advantages from a usage point of view, but they are more difficult to build and modify than the other models. Accepting the difficulties involved in attempting to synthesize the findings from the review into such a model, we chose this option in an attempt to support practical management of requirements engineering risks as well as possible. Moreover, this choice is consistent with the adoption of a systems approach (Iivari 1992) to fit contingency factors to resolution options.

4.2. A Contingency Model

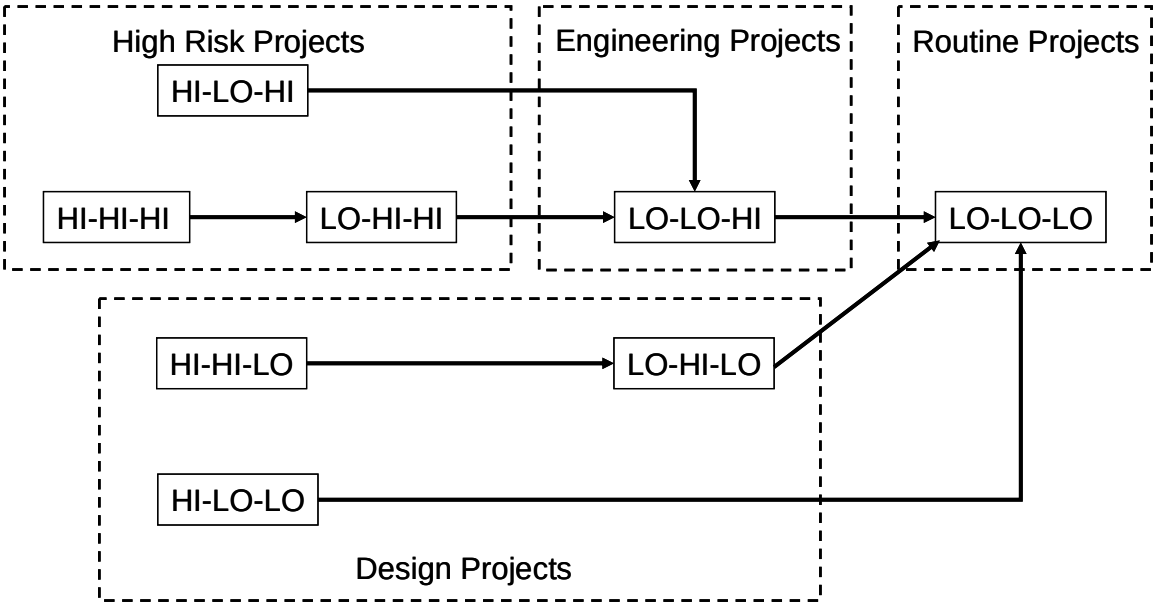
McFarlan (1982) provides the exemplary risk-strategy model in the software development literature and other models of this type have been proposed by Donaldson et al. (2001) and Keil et al. (1998). McFarlan's model (1982) distinguishes between three types of software development risks (size of project, experience with technology, and understanding of task); it suggests to assess each risk using a high-low scale; and, it proposes four basic tactics to resolve risks (external integration, internal integration, project planning, and project control). The model leads to 2^3=8 archetypical project situations and suggests for each of them a specific combination of tactics to effectively resolve risks. The model can be used repeatedly over the project life-cycle as the risk profile of a project changes. Our proposed model for managing requirements engineering risks has used McFarlan's model (1982) as template.

Adopting a high-low scale for assessing complexity, reliability, and availability risks leads to 2^3=8 different types of requirements engineering situations. Figure 1 illustrates the resulting archetypical situations and how they relate to each other as risks are resolved according to the Prioritizing Principle. Each situation is characterized by availability-reliability-complexity risks (HI=high; LO=low). Based on the characteristics of the eight situations and the relationships between them, we propose to distinguish between for types of projects: high-risk projects, engineering projects, design projects, and routine projects (see Figure 1). In the following, we review each of these, the risk profiles that characterize them, and the recommended requirements engineering tactics for addressing risks. The resulting contingency model is summarized in Table 7.

High risk projects. These projects face complex requirements while at the same time having to deal with difficult issues related to the availability and reliability of relevant information. Projects that are assessed as HI-HI-HI (type 1 in Table 7) should mainly focus on requirements discovery to ensure strong connections to would-be-users and the context in which they operate (cf. the Resolution Principle and the Prioritizing Principle). At the same time, these projects must adopt moderate levels of experimentation and specification tactics from the outset to help capture and assess information about requirements as it is discovered (cf. the Interaction Principle). It is important that these complementary tactics are not too heavily emphasized because that might create barriers towards effective discovery of requirements. Projects that are assessed as HI-LO-HI (type 2 in Table 7) should also mainly focus on requirements discovery (cf. the Resolution Principle and the Prioritizing Principle). However, as requirements are highly reliable, they only need complementary specification techniques to help capture information as it

is discovered (cf. the Interaction Principle). Finally, projects that are assessed as LO-HI-HI (type 3 in Table 7) are well connected to would-be-users and the context in which they operate. They should mainly focus on experimentation tactics to ensure reliable requirements (cf. the Resolution Principle and the Prioritizing Principle). In addition, they should adopt complementary specification tactics to document requirements as they are suggested and validated (cf. the Interaction Principle). All high risk projects have a weak understanding of the development task and they need to give high priority to external integration tactics (McFarlan 1982). As these risks are resolved, they should increasingly concentrate on internal integration, project planning, and project control to address the high complexity involved. Too early emphasis on these tactics can create barriers towards effective integration between would-be-users and the development team. In Davis' terms (1982) high risk projects involve high task uncertainty and they should adopt approaches based on combinations of experimentation and specification tactics.

Figure 1 Relation between archetypical requirements engineering situations



Engineering projects. These projects face a complex set of reliable requirements. The available requirements reflect business and user needs and they remain relatively stable over the project life-cycle. Projects that are assessed as LO-LO-HI (type 4 in Table 7) can afford to focus mainly on specification tactics (cf. the Resolution Principle). These projects face low risks related to understanding the task, but the high complexity risk suggests that they should emphasize internal integration, project planning, and project control (McFarlan 1982). According to Davis' framework (1982), engineering projects should mainly be based on specification tactics.

Design projects. These projects will eventually face relatively simple requirements, but there are serious risks related to the availability and reliability of information about requirements. The key challenge in these projects is to design a viable solution. Such projects should identify and validate requirements through interaction with would-be-users and the business context. Projects that are assessed as HI-HI-LO (type 5 in Table 7) should mainly focus on discovery tactics to interact effectively with would-be-users and the context in which they operate (cf. the Resolution Principle and the Prioritizing Principle). At the same time, these projects must adopt complementary experimentation tactics from the outset to help validate information about requirements as it is discovered (cf. the Interaction Principle). Because requirements are relatively simple, there is no need to adopt comprehensive specification tactics. Projects that are assessed as HI-LO-LO (type 6 in Table 7) should proceed in a similar fashion, except they need not concentrate on the reliability of requirements. Finally, projects that are assessed as LO-HI-LO (type 7 in Table 7) have access to relevant information about requirements, but the information is highly unreliable. These projects must emphasize experimentation tactics

to stabilize requirements (the Resolution Principle). All design projects face high risks related to understanding the task and they need to give high priority to external integration tactics (McFarlan 1982). As the complexity is low, there is little need to emphasize internal integration, project planning, and project control. In Davis' terms (1982), design projects should mainly be based on a combination of discovery and experimentation tactics.

Routine projects. Finally, there are routine projects that are assessed as LO-LO-LO (type 8 in Table 7). In these projects, requirements are available and stable, and the development team understands them well and knows from previous experience how to design and develop software that meets the requirements. Routine projects require no special attention from a requirements engineering perspective; straightforward approaches can be adopted to develop the software. McFarlan suggests that such projects should concentrate entirely on internal integration to make sure that the development team is capable and committed to develop the requested software (McFarlan 1982). Davis (1982) suggests that routine projects should be based on direct and informal interactions with would-be-users and the business context, or alternatively, if similar software is available they should be based on modifying or imitating existing software.

The distinctions and logic in Figure 1 express a synthesis of the key findings from the literature analysis. This synthesis and the elaboration into the four types of project situations provide the rationale for the contingency model summarized in Table 7. In the model, we have expressed levels of risks using the high-low scale and we have expressed the degree to which individual tactics should be emphasized in designing a comprehensive strategy for risk resolution using a weak-medium-strong scale.

Table 7 Managing requirements engineering risks

	Avai-Lability	Relia-bility	Com-plexity	Dis-covery	Experi-mentation	Speci-fication
1	High	High	High	Strong	Medium	Medium
2	High	Low	High	Strong	Weak	Medium
3	Low	High	High	Weak	Strong	Medium
4	Low	Low	High	Weak	Weak	Strong
5	High	High	Low	Strong	Medium	Weak
6	High	Low	Low	Strong	Weak	Weak
7	Low	High	Low	Weak	Strong	Weak
8	Low	Low	Low	Weak	Weak	Weak

5. DISCUSSION AND IMPLICATIONS

The results from the analysis of the literature show that the portfolio of requirements engineering research recognizes the problems involved in practice (e.g. Table 3) and provides a rich variety of techniques to guide practice (e.g. Tables 4 and 5). Most techniques focus, however, on solving particular requirements engineering problems and, only a handful of papers discuss how techniques can be combined. There is little meta level research that provides a structured understanding of the field, its problems and challenges, and the techniques available to support practice. Such research is particularly important because it provides guidance to studying the literature and to adapting insights from the literature to practice.

While there are relatively up-to-date surveys of requirements engineering techniques available (Byrd et al. 1992; Davis 1982; Keil et al. 1995; Nuseibeh et al. 2000), none of them link the identified types of techniques to different types of requirements engineering situations. In fact, the only models in the reviewed literature that can help practitioners design appropriate requirements engineering approaches date back to the early eighties (Alter et al. 1978; Davis 1982; McFarlan 1981; McFarlan 1982). As a consequence, these models do not address the shifts that have occurred in requirements engineering theory

and practices as business software is increasingly produced to markets and used by customers and business partners across organizational boundaries.

Our analysis of the literature suggests, that today's business software projects face situations involving requirements availability risks, requirements reliability risks, as well as requirements complexity risks. To address such differentiated risk profiles, the analysis suggests that practitioners should design approaches that combine requirements discovery tactics, requirements experimentation tactics, as well as requirements specification tactics. Moreover, the analysis identifies principles for applying requirements engineering tactics to resolve risks: the Resolution Principle (that helps link relevant tactics to specific risks), the Prioritizing Principle (that helps decide on which risks to focus on as a project evolves), and the Interaction Principle (that helps combine different tactics into a comprehensive strategy that addresses the risk profile as a whole).

The synthesis of these findings into a contingency model for managing requirements risks (see Table 7), identifies eight different requirements engineering risk profiles and for each of these it suggests a combination of tactics to resolve the risks. In addition, the model suggests (see Figure 1) to distinguish between four archetypical requirements engineering projects: high risk projects, engineering projects, design projects, and routine projects. Each of these poses different challenges, they call for different strategies, and they will, consequently, require development teams with different skill profiles, mindsets, collaboration patterns, and management practices. This synthesis and the underlying insights from the literature analysis have implications for both research and practice.

5.1 Implications for research

The paper has highlighted the continuously growing portfolio of techniques and still more differentiated risk profile involved in requirements engineering. As the literature provides little guidance in navigating this increasingly complex landscape, we encourage researchers to (cf. Hickey et al. 2004): 1) deepen our understanding of the characteristics that differentiate today's requirements engineering projects; 2) develop surveys of available techniques that help distinguish them with respect to their usefulness in different types of requirements engineering situations; and, 3) further develop and validate contingency models for managing requirements engineering risks.

The first research challenge could start out from available knowledge about software risks in general (Barki et al. 1993; Lyytinen et al. 1998) and requirements engineering risks in particular (Davis 1982) (see Table 4). General risk measures need to be projected into the requirements engineering space and requirements engineering risk measures need to be updated to reflect today's practices. The goal of these efforts should be to develop useful categories of requirements risks and related measures that can be used to identify and assess risk profiles in requirements engineering practice. One approach would be to develop a generic set of requirements risks across all types of projects and software. Another approach would be to categorize types of software (e.g. custom versus package) (Keil et al. 1995) or types of projects (e.g. in-house or outsourced) to develop more specialized measures of the involved risks.

The second research challenge is to further develop and refine available attempts to categorize requirements engineering techniques (Byrd et al. 1992; Darke et al. 1997; Davis 1982; Keil et al. 1995; Nuseibeh et al. 2000). The goal of this research is to take

stock of the available portfolio of techniques and provide guidance on how to categorize, assess, and select specific techniques. Such insights can guide practical requirements engineering as well as continued efforts to develop a better and more comprehensive portfolio of techniques. This research should survey and assess techniques beyond those presented in the analyzed literature, it should critically contrast the espoused benefits and actual effects of using the techniques, and it should differentiate techniques based on their ability to resolve specific types of requirements risks, for example as suggested in Table 5.

Finally, the third research challenge should further develop and apply contingency models (including the one proposed in this paper) to practical management of requirements engineering risks. This would call for empirical work on validating the applicability of our proposed principles and tactics in real world situations under different contextual factors. These efforts should be tightly linked to requirements engineering practices based on a variety of research approaches: surveys of how practitioners select and combine requirements engineering techniques (Blackburn et al. 1996; Chatzoglou et al. 1996; Rai et al. 2000); case studies of the relationship between practices and techniques, of how and why techniques are adopted and combined, and of the effects that techniques have on resolving requirements risks (Browne et al. 2001; Darke et al. 1997; Elboushi et al. 1997; Haumer et al. 1998; Kujala 2003; Liu et al. 1998). These activities could be followed by design research (Hevner et al. 2004) studies to develop complementary methods to better cover the portfolio of requirements engineering risks. Finally, action research projects could develop, apply, modify, and validate proposed models for managing requirements engineering risks in business contexts. In support of

the latter type of research, Iversen et al. (2004) provides a comprehensive action research approach to develop risk management practices within the information systems and software engineering disciplines. This approach can be used to develop models tailored to a particular business (e.g. that provide package software solutions for markets) or, models to be applied in particular types of projects (e.g. high-risk, engineering, or design projects).

5.2 Implications for practice

While the review is limited to the academic literature on requirements engineering, the findings have direct impacts on development of business software. Practitioners are advised to distinguish between different types of requirements engineering projects and situations. The proposed contingency model provides guidelines for how to do so. First, practitioners should assess each new requirements engineering project. To that end they should study Table 4 and use the suggested measures as indicators to help understand the risk profile of the project. Second, they can use Table 7 to arrive at an abstract strategy to address the risks they face. Third, they then translate the strategy into concrete plans for action by identifying specific techniques corresponding to the suggested combination of requirements engineering tactics (see table 7). This can be done by critically reviewing the techniques they are currently using or by exploring alternative techniques in Table 5. Finally as suggested by Figure 1, practitioners are encouraged to reassess risks and adjust requirements engineering strategy as they go along. Lyytinen et al. (1996; 1998) argue that software risk management is a very inexpensive and low-risk technology. Risk management practices help shape practitioners' attention more sharply on the challenges

they face (Lyytinen et al. 1998) and they provide useful guidance on what approaches to adopt.

In summary, we encourage researchers to consider and help bridge the gap between the portfolio of available techniques and the profile of risks that practitioners face in requirements engineering. At the same time we encourage practitioners to adopt risk management practices to help design approaches to requirements engineering that apply to the type of project and situations they are involved in.

6. LIMITATIONS

This research has, as any other scientific efforts, shortcomings. Most importantly, we have limited ourselves to analyze and synthesize scientific papers published in information systems and software engineering journals. The subject of why, when, and how requirement engineering techniques should be used in different types of project situations lends itself strongly towards empirical research. The literature on the subject is, however, extensive, and we felt a need to carefully review this body of knowledge before engaging ourselves in further empirical studies. Also, we have not included analyses of the extensive practitioner oriented literature on requirements engineering. Such analyses could provide additional and valuable insights into the types of techniques that are available for requirements engineering and into the espoused theories about the applicability of different types of techniques. Finally inspired by Webster and Watson (2002), we approached the review of the literature with the ambition to extend current theories. For that reason, we designed the literature analysis with the specific goal of developing an up-to-date contingency model for managing requirements engineering

risks. While this approach has helped us focus the analysis, it has also given us a specific and limited perspective on the extensive knowledge that is available about requirements engineering.

7. CONCLUSION

The objective of this research was to analyze what we know about requirements engineering risks and techniques in the context of developing business software, and to synthesize the insights from the analysis into an up-to-date understanding of why, when, and how requirement engineering techniques should be used in different types of project situations. To that end, we developed a rigorous procedure that helped us identify 91 scientific papers on the subject in leading information systems and software engineering journals. We also adopted a simple conceptual framework to structure the analysis of the literature. The literature analysis led to a review the risks involved in requirements engineering, the techniques that are available to resolve these risks, and the principles by which techniques can be applied to resolve risks. The findings from the analysis were subsequently synthesized into a contingency model for managing requirements engineering risks. The model has implications for future research and it suggests how practitioners can use insights from the literature to navigate the requirements engineering landscape.

REFERENCES

Alter, S., and Ginzberg, M. "Managing uncertainty in MIS implementation," *Sloan Management Review* (20:1) 1978, pp 23-31.

Andrews, D.C. "JAD: a crucial dimension for rapid applications development.," *Journal of Systems Management* (42:3) 1991, pp 23-31.

Apte, U., Sankar, C.S., Thakur, M., and Turner, J.E. "Reusability-Based Strategy for Development of Information Systems: Implementation Experience of a Bank," *MIS Quarterly* (14), December 1990, pp 421-433.

Barki, H., Rivard, S., and Talbot, J. "Toward an Assessment of Software Development Risk," *Journal of Management Information Systems* (10:2) 1993, pp 203-225.

Basili, V.R., and Rombach, H.D. "The TAME project: towards improvement-oriented software environments," *IEEE Trans. Software Engineering* (14:6), June 1988, pp 758-773.

Baskerville, R., Levine, L., Pries-Heje, J., Ramesh, B., and Slaughter, S. "How Internet software companies negotiate quality," *Computer* (34:5), May 2001, pp 51-+.

Bersoff, E.H., and Davis, A.M. "Impacts of Life-Cycle Models on Software Configuration Management," *Communications of the Acm* (34:8), Aug 1991, pp 104-118.

Blackburn, J.D., Scudder, G.D., and VanWassenhove, L.N. "Improving speed and productivity of software development: A global survey of software developers," *IEEE Transactions on Software Engineering* (22:12), Dec 1996, pp 875-885.

Boehm, B. "A Spiral model of software development and enhancement," *IEEE Computer* (21:5) 1988, pp 61-72.

Boehm, B. "Software Risk Management: Principles and Practices," *IEEE Software* (8:1), January/February 1991, pp 32-41.

Boehm, B., and Ross, R. "Theory-W software project management principles and examples," *IEEE Trans. Software Engineering* (15:7), July 1989, pp 902-916.

Bostrom, R.P. "MIS Problems and Failures: A Socio-Technical Perspective PART 1:THE CAUSES," *MIS Quarterly* (1:3), September 1977, pp 17-32.

Bostrom, R.P. "Successful application of communication techniques to improve the systems development process," *Information & Management* (16:5), May 1989, pp 279-275.

Bowen, J.P., and Hinchey, M.G. "10-Commandments of Formal Methods," *IEEE Computer* (28:4), Apr 1995, pp 56-63.

Broadbent, M., and Koenig, M.E.D. "Information and Information Technology Management," *Annual Review of Information Science and Technology* (23) 1988, pp 237-270.

Brooks, F.P. "No Silver Bullet - Essence and Accidents of Software Engineering," *IEEE Computer* (20:4), April 1987, pp 10-19.

Browne, G.J., and Ramesh, V. "Improving information requirements determination: a cognitive perspective," *Information & Management* (39:8), Sep 2002, pp 625-645.

Browne, G.J., and Rogich, M.B. "An empirical investigation of user requirements elicitation: Comparing the effectiveness of prompting techniques," *Journal of Management Information Systems* (17:4), Spr 2001, pp 223-249.

Bryant, J. "Requirements capture using SODA," *European Journal of Information Systems* (6:3), Sep 1997, pp 155-163.

Byrd, T.A., Cossick, K.L., and Zmud, R.W. "A Synthesis of Research on Requirements Analysis and Knowledge Acquisition Techniques," *Mis Quarterly* (16:1), Mar 1992, pp 117-138.

Chatzoglou, P.D., and Macaulay, L.A. "Requirements capture and IS methodologies," *Information Systems Journal* (6:3), Jul 1996, pp 209-225.

Chen, J.Y.J., and Chou, S.C. "Consistency management in a process environment," *Journal of Systems and Software* (47:2-3), Jul 1 1999, pp 105-110.

Chen, M., and Nunamaker Jr., J.F. "The Architecture and Design of a Collaborative Environment for Systems Definition," *Data Base* (22:1-2), Winter-Spring 1991, pp 22-29.

Curtis, B., Kellner, M.I., and Over, J. "Process modeling," *Communications of ACM* (35:9), September 1992, pp 75-90.

Curtis, B., Krasner, H., and Iscoe, N. "A field study of the software design process for large systems," *Communications of ACM* (31:11), November 1988, pp 1268-1287.

Cysneiros, L.M., and Leite, J.C.S.D. "Nonfunctional requirements: From elicitation to conceptual models," *Ieee Transactions on Software Engineering* (30:5), May 2004, pp 328-350.

Darke, P., and Shanks, G. "User viewpoint modelling: understanding and representing user viewpoints during requirements definition," *Information Systems Journal* (7:3), Jul 1997, pp 213-239.

Davidson, E.J. "Technology frames and framing: A socio-cognitive investigation of requirements determination," *MIS Quarterly* (26:4), Dec 2002, pp 329-358.

Davis, G. "Strategies for information requirements determination," *IBM Systems Journal* (21:1) 1982, pp 4-31.

Dennis, A.R., George, J.F., Jessup, L.M., Nunamaker Jr., J.F., and Vogel, D.R. "Information Technology to Support Electronic Meetings," *MIS Quarterly* (12:4) 1988, pp 591-624.

Donaldson, S.E., and Siegel, S.G. *Successful Software Development* Prentice Hall, Upper Saddle River, NJ, 2001.

Drehmer, D.E., and Dekleva, S.M. "A note on the evolution of software engineering practices," *Journal of Systems and Software* (57:1), Apr 27 2001, pp 1-7.

Duggan, E.W. "Generating systems requirements with facilitated group techniques," *Human-Computer Interaction* (18:4) 2003, pp 373-394.

Duggan, E.W., and Thachenkary, C.S. "Integrating nominal group technique and joint application development for improved systems requirements determination," *Information & Management* (41:4), Mar 2004, pp 399-411.

Elboushi, M.I., and Sherif, J.S. "Object-oriented software design utilizing quality function deployment," *Journal of Systems and Software* (38:2), Aug 1997, pp 133-143.

Fazlollahi, B., and Tanniru, M.R. "Selecting a Requirement Determination Methodology-Contingency Approach Revisited," *Information & Management* (21:5), Dec 1991, pp 291-303.

Fitzgerald, B. "Formalized systems development methodologies: A critical perspective," *Information Systems Journal* (6:1), Jan 1996, pp 3-23.

Frolick, M.N., and Robichaux, B.P. "Eis Information Requirements Determination - Using a Group Support System to Enhance the Strategic Business Objectives Method," *Decision Support Systems* (14:2), Jun 1995, pp 157-170.

Gillenson, M., and Stutz, J. "Academic Issues in MIS: Journals and Books," *MIS Quarterly* (15:4) 1991, pp 147-452.

Glass, R.L., Vessey, I., and Conger, S.A. "Software Tasks - Intellectual or Clerical," *Information & Management* (23:4), Oct 1992, pp 183-191.

Grudin, J. "Interactive Systems - Bridging the Gaps between Developers and Users," *Computer* (24:4), Apr 1991, pp 59-69.

Hardgrave, B., and Walstrom, K. "Forums for Management Information Systems Scholars," *Communications of ACM* (38:3) 1997, pp 93-102.

Haumer, P., Pohl, K., and Weidenhaupt, K. "Requirements elicitation and validation with real world scenes," *Ieee Transactions on Software Engineering* (24:12), Dec 1998, pp 1036-1054.

Hausler, P.A., Linger, R.C., and Trammell, C.J. "Adopting Cleanroom Software Engineering with a Phased Approach," *IBM Systems Journal* (33:1) 1994, pp 89-109.

Hevner, A.R., March, S.T., and Park, J. "Design Research in Information Systems Research," *Mis Quarterly* (28:1) 2004, pp 75-105.

Hevner, A.R., and Mills, H.D. "Box-structured methods for systems development wirh objects," *IBM Systems Journal* (32:2) 1993, pp 232-251.

Hevner, A.R., and Mills, H.D. "Box-Structured Requirements Determination Methods," *Decision Support Systems* (13:3-4), Mar 1995, pp 223-239.

Hickey, A.M., and Davis, A. "A Unified Model of Requirements Elicitation," *Journal of Management Information Systems* (20:4) 2004, p 65–84.

Hirschheim, R., and Newman, M. "Symbolism and Information Systems Development: Myth, Metaphore and Magic," *Information Systems Research* (2:1) 1991, pp 29-62.

Holsapple, C., Johnson, L., Manakyan, H., and Tanner, J. "Business Computing Research Journals: A Normalized Citation Analysis," *Journal of Management Information Systems* (11:1) 1994, pp 131-140.

Holtzblatt, K. "Requirements Gathering: The Human Factor," *Communications of ACM* (38:5) 1995, pp 31-33.

Houston, D.X., Mackulak, G.T., and Collofello, J.S. "Stochastic simulation of risk factor potential effects for software development risk management," *Journal of Systems and Software* (59:3), Dec 15 2001, pp 247-257.

Iivari, J. "The organizational fit of information systems," *Journal of Information Systems* (2) 1992, pp 3-29.

Iversen, J.H., Mathiassen, L., and Nielsen, P.A. "Managing Risk in Software Process Improvement: An Action Research Approach," *Mis Quarterly* (28:3), September 2004, pp 395-433.

Jayaratna, N. *Understanding and Evaluating Methodologies* McGraw Hill, London, 1994.

Jenkins, A.M., Naumann, J.D., and Wetherbe, J.C. "Empirical investigation of systems development practices and results," *Information & Management* (7:2), April 1984, pp 73-82.

Jones, R.M., Candy, L., and Edmonds, E.A. "Knowledge-Based System Requirements," *Knowledge-Based Systems* (6:1), Mar 1993, pp 31-37.

Katerattanakul, P., Han, B., and Hong, S. "Objective Quality Ranking of Computing Journals," *Communications of ACM* (46:10), October 2003, pp 111-114.

Keil, M., and Carmel, E. "Customer-developer links in software development," *Communications of the Acm* (38:5) 1995, pp 33-44.

Keil, M., Cule, P.E., Lyytinen, K., and Schmidt, R.C. "A Framework for Identifying Software Project Risks," *Communications of the Acm* (41:11) 1998, pp 76-83.

Kickert, W.J.M. "Research note: Research models underlying situational dependency," *Organization Studies* (4) 1983, pp 55-72.

Kraut, R.E., and Streeter, L.A. "Coordination in software development," *Communications of ACM* (38:3), March 1995, pp 69-81.

Kujala, S. "User involvement: a review of the benefits and challenges," *Behaviour & Information Technology* (22:1), Jan-Feb 2003, pp 1-16.

Larsen, T.J., and Naumann, J.D. "An Experimental Comparison of Abstract and Concrete Representations in Systems-Analysis," *Information & Management* (22:1), Jan 1992, pp 29-40.

Lee, W.J., Cha, S.D., and Kwon, Y.R. "Integration and analysis of use cases using modular Petri nets in requirements engineering," *Ieee Transactions on Software Engineering* (24:12), Dec 1998, pp 1115-1130.

Leifera, R., Leeb, S., and Durgeea, J. "Deep structures: Real information requirements determination," *Information & Management* (27:5), November 1994, pp 275-285.

Leite, J., and Freeman, P.A. "Requirements Validation through Viewpoint Resolution," *Ieee Transactions on Software Engineering* (17:12), Dec 1991, pp 1253-1269.

Liou, Y.I., and Chen, M. "Using group support systems and joint application development for requirement specification," *Journal of Management Information Systems* (10:3) 1993, pp 25-41.

Liu, S., Offutt, A.J., Ho-Stuart, C., Sun, Y., and Ohba, M. "SOFL: A formal engineering methodology for industrial applications," *Ieee Transactions on Software Engineering* (24:1), Jan 1998, pp 24-45.

Lyytinen, K. "Different Perspectives on Information-Systems - Problems and Solutions," *Computing Surveys* (19:1), Mar 1987, pp 5-46.

Lyytinen, K. "Expectation Failure Concept and Systems Analysts View of Information-System Failures - Results of an Exploratory-Study," *Information & Management* (14:1), Jan 1988, pp 45-56.

Lyytinen, K., Mathiassen, L., and Ropponen, J. "A framework for software risk management," *Journal of Information Technology* (11:4), Dec 1996, pp 275-285.

Lyytinen, K., Mathiassen, L., and Ropponen, J. "Attention shaping and software risk - A categorical analysis of four classical risk management approaches," *Information Systems Research* (9:3), Sep 1998, pp 233-255.

Maiden, N.A.M., and Hare, M. "Problem domain categories in requirements engineering," *International Journal of Human-Computer Studies* (49:3), Sep 1998, pp 281-304.

Marakas, G.M., and Elam, J.J. "Semantic structuring in analyst acquisition and representation of facts in requirements analysis," *Information Systems Research* (9:1), Mar 1998, pp 37-63.

Mathiassen, L., Seewaldt, T., and Stage, J. "Prototyping and Specifying: Principles and Practices of a Mixed Approach," *Scandinavian Journal of Information Systems* (7:1) 1995, pp 55-72.

Mathiassen, L., and Stage, J. "The Principle of Limited Reduction in Software Design," *Information, Technology and People* (6:2) 1992.

McFarlan, F.W. "Portfolio Approach to Information Systems," *Harvard Business Review* (59:5) 1981, pp 142-150.

McFarlan, W. "Portfolio Approach to Information Systems," *Journal of Systems Management* (33:1) 1982, pp 12-19.

Mills, H.D. "The management of software engineering - Part I: Principles of software engineering (Reprinted from IBM Systems Journal, vol 19, 1980)," *IBM Systems Journal* (38:2-3) 1999, pp 289-295.

Minzberg, H. *Structure in fives: designing effective organizations* Prentice-Hall International, New Jersey, 1983.

Montazemi, A.R., and Conrath, D.W. "The Use of Cognitive Mapping for Information Requirements Analysis," *MIS Quarterly* (10:1) 1986, pp 44-56.

Mylonopoulos, N., and Theoharakis, V. "On-Site: Global Perceptions of IS Journals," *Communications of ACM* (44:9), September 2001, pp 29-33.

Nidumolu, S. "The Effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable," *Information Systems Research* (6:3), September 1995, pp 191-219.

Nunamaker, J.F., Dennis, A.R., Valacich, J.S., Vogel, D., and George, J.F. "Electronic meeting systems to Support Group Work," *Communications of the Acm* (34:7) 1991, pp 40-61.

Nuseibeh, B., and Easterbrook, S. "Requirements engineering: a roadmap," Future of Software Engineering, ICSE 2000, ACM Press, Limerick, Ireland, 2000, pp. 35-46.

Nuseibeh, B., Kramer, J., and Finkelstein, A. "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification.," *IEEE Trans. Software Engineering* (20:10) 1994, pp 760-773.

Pai, W.C. "A quality-enhancing software function deployment model," *Information Systems Management* (19:3), Sum 2002, pp 20-24.

Pedersen, T.B., Jensen, C.S., and Dyreson, C.E. "A foundation for capturing and querying complex multidimensional data," *Information Systems* (26:5), Jul 2001, pp 383-423.

Peffers, K., Gengler, C.E., and Tuunanen, T. "Extending critical success factors methodology to facilitate broadly participative information systems planning," *Journal of Management Information Systems* (20:1), Sum 2003a, pp 51-85.

Peffers, K., and Tang, Y. "Identifying and Evaluating the Universe of Outlets for Information Systems Research: Ranking the Journals," *Journal of Information Technology Theory and Application* (5:1) 2003b, pp 63-84.

Quintas, P. "A Product-Process Model of Innovation in Software-Development," *Journal of Information Technology* (9:1), Mar 1994, pp 3-17.

Rai, A., and Al-Hindi, H. "The effects of development process modeling and task uncertainty on development quality performance," *Information & Management* (37:6), Sep 2000, pp 335-346.

Ramamoorthy, C.V., and Tsai, W.T. "Advances in software engineering," *Computer* (29:10), Oct 1996, pp 47-&.

Ramesh, B., and Sengupta, K. "Multimedia in a Design Rationale Decision-Support System," *Decision Support Systems* (15:3), Nov 1995, pp 181-196.

Ramesh, V., and Browne, G.J. "Expressing casual relationships in conceptual database schemas," *Journal of Systems and Software* (45:3), Mar 15 1999, pp 225-232.

Ravichandran, T., and Rai, A. "Total quality management in information systems development: Key constructs and relationships," *Journal of Management Information Systems* (16:3), Win 1999, pp 119-155.

Ravichandran, T., and Rai, A. "Quality management in systems development: An organizational system perspective," *Mis Quarterly* (24:3), Sep 2000, pp 381-415.

Rolland, C., Souveyet, C., and Ayed, M.B. "Guiding Lyee user requirements capture," *Knowledge-Based Systems* (16:7-8), Nov 2003, pp 351-359.

Rolland, C., Souveyet, C., and Ben Achour, C. "Guiding goal modeling using scenarios," *Ieee Transactions on Software Engineering* (24:12), Dec 1998, pp 1055-1071.

Ropponen, J., and Lyytinen, K. "Can software risk management improve system development: An exploratory study," *European Journal of Information Systems* (6:1), Mar 1997, pp 41-50.

Ropponen, J., and Lyytinen, K. "Components of software development risk: How to address them? A project manager survey," *Ieee Transactions on Software Engineering* (26:2), Feb 2000, pp 98-112.

Salaway, G. "An Organizational Learning Approach to Information Systems Development," *Mis Quarterly* (20:1) 1987, pp 244-264.

Sawyer, S., and Guinan, P.J. "Software development: Processes and performance," *IBM Systems Journal* (37:4) 1998, pp 552-569.

Stallinger, F., and Grunbacher, P. "System dynamics modelling and simulation of collaborative requirements engineering," *Journal of Systems and Software* (59:3), Dec 15 2001, pp 311-321.

Telem, M. "Information Requirements Specification I: Brainstorming a Collective Decision- Making Approach," *Information Processing & Management* (24) 1988, pp 549-557.

Trammell, C.J., Pleszkoch, M.G., Linger, R.C., and Hevner, A.R. "The incremental development process in Cleanroom software engineering," *Decision Support Systems* (17:1), April 1996, pp 55-71.

Walz, D., Elam, J., and Curtis, B. "Inside a software design team: Knowledge acquisistion, Sharing and Integration," *Communications of the Acm* (36:10) 1993, pp 62-77.

Van de Ven, A.H., and Drazin, R. "The concept of fit in contingency theory," *Research in Organizational Behaviour* (7) 1985, pp 333-365.

van Lamsweerde, A., and Letier, E. "Handling obstacles in goal-oriented requirements engineering," *Ieee Transactions on Software Engineering* (26:10), Oct 2000, pp 978-1005.

Watson, H.J., and Frolick, M.N. "Determining information requirements for an EIS," *MIS Quarterly* (17:3) 1993, pp 255-269.

Webster, J., and Watson, R.T. "Analyzing the Past to Prepare for the Future: Writing a Literature Review," *MIS Quarterly* (26:2), June 2002, pp xiii-xx.

Vessey, I., and Conger, S.A. "Requirements Specification: Learning Object, Process, and Data Methodologies," *Communications of the Acm* (37:5) 1994, pp 102-113.

Wetherbe, J.C. "Executive Information Requirements - Getting It Right," *Mis Quarterly* (15:1), Mar 1991, pp 51-65.

Whitman, M., Hendrickson, A., and Townsend, A. "Research Commentary. Academic Rewards for Teaching, Research and service: Data and Discourse," *Information Systems Research* (10:2) 1999, pp 99-109.

Willcocks, L., and Margetts, H. "Risk assessment and information systems," *European Journal of Information Systems* (3:2), April 1994, pp 127-138.

Zmud, R.W. "Management of Large Software Development Efforts," *MIS Quarterly* (4:2), June 1980, pp 45-55.

Zultner, R.E. "TQM for technical teams," *Communications of the Acm* (36:10), October 1993, pp 79-91.

APPENDIX 1 - JOURNAL LIST FOR LITERATURE REVIEW

1. ACM Computing Surveys
2. ACM SIGecom Exchanges
3. ACM Trans. on Database Systems
4. ACM Trans. on Information Systems
5. AI Magazine
6. Artificial Intelligence
7. Australian J. of Information Systems
8. Behavior & Information Technology
9. Communications of the ACM
10. Communications of the AIS
11. Computer Journal
12. Computer Supported Cooperative Work
13. DATA BASE
14. Decision Support Systems
15. Electronic Commerce Research and Application
16. Electronic Markets
17. e-Service J.
18. European J. of Information Systems
19. Expert Systems w. Applications
20. Human-Computer Interaction
21. IBM Systems J.
22. IEEE Computer
23. IEEE Trans. on Software Engineering
24. Information & Management
25. Information and Organization
26. Information Processing & Management
27. Information Research
28. Information Resources Management J.
29. Information Systems
30. Information Systems Frontiers
31. Information Systems J.
32. Information Systems Management
33. Information Systems Research
34. Information Technology & People
35. Information Technology and Management
36. Informing Science
37. Int. J. of Human-Computer Studies
38. Int. J. of Electronic Commerce
39. Int. Journal of Human Computer Study
40. Int. Journal of Information Management
41. J. of Computer and System Sciences
42. J. of Computer Information Systems
43. J. of Computer IS
44. J. of Database Management
45. J. of End-User Computing
46. J. of Global Information Management
47. J. of Global Information Technology Management
48. J. of Information Systems Education
49. J. of Information Technology
50. J. of IT Cases & Applications
51. J. of Management Information Systems
52. J. of Strategic Information Systems
53. J. of Strategic IS
54. J. of Systems and Software
55. J. of the ACM
56. J. of the Association for Information Systems
57. J. of Information Technology Theory & Application
58. J. of Information Systems Management
59. J. of Information Technology
60. J. of Information Technology Education
61. J. of Management
62. J. of Organizational Computing and EC
63. Knowledge Based Systems
64. MIS Quarterly
65. MISQ Discovery
66. Scandinavian J. of Information Systems
67. The Information Society
68. Wirtschaftsinformatik

APPENDIX 2 – TABLE OF REVIEWED ARTICLES

Journal	Article
IEEE Trans. On Software Engineering (12)	Basili V.R. and H.D. Rombach (1988)
	Blackburn J.D., G.D. Scudder and L.N. VanWassenhove (1996)
	Boehm B. and R. Ross (1989)
	Cysneiros, L.M. and Leite, J. (20004)
	Haumer P. , K. Pohl and K. Weidenhaupt (1998)
	Lee W.J., S.D. Cha and Y.R. Kwon (1998)
	Leite J. and P.A. Freeman (1991)
	Liu S. , A.J. Offutt C., Ho-Stuart Y. Sun and M. Ohba (1998)
	Nuseibeh B. , J. Kramer and A. Finkelstein (1994)
	Rolland C., C. Souveyet and C. Ben Achour (1998)
	Ropponen J. and K. Lyytinen (2000)
	van Lamsweerde A. and E. Letier (2000)
MIS Quarterly (11)	Apte U. , C.S. Sankar M., Thakur and J.E. Turner (1990)
	Bostrom R.P. (1977)
	Byrd T.A., K.L. Cossick and R.W. Zmud (1992)
	Davidson E.J. (2002)
	Dennis A.R., J.F. George ,L.M. Jessup, J.F. Nunamaker Jr. and D.R. Vogel (1988)
	Montazemi A.R. and D.W. Conrath (1986)
	Ravichandran T. and A. Rai (2000)
	Salaway G. (1987)
	Watson H.J. and M.N. Frolich (1993)
	Wetherbe J.C. (1991)
	Zmud R.W. (1980)
Communications of the ACM (10)	Bersoff E.H. and A.M. Davis (1991)
	Curtis B., H. Krasner and N. Iscoe (1988)
	Curtis B., M.I. Kellner and J. Over (1992)
	Holtzblatt K. (1995)
	Keil M. and E. Carmel (1995)
	Kraut R.E. and L.A. Streeter (1995)
	Nunamaker J.F. , A.R. Dennis, J.S. Valacich, D. Vogel and J.F. George (1991)
	Vessey I. and S.A. Conger (1994)
	Walz D. , J. Elam and B. Curtis (1993)
	Zultner R.E. (1993)
Information & Management (10)	Bostrom R.P. (1989)
	Browne G.J. and V. Ramesh (2002)
	Duggan E.W. and C.S. Thachenkary (2004)
	Fazlollahi B. and M.R. Tanniru (1991)
	Glass R.L., I. Vessey and S.A. Conger (1992)
	Jenkins A.M. , J.D. Naumann and J.C. Wetherbe (1984)
	Larsen T.J. and J.D. Naumann (1992)
	Leifera R. , S. Leeb and J. Durgeea (1994)
	Lyytinen K. (1988)
	Rai A. and H. Al-Hindi (2000)
J. of Systems and Software (6)	Chen J.Y.J.and S.C. Chou (1999)

	Drehmer D.E. and S.M. Dekleva (2001)
	Elboushi M.I. and J.S. Sherif (1997)
	Houston D.X., G.T. Mackulak and J.S. Collofello (2001)
	Ramesh V. and G.J. Browne (1999)
	Stallinger F. and P. Grunbacher (2001)
IBM Systems J. (5)	Davis G. (1982)
	Hausler P.A., R.C. Linger and C.J. Trammell (1994)
	Hevner A.R. and H.D. Mills (1993)
	Mills H.D. (1999)
	Sawyer S. and P.J. Guinan (1998)
IEEE Computer (5)	Baskerville R., L. Levine, J. Pries-Heje, B. Ramesh and S. Slaughter (2001)
	Boehm B. (1988)
	Bowen J.P. and M.G. Hinchey (1995)
	Brooks F.P. (1987)
	Ramamoorthy C.V. and W.T. Tsai (1996)
J. of Management Information Systems (5)	Barki H. S. Rivard and J. Talbot (1993)
	Browne G.J. and M.B. Rogich (2001)
	Liou Y.I. and M. Chen (1993)
	Hickey and Davis (2004)
	Ravichandran T. and A. Rai (1999)
Decision Support Systems (4)	Frolick M.N. and B.P. Robichaux (1995)
	Hevner A.R. and H.D. Mills (1995)
	Ramesh B. and K. Sengupta (1995)
	Trammell C.J., M.G. Pleszkoch, R.C. Linger and A.R. Hevner (1996)
European J. of Information Systems (4)	Bryant J. (1997)
	Chatzoglou P.D. and L.A. Macaulay (1996)
	Ropponen J. and K. Lyytinen (1997)
	Willcocks L. and H. Margetts (1994)
Information Systems Research (4)	Hirschheim R. and M. Newman (1991)
	Lyytinen K., L. Mathiassen and J. Ropponen (1998)
	Marakas G.M. and J.J. Elam (1998)
	Nidumolu S. (1995)
Information Systems J. (3)	Fitzgerald B. (1996)
	Darke P. and G. Shanks (1997)
	Lyytinen K., L. Mathiassen and J. Ropponen (1996)
J. of Information Technology (2)	Andrews D.C. (1991)
	Quintas P. (1994)
Knowledge-Based Systems (2)	Jones R.M., L. Candy and E.A. Edmonds (1993)
	Rolland C., C. Souveyet and M.B. Ayed (2003)
ACM Computing Surveys (1)	Lyytinen K. (1987)
Behavior & Information Technology (1)	Kujala S. (2003)
DATA BASE (1)	Chen M. and J.F. Nunamaker Jr. (1991)
Human-Computer Interaction (1)	Duggan E.W. (2003)
Information Systems Management (1)	Pai W.C. (2002)
Int. J. of Human-Computer Studies (1)	Maiden N.A.M. and M. Hare (1998)
J. of Systems Management (1)	McFarlan W. (1982)
Scandinavian J. of Information Systems (1)	Mathiassen L., T. Seewaldt and J. Stage (1995)