TECHNISCHE
UNIVERSITÄT
MÜNCHEN

Software Development Risk Management Model- a goal-driven approach

Shareeful Islam

Lehrstuhl für Software & Systems Engineering
Institut für Informatik
Technische Universität München

Technische Universität München
Institut für Informatik
Lehrstuhl für Software & Systems Engineering

Software Development Risk Management Model- a goal-driven approach

Shareeful Islam

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Hans Michael Gerndt
Prfer der Dissertation:	
1.	Univ.-Prof. Dr. Dr. h.c. Manfred Broy
2.	Univ.-Prof. Dr. Martin Bichler

Die Dissertation wurde am 11.11.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 25.03.2011 angenommen.

Abstract

Every software project by its inherent nature is unique and contains significant numbers of uncertainties from various perspectives such as time-to-market, budget and schedule estimation, product deployment or maintenance. If failing to control these uncertainties, it imposes potential risks not only during the development phases but also throughout the life cycle of the product. Software risk management is an effective tool to control these risks and contributes to increase the likelihood of project success. Risk management needs to be integrated as early as possible from a holistic perspective into the development. However a comprehensive risk management practice is not always possible due to resource problems, more emphasize on budget and schedule constraints and difficulties to concretely estimate the benefit of risk management.

This thesis proposes a Goal-driven Software Development Risk Management Model (GSRM) that explicitly integrates into the requirements engineering phase. The integration provides an early warning of potential problems so that both preventive and corrective actions can be undertaken to avoid the causes of project failure. The framework is comprised of four layers, i.e., goal, obstacle, assessment and treatment, that support the identification, assessment, treatment and documentation of risks in relation to project-specific goals. GSRM is implemented in active on-going software development projects to empirically evaluate its usefulness, particular advantages and limitations in an industrial context. The results show that goal-driven approach is suitable for risk management and risk management is well integrated into requirements engineering phase. It is not always necessary to rank budget and schedule related goals and risk factors at the highest priority for risk management. At the early stage of the project risk factors related to estimation, project management, project scope, requirements, change management and human (i.e. customer/user and practitioner) and at the later stage risk factors related to user satisfaction and product usage are more frequent and severely affect meeting the project goals. If project risk factors are beyond the control of a project manager and project development environment, it is difficult to control the risks. The results conclude that early risk management practice is necessary and GSRM contributes to this direction for a successful project outcome.

Zusammenfassung

Jedes Software-Entwicklungsprojekt ist einzigartig und geprägt von schwer planbaren Einflussfaktoren, wie time-to-market oder Budget, aber auch von Einflüssen resultierend aus der Integration und der Wartung. Die Beherrschung dieser Einflussgrößen ist unabdingbar für die Minimierung der Risiken während der Entwicklung als auch während des gesamtem Software-Lebenszyklus. Software Risikomanagement stellt ein effektives Mittel zur Risikobeherrschung dar. Häufig ist ein umfassendes Risikomanagement jedoch aufgrund fehlender Ressourcen oder fehlendem Domänenwissen nicht realisierbar. Idealerweise muss Risikomanagement aber in den gesamten Entwicklungsprozess integriert sein, insbesondere auch in die ersten Phasen der Enwicklung.

Der Beitrag der Dissertation ist ein Modellierungsframework zum Risikomanagement. Wir schlagen einen zielbasierten Ansatz vor (Engl: Goal-driven Software Development Risk Management Model (GSRM)) und integrieren diesen in die erste Phase des Entwicklungprozess, in das Requirements Engineering. Diese Integration trägt dazu bei, frühzeitig potentielle Probleme zu erkennen und diese in Form von korrigierenden Maßnahmen zu umgehen. Das Framework gliedert sich in vier Abstraktionsebenen, i.e., goal,obstacle, assessment and treatment, die eine methodische Handlungsrichtlinie zur Identifikation, Dokumentation und Behandlung von Risiken in Zusammenhang mit Zielen unterstützt. Die Integration der Ziele führt insbesondere zu nachvollziehbaren und reproduzierbaren Spezifikationsdokumenten.

Die Tragfähigkeit des Ansatzes wird in Fallstudien unter Einbeziehung laufender Software-Entwicklungsprojekte hinsichtlich Anwendbarkeit und seiner Vor- und Nachteile evaluiert. Die Ergebnisse werden sowohl die Integrationsfähigkeit des Ansatzes in das Requirements Engineering, als auch seine Anwendbarkeit aufzeigen. Wir werden Beobachtungen darlegen, dass das Risikomanagement unmittelbar durch die fehlende Einbindung des Projektmanagers in Projektinhalte erschwert wird, und dass der Projektkontext, wie auch die Projektkomplexität die Risiken gleichermaßen beeinflussen, unabhängig von der ursprünglichen Risikoeinschätzung des Projektes.

Wir schließen die Arbeit mit einer Zusammenfassung der Ergebnisse, stellen die Relevanz eines frühzeitigen Risikomanagements dar und belegen, wie wir mit dem Beitrag dieser Dissertation dieses Risikomanagement unterstützen.

Acknowledgments

It is very difficult to manage a funded PhD research from Bangladesh specifically in the field of software engineering. I am very grateful to the scholarship agency DAAD(German Academic Exchange Service) for giving me an opportunity to pursue my PhD study in a country like Germany. I would like to thank Prof. Manfred Broy for giving me the opportunity to work in a challenging and competitive research environment and support me in all aspects during the course of the dissertation. His support and critical comments made this work much better. I also want to thank Prof. Dr. Martin Bichler for co-supervising this thesis and providing me time to discuss the updated status of the thesis.

I would like to thank Siv Hilde Houmb, Telenor GBD& R, & SecureNOK Ltd., Norway for her continuous support throughout the work on my dissertation. She helped me all the way to complete this dissertation from her long experienced as a risk expert in the industry.

I would also like to thank Axel van Lamsweerde and Robert Darimont for their valuable suggestion to work with KAOS goal modeling language for risk management and permit me to use KAOS goal modeling tools Objectiver version 2.

During my research work, I had collaborations with Jan Jürjens from Technische Universität Dortmund & Fraunhofer-Institute for Software- and Systems-Engineering (ISST) , Haris Mouratidis from University of East London, Kurt Schneider and Eric Knauss from Leibniz Universität Hannover and Stefan Wagner and Daniel Mendez-Fernandez from Technische Universität München. And with whom I published several research papers in the direction of this dissertation. Jan Jürjens also gave me useful insights about how to do research and how to write papers. I would like to thank all of them.

There are several persons that read (parts of) my dissertation and provided me very useful comments: Alarico Campetelli, Dmitriy Golubitskiy, Daniel Ratiu Maximilian Irlbeck , Philipp Neubeck from TUM Germany, Siv Hilde Houmb from SecureNOK Ltd. Norway. Thanks all of you.

My stay at the chair was made much easier by: Silke Müller, Eleni Nikolaou-Weiss, Marina Franke, Philipp Neubeck who solved my problems all the time when I appeared in front of them. Thanks all of you.

Above all, I want to thank my parents, brother, especially my wife and our little son Maheer to support me while I was struggling to do my research and for giving me the greatest joy of my life.

At the end, I would also like to thank all the faculty members and stuff of Institute of Information Technology(IIT), University of Dhaka, Bangladesh for a wonderful time and every support before starting my PhD work.

Contents

CHAPTER 1

Introduction

Contents

1.1 Background and Motivation

Every software development project has unique demands, both in terms of project specific features and mostly tacit customer/user expectations. Software project commonly contains a significant amount of uncertainties, such as continuous change of project scope to accommodate the latest market demand, requirements evolution [KS04], imprecise estimation of budget and schedule [FcY04] and high system complexity. These factors raise the chance of potential risks during the life cycle of the product. Therefore, software engineering is more prone to risks compared to many other engineering domains. These risks possibly lead to delays, economic loss, customer dissatisfaction, market refuse and as a whole reduce the ability to a successful project completion. But software development and its roles in our society have increased substantially over the past decades. Software becomes more important while organizations incorporate the Internet and endeavour to operate both the physical and cyber world of the market space [TTL00]. The growth rate of worldwide software cost is 12% per year [Nuk99] and the importance of software is likely to be continuously increased. But, global IT software industry stands to lose billions of dollar annually through project failures or projects that are not delivered according to the specification [Cha05]. The literature is packed with such horror stories [Gla98]. Uncertainties and their consequences in software projects should constantly be under control. Software risk management can be used as a tool to manage these risks and to reason under the high degree of uncertainties involved.

Risk management in software development needs to be included as early as possible during the development ,in particular, within the Requirements Engineering (RE) phase. The reason for considering risk management within requirements engineering phase is that requirement problems are one of the main causes of project failure [Gla98]. Problems related to requirement completeness, correctness, stability and meeting project goals and objectives are numerous, frequent and continual. These problems are the most expensive software problems that persist throughout the life cycle of software product [vL09]. The causes of these problems are generated from various perspectives including users, practitioners, project execution or knowledge. Risk management in requirements engineering contributes to identify and analyze factors related to requirements and other problems from the early stage. Boehm [Boe81] reported that the cost to fix requirements defects is 5 times higher during design, 10 times higher during implementation, 20 times higher during unit testing and up to 200 times higher after the system delivery. Requirements engineering lays the foundation for successful projects development regarding cost and quality [Bro06, BFI$^+$09]. The precise elaboration of requirements supports both subsequent software development processes like architectural design and project management processes like the definition of contractual agreements [HWFP07].

Risk management in early stage of the software projects is critical and contributes effectively to increase the likelihood of project success. A project without risk management admits serious problems such as reworks of artifacts and overruns of cost and schedule only after the risks appear during the development. The organizations continue to invest time and resources in strategically important software projects. Therefore managing software development risks that are associated with organizational key areas turns into a critical area of concern. Thus we are motivated for an effective risk management practice which must be *systematic*, *incur low overhead to the development* and *be explicitly included in early stage of software project*.

1.2 Problem Domain

The complexity of software has increased significantly over the years [Bro95]. Despite the advancements in technology and development process, software projects still face similar problems repeatedly. Poor project outcomes persist even though numerous studies scrutinize and document the causes of project failure. Study results showed that the causes of most of these failures have little to do with technical issues even though project managers have a common tendency to concern more on the technical things. The results also concluded that failed projects suffer from the poor management of people related problems rather than technical problems [McC96, Lin99]. McManus [McM04] identified that 65% of the project failures are accounted by management issues and 35% by technical issues. Humans involved in every part of the software development activities incur errors, make wrong assumptions and show poor team performance which pose potential risk [ID08]. Paulk [MCCW93] stated that end user involvement is one of the most important contributors to successful project development. The current practice of software risk management concerns more on the goals related to schedule, cost and quality. Nevertheless, certain goals have gained importance recently such as outsourcing and co-ordination of the global software development, supporting critical business processes, time-to-market, produce deployment and assurance of safety, security and privacy issues. Software development risk management needs to focus on these goals depending on the specific project context.

At present risk management in software domain is still not a mature discipline. Several researches [Boe91, Kar95, Kon01, Roy04] contributed to the development of a single integrated framework for performing software risk management activities. The problem is not that developers and project managers are not aware about the importance of risk management and its positive impact to the project's outcome, but that risk management is not effectively applied in practice [Rop99, Pfl00]. Risks are intangible by nature and address issues related not only to present but also to future. Risk estimation is a difficult undertaking where factors are mostly fuzzy. This inherent nature of risks makes it difficult to measure the concrete risk management benefit even if the project succeeds [Ban08]. Visible development costs always get more attention in the project and lack of resources to perform risk management are the most potential barriers for risk management [OG09]. Furthermore, project-specific risks are less obvious and more difficult to predict their impact. Risk perception varies by the project stakeholders in particular among the management, customer/user, and practitioners based on the roles, responsibilities and expectations from the project. One can easily overlook or misjudge the risks [Sch08, ID08]. The project practitioners are not always aware about the methods and techniques required to perform the activities related to the software development risk management.

There is still a lack of comprehensive guidelines on how to integrate risk management activities at the early development stage. A survey study result on several companies concluded that risk management and its integration into development process is poorly done [NKM08]. If risk management activities are merely employed from the design phase on, the result may include expensive revision of the design artifacts or major rework of the elicited requirements. This may also pose additional problems later on depending on the competence and ability of developers for late discovered risks, mistakes of the requirements or inconsistent design and may also end with passive customer/user involvement. Some recent works have tackled the problem of considering risks in the early phase of system development [AG07, BFH08, vL09]. However the focus is more on the requirement related problems rather than the holistic software development risk perspective. For instance, Lamsweerde [vL09] introduces risk management as a part of requirement evaluation and the main focus is to ensure completeness of the requirements.

1.2.1 Overall Goals of the Thesis

The main thesis goals are: *to develop a systematic and easy to use effective risk management practice which can be explicitly integrated in requirements engineering phase.*

1.2.2 Research Question

We have articulated the research questions for this dissertation based on our findings from the literature survey and current needs in order to improve the risk management practice in software projects.

- **RQ.1:** Can risk management be integrated in the early phase of software development projects?
- **RQ.2:** What is the effect of the goal-driven approach on the software risk management?
- **RQ.3:** What are the main goals of early development contributing to a successful project outcome?
- **RQ.4:** How can the software development risks be assessed and managed from a holistic perspective to satisfy the goals?

1.3 Research Contribution

The novelty of this dissertation is the improvement of the risk management practice explicitly within the early development stage. Thus the contributions of this research towards the research questions are summarized as:

1. **Goal-driven risk management model from holistic perspective:** This research contributes an integrated modeling framework for software development risk management, called *Goal-driven Software Development Risk Management Model(GSRM)* [IJH09, Isl09]. The framework is comprised of a conceptual model, analysis techniques and a methodology to systematically identify, model, analyze and control risks to attain the project specific goals. GSRM includes goals as the objectives, expectations and constraints from the development components by focusing on the project success factors for the risk management. The model considers goals beyond schedule, budget and quality and realizes the importance of motivating project stakeholders in particular customer/user to take an active part in the software project. It focuses on the non-technical components such as project execution, customers/users, project participants and usage environment, along with the technical components such as development process, system specification and tools as a holistic view for the risk management. Goal-driven approach for risk management aids to identify both generic and project specific risks, understands their characteristics and provides support for effective control actions to mitigate the risks.
2. **Systematic risk assessment and management during requirements engineering:** The proposed model supports an effective, systematic and straight forward way to manage software risks. We define the activities, tasks, steps and underlying roles for a detailed goal-driven risk management process model [IH10, IH11]. We follow an artifact oriented view by considering the content and associated concept of the individual artifact as work product of the risk management activities. Artifact oriented view emphasizes on the results rather than dictating a strict process. It ensures consistency and completeness among domain-specific results, independent of taken decisions and chosen methods. Thus artifact orientation, i.e., concepts and attributes and process model, i.e., activities, tasks and roles are the integration points that allow to incorporate the risk management activities in the requirements engineering phase. The early integration supports to assess and manage risks related to the project execution, development environment, operational constraints and project stakeholders even before elicitation of the system requirements. Goal-oriented Requirement Engineering (GORE) [vL09, BPG+04] refers to the use of goals, has long been recognized in requirements engineering community to elicit, evaluate, negotiate, elaborate, structure and document requirements. GSRM is a goal-driven approach and extends the KAOS goal modeling techniques to accommodate the risk management activities.
3. **Goal-risk taxonomy:** To effectively identify and categorize goals and risk factors, this research characterizes the software development components into five different dimensions. They are: project execution, process, product, human, and environment (internal and external). Every component is further described with elements and factors which are relevant for the component. This *component-element-factor* hierarchy supports to identify and categorize goals and risk factors. We develop a goal-risk taxonomy by following the basic concept of this research and the results of the empirical investigation. The taxonomy includes general characteristics of the development components which need early attention and a set of goals and risk factors. A total

of 200 closed questions (i.e., as shown in appendix A) and a requirements errors checklist (i.e., as shown in appendix C) are also developed, as a part of the taxonomy, to identify the risk factors.

1.4 The Approach

We present a risk management model to effectively address the risks that obstruct the successful project outcomes. The approach explicitly models the relations between the goals based on the software development components and project success indicators with the risk factors that obstruct these goals. Risks are then assessed and suitable control actions are selected to mitigate the risks so that the project can attain its goals. Therefore, in our approach, it is important to model the relationship among goals, obstacles and treatment. The reason for choosing goal modeling language is that goals and risks are complementary entities of a software project. A risk is usually defined as negation to a single or multiple goals or a loss of attainment of some corresponding objectives. Risks always shadow the goals and certain goals may be risky. The goal-driven approach anchors the risk management process. It allows to trace and rationalize the risk factors, events and control actions to the goals. Figure 1.1 depicts the conceptual view of the proposed approach. The goal-risk model and associated artifacts play the main role of this approach. Goals are derived from the development components, project stakeholders' expectation and project success indicators. These goals are obstructed by the software risks. Risks are assessed and suitable treatment actions are identified to attain the goals. We define the goals and software risks as used in our proposed approach.

Definition:*Goals*

The Goals are the objectives, expectations and constraints of the stakeholder and development components, as prescriptive statements of intents in development, operation and maintenance of software project whose satisfaction contributes to the overall project success.

Definition:*Software Risks*

The Software Risks are obstacles caused by the risk factors and associated events whose combined consequences raise the chance of single or multiple undesirable circumstances that obstruct the goals and certainly reduce the likelihood of the project success.

A successful software development project relies on many factors. The affect of these factors varies from project to project. Our approach identifies goals from these factors. Risks are concerned with the negative consequence that directly or indirectly obstruct these goals. These key definitions are necessary to communicate the scope and contribution of this work. For instance, we consider software development risks beyond their traditional definition by incorporating links to the project specific goals and not only budget, schedule and quality. Furthermore, the *goal-risk model* and *causal relationship model*, as shown in Figure 1.1 are the two main models of the proposed approach. A goal-risk model refines the higher level goal to sub-goals and includes obstruction links from the risks to the relevant single or multiple goals. A causal relationship model establishes the link from risk factors to events and their overall consequence to the goals negation. We consider the

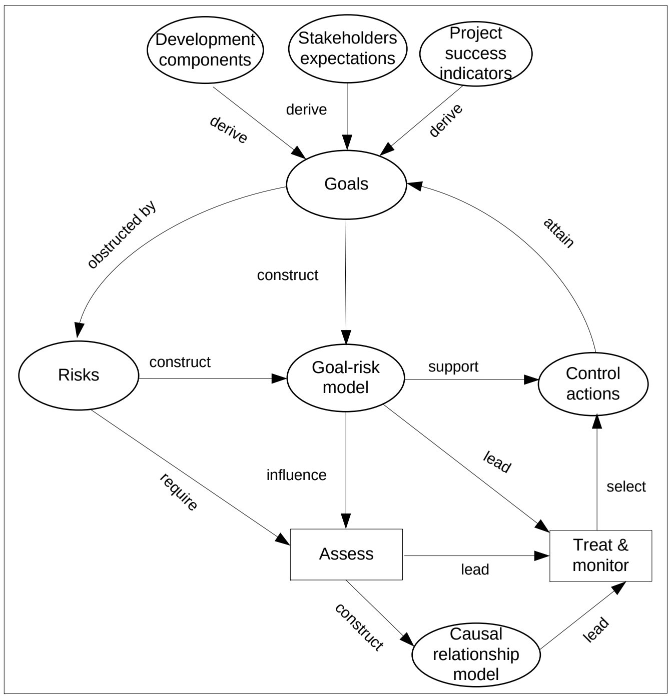


Figure 1.1: Conceptual view of the proposed approach

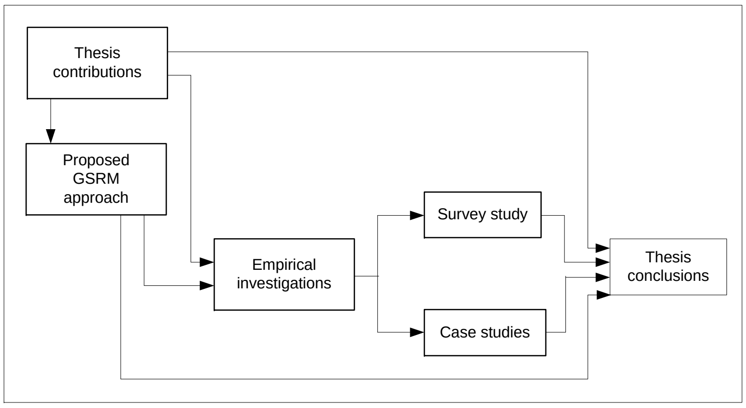


Figure 1.2: Empirical Investigation

integration points from artifacts and process oriented perspectives of the requirements engineering and risk management. This eases to explicitly integrate the risk management activities into early requirements engineering phase.

1.5 Empirical Evaluation

We have chosen empirical study to evaluate the proposed approach specifically the main contribution of this research. We follow survey and case study method for this purpose and employ action research so that risk management results can effectively be contributed to mitigate the risks that exist in the project. The studies focus to assess the usefulness of GSRM. Survey studies results were presented in the literature to identify the risk factors and their impact on the software projects. However only few studies focused on risk management impact in software development projects. It is also difficult to validate a single method separately in an active on-going software development project. Furthermore, subjective nature of the risk, evolution of technology, limited resources and constant pressure to meet budget and schedule constraints make the task more challenging. Therefore limited data points exist which consider the usefulness and limitations of risk management in software development projects. Our survey study focuses to identify the goals and risk factors. And the case studies combine action research, where GSRM is implemented into on-going software development projects. Therefore goals and risk factors are assessed from the running project context and implemented control actions are used to mitigate the identified risks. We report the detailed study design, result, study validity and lessons learned. Figure 1.2 shows empirical evaluation by considering the thesis contribution, GSRM and the contribution to the conclusion. Our result concluded, that goal-driven approach is effective for risk management and risk management is well integrated into requirements engineering phase. We believe that the empirical study results contribute to summarize the risk management impact on the software projects.

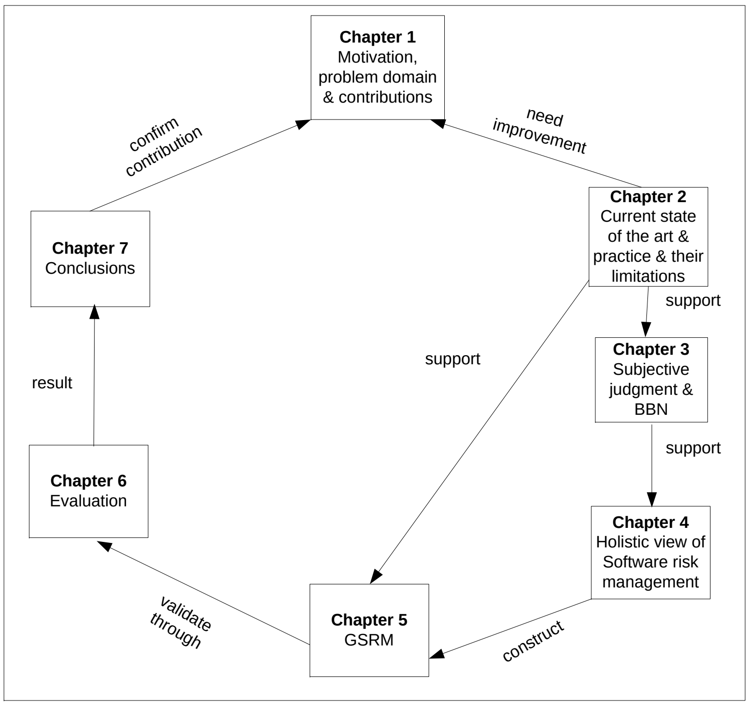


Figure 1.3: Over view of the Thesis Structure

1.6 Structure of the Thesis

This first chapter provides the motivation of this research work by investigating the existing literature of software risk management and problems associated to the current state of risk management practice. The chapter also includes the main contribution of this research. Figure 1.3 shows the relationship among the thesis chapters. The figure depicts a closed loop of the whole thesis contribution. It starts with claims as research contributions and finally confirms the claims at the end of the thesis. Besides this chapter, the thesis is organized into the following chapters:

Chapter 2: presents the literature survey on the state-of-the-art and state-of-the-practice related to software risk management. The chapter starts with the basic concepts of software risk management and outlines the existing approaches and techniques for the software risk management. It then continues with the current practices and study results on risk management. A short introduction of Goal-oriented Requirement Engineering (GORE) is given. Finally the chapter summarizes the existing state of the art and outlines the main thesis contribution.

Chapter 3 provides a brief overview of subject expert judgement and Bayesian Belief Network(BBN). Both of these methods are employed by this research for the risk estimation and causal relationship model.

Chapter 4: describes the foundational concept of the goal-driven risk management model from a holistic perspective. It specifies the early development components and associated elements and factors within the component. This component-element-factor hierarchy allows to identify and categorize the goals and risk factors during the software development. It also supports the risk management from a holistic perspective.

Chapter 5: introduces the Goal-driven Software Development Risk Management Model(GSRM), as the main contribution of this research. The first part of the chapter explains in details the basic concepts of the modeling framework w.r.t. four different layers (i.e. goal, obstacle, assessment and treatment) of GSRM and its meta-model. Then it outlines the process model in particular the underlying activities and tasks involved in GSRM. The chapter is concluded with the main integration principles of GSRM into requirements engineering.

Chapter 6: evaluates the proposed software development risk management model through empirical study. The first report is based on the survey focused to identify the goals and risk factors from the experienced software practitioner within offshore outsourced software development environment. The case studies implements the GSRM into active on-going software development projects. Finally the chapter includes a goal-risk taxonomy for software project.

Chapter 7: presents the conclusions on the overall thesis contribution. The research results are summarized and evaluated. In addition, this section includes a discussion of the strengths and weaknesses of GSRM as well as some insights into ongoing and future work.

Appendix A: comprises 200 closed questions to identify the risk factors of a software project.
Appendix B: provides the open questions to evaluate the GSRM.
Appendix C: provides a checklist to identify requirements errors.

Figure 1.4 illustrates at a glance our contributions (i.e., left side of Figure 1.4) and its distribution to the structure of this dissertation (i.e., right side of Figure 1.4).

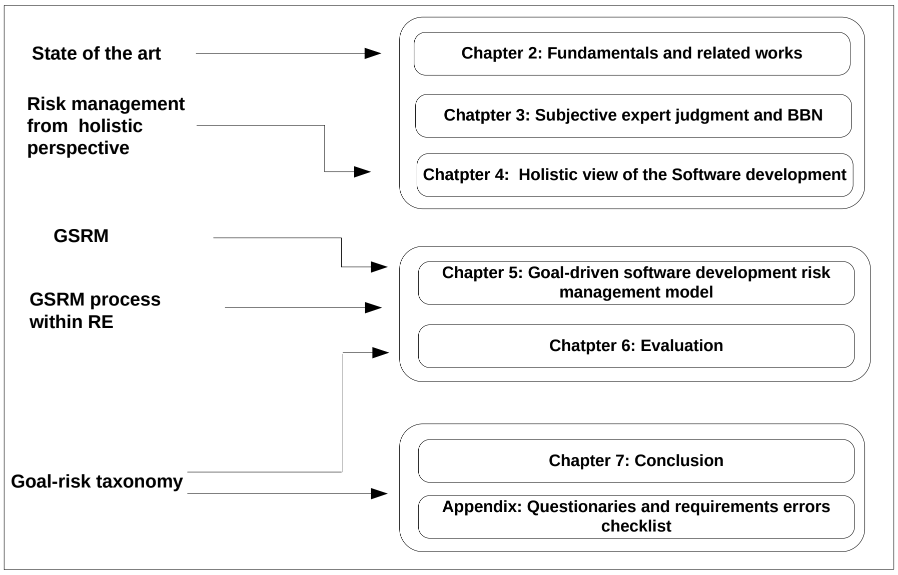

State of the art
Risk management from holistic perspective
GSRM
GSRM process within RE
Goal-risk taxonomy
Chapter 2: Fundamentals and related works
Chatpter 3: Subjective expert judgment and BBN
Chatpter 4: Holistic view of the Software development
Chapter 5: Goal-driven software development risk management model
Chatpter 6: Evaluation
Chapter 7: Conclusion
Appendix: Questionaries and requirements errors checklist

Figure 1.4: Contributions Distribution over the Chapters

CHAPTER 2

Fundamentals and Related Work

Contents

In this chapter, we provide background information and related work which are relevant for this research. It includes definitions of some basic terms and concepts of software risk management used by the research and industry communities. The commonly used software risk management frameworks are presented in details. The chapter also includes several study results about risk factors and their influence on software development project. It also includes goals related to project success and some findings about the obstacles to implement an effective risk management practice. Goal oriented requirements engineering approaches are presented afterward. Finally we present the summary of the extensive literature survey and the main contributions of this research work. Figure 2.1 shows the summary of our extensive investigation on state of the art on software risk management. It includes risk management frameworks and survey study results on risk factors, risk factor impact, risk management barriers and factors related to project success.

2.1 Basic Concepts

This section defines the basic concepts of software risk management terminology used in research and industry. In our everyday use, *risk* refers to an element or effect that has some potentially damaging consequences. However, in engineering work domain, risk associated with a project is a measure of the possible costs associated with a single or multiple undesirable events and considers both likelihood of event as well as severity of damage done if the identified event occurs and would occur. On the other hand, in decision theory, risk is treated as an outcome of either positive or negative consequences, reflects the variation in distribution

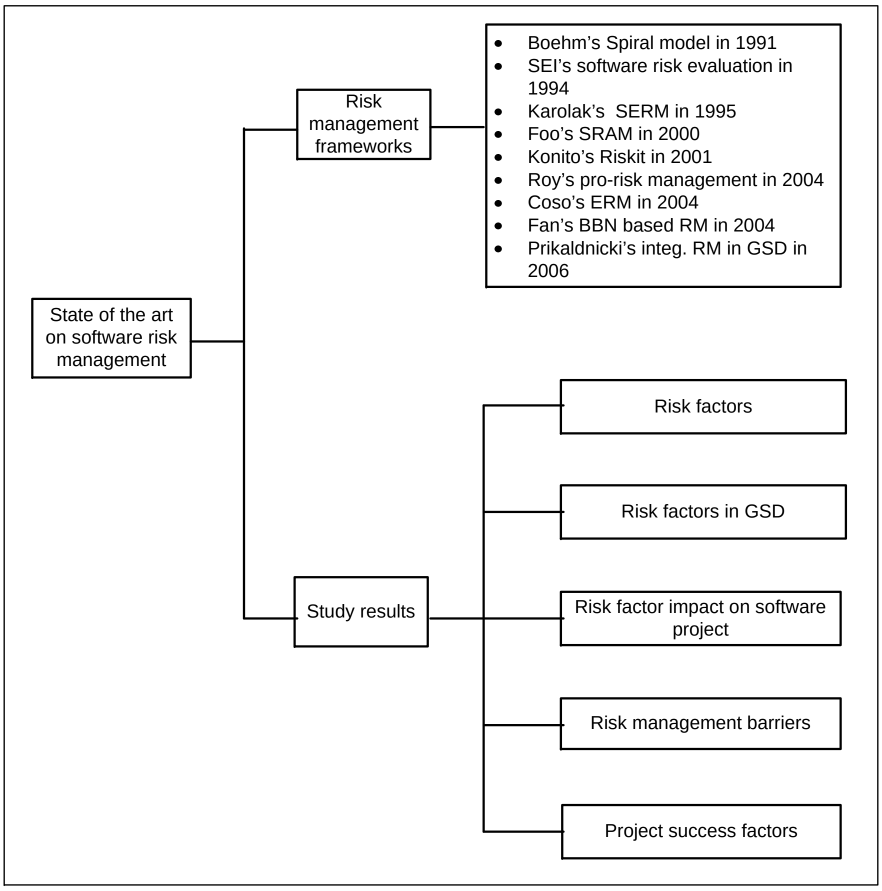

State of the art on software risk management
Risk management frameworks
Boehm's Spiral model in 1991
SEI's software risk evaluation in 1994
Karolak's SERM in 1995
Foo's SRAM in 2000
Konito's Riskit in 2001
Roy's pro-risk management in 2004
Coso's ERM in 2004
Fan's BBN based RM in 2004
Prikaldnicki's integ. RM in GSD in 2006
Study results
Risk factors
Risk factors in GSD
Risk factor impact on software project
Risk management barriers
Project success factors

Figure 2.1: State of the art on Software Risk Management

of possible outcomes [Arr57]. However empirical study results showed that the perception of risk used in decision theory is not consistent with the term risk defined by the industry managers [MS87]. They are quite insensitive to estimate the likelihood of risk outcomes.

2.1.1 Software Risk

Risk, in ISO Guide 73:2002, is defined as, *"combination of the probability of an event and its consequence"* [Iso02]. It is constituted upon three basic concepts: *event*, *likelihood*, and *severity*. However, the main focus is on undesirable events which pose a loss in a specific context. Software risk, is defined as, the possibilities of suffering a loss such as budget or schedule over-runs, customer dissatisfaction, poor quality and passive customer involvement due to an undesirable event and its consequences during the life cycle of the project.

2.1.2 Risk Event and Likelihood

Risk event is the occurrence of a particular set of negative circumstances or discovery of information that depicts negative circumstances. The event can be certain or uncertain and can be influenced by a single occurrence or a series of occurrences.

Likelihood refers to the notion to which an event is probable to occur. Therefore an event is modeled through likelihood of uncertainty by several mathematical theories such as probability theory [Ros97], expected utility theory [Hog87], Dempster-Shaffer theory of evidence [Sha76] and fuzzy set [Zad99]. These theories are developed for different purposes and represent different classes of uncertainties. For instance, the axioms of probability theory are [Fis67]:

- **Axiom I:** Every random event A, there corresponds a certain number P(A), called the probability of A, which satisfies the inequality $0 \leq P(A) \leq 1$
- **Axiom II:** The probability of a certain event equals one, i.e., P(E) = 1
- **(**Axiom III:) The probability of the possible alternatives of a finite or denumerable number of pair wise exclusive events equals to the sum of probabilities of these events.

Hagan et al. [OBD$^+$06] argued that uncertainty could be divided into two classes, i.e., aleatory uncertainty due to randomness and epistemic uncertainty due to incompleteness of knowledge. However uncertainty may also appear even though one has complete knowledge in a situation such as *rolling a dice* despite of knowing possible outcomes. This follows aleatory uncertainty by randomly chosen the outcome by rolling a dice. Smithson [Smi89] also argues that uncertainty could also emanate from imprecision due to vagueness. These three categories of uncertainty aleatory, epistemic and imprecision, in particular epistemic and imprecision, frequently exist in software projects and pose for a potential risk. For instance, practitioners' lack of knowledge about the customer business domain is a very common and frequent problem in the software projects. This incomplete knowledge causes an epistemic uncertainty within the development environment. Requirement engineer imprecisely elicits the requirements, as a result customer business needs are unclear and vague. Product is not precisely deploy in the user environment. There exists a cause for the occurrence of risk event which is known as risk factor. Risk factors and events are commonly categorized under a specific classification.

Risk Factor Risk factor is a cause or characteristic that typically influences the possibility of a risk event occurrence. For instance, "passive participation of customer/user in requirements elicitation" as risk factor causes the risk event "incomplete requirements". Thus risk factor is always related with the risk event and depend on specific context. However depending on the context, a risk factor can be also treated as a risk event.

Risk Category Risk category is a grouping or a class of risk. In this dissertation, we focus on software development components to category the risk. These categories are from project execution, product, process, human and environment.

2.2 Risk Management in Software Project

Risk management in software project describes an integrated engineering approach with methods, processes and artifacts for identifying, analyzing, controlling and continuously monitoring risks in order to reduce the chance of project failure. It is a practical way to manage risks in the project. Integration of risk management into software development activities is critical and requires initiating as early as possible. Early initiation of risk management allows a proactive decision making situation to control the threats before creating problems to the software development project. Therefore risk management in software project requires undertaking decision making activities. Several works exist in the literature in the direction of risk management frameworks. However the initial contribution for software risk management is done by Boehm at late 1980's [Boe91]. After that, there are many contributions have produced well-documented framework for risk management such as SEI's software risk management approach [CKM$^+$93, ADH$^+$96], Karolak [Kar95], konito [Kon01], Roy [Roy04] and Prikaldnicki specifically for the global software development [RPJLNA06].

2.2.1 Principals of Software Risk Management

Among the several contributions in software risk management domain, there is however a consensus that the risk management commonly comprises of two general phases including risks assessment and control. The risk assessment phase involves the following activities, generally performed in a sequence:

- **Risk identification -** This activity provides a list of risk items which threat to the software project. Risk identification describes detailed about the risk of software project including risk factors, risk category and the risk itself. Various approaches can be used to identify risks. These approaches may include questionnaires, taxonomies, brainstorming, scenario analysis and analyzing project documents.
- **Risk analysis -** This activity estimates the probability of the risk event occurrence and loss magnitude by the risk event. Therefore risk analysis transforms the raw risk information into a decision enabling knowledge by evaluating a risk exposure level within a chosen scale. Estimation can be quantitative or qualitative. Depending on the context, it is necessary to define which risks will be evaluated using a qualitative or quantitative scale. The scale should be used in a consistent way.
- **Risk prioritization -** Finally the identified and analyzed risks are ranked by their relative weight. Software projects generally contain a large number of risk factors. Therefore it is not always possible to mitigate every factor. Initial consideration can be on the high prioritized risks. Risk exposure and leverage would be effectively eased to rank the risks.

The risk control phase includes the following activities:

- **Risk management plan -** This activity supports to address each risk item or prioritized risk items based on suitable cost-effective potential control strategies. Various treatment alternatives to address risk should be considered to reduce or eliminate risks. Main focus is to proactively minimize the risks before it occurs (mitigation planning) or to reactively recover it from the loss and restore the normal process state (contingency planning). Plan should focus whether risks are acceptable to the stakeholders, if not, then what are the possible control actions to reduce the risks.
- **Implementation of risk control actions-** The selected potential control action need to be implemented to control the risk at an acceptable level.
- **Monitoring -** The activity tracks effectiveness of the implemented control actions as well as project progress toward resolving the risks. All risks in particular high priority risks should be monitored at a certain interval so that, if required, corrective actions should be undertaken for a specific risk. Risk monitoring also seeks out new risk and their sources throughout the development life cycle.
- **Risk resource repository -** The activity builds a repository based on the artifacts including risk factors, risk events, rank of risk, control action and status

produced from the risk management steps under a specific project context. This facilitates to reuse risk based knowledge for software projects.

2.2.2 Risk Management Frameworks

The theoretical foundation of putting risk management into a single framework is initially contributed by Boehm [Boe91]. Boehm's risk-driven Spiral model was the first life cycle model to integrate risk management throughout software development life cycle in an iterative manner. Later on, Boehm extended the original Spiral model using the theory W (Win-Win) model [BEK+98] to satisfy the objectives and concerns of the stakeholders. The model also supports risk identification, resolution and continuous monitoring of risks. However the approach requires intensive active involvement of project customer/user, which is difficult to attain in real on-going project situation.

The Software Engineering Institute (SEI) provides a comprehensive framework to support a continuous software risk management by software risk evaluation [SJ94]. The approach concerns identification, analysis, communication and mitigation strategies for software risk management. It depends on risk taxonomy and consists of constructs used for organizing risk information. Therefore risk taxonomy and associated questionnaire [CKM+93] is the central element of SEI's approach. The taxonomy covers 194 questions of several areas including requirements, designing, coding, testing, integration, and engineering specialties under product engineering, development process, development system, management process, management methods, work environment under development environment and resources, contracts and program interfaces under program constraints. The SEI Continuous Risk Management Guidebook [ADH+96] provides practical techniques for providing processes, methods and tools for continuously managing risks during all phases of software development life cycle. However, the guidebook does not include theoretical background information about risk management as well as key limitations and biases associated with the techniques it presents. Team risk management concerns the development of methods, processes and tools for building relationships among project stakeholders during software development. These three different groups support each other for the successfully completion of risk management.

Karolak proposed Software Engineering Risk Model (SERIM) [Kar95] framework by taking Just-In-Time (JIT) software approach. SERIM attempts to minimize the amount of risks involved, while optimizing the contingency strategies for problematic situations. The approach considers three main risk elements, i.e., technology, cost and schedule from technological and business perspectives. These elements are interconnected with 81 risk factors. The risk factors are influenced from organization, estimation, monitoring, development methodology, tools, risk culture and usability. Every risk factor is associated with a specific risk metric and question. These questionnaires, such as a checklist to identify specific risk are answered by project representatives and converted to numerical values through metrics. The network of answers and their weights are used to calculate risk factor values, using probability tree. Karolak's model gives quantified estimates of project's risks along the risk factor categories. However, no empirical validation report is available on the SERIM approach as well as it does not show when and where the risk management initiates during the development and who will be involved into the process.

Kontio proposed the Riskit methodology [Kon01], which provides a complete conceptual framework for risk management using a goal/expectation approach from the stakeholders and risks which threaten the goals. The approach provides precise and unambiguous definitions of risks and aims at modeling and documenting risks qualitatively. At the heart of the approach, is the visual formalism of the risk by analyzing risk factors, risk events, risk reactions, risk effect sets and utility loss that would occur due to a risk events. Risks are ranked in terms of probability and utility loss by a specific Riskit Pareto ranking technique. Riskit also helps for systematically managing the project starting from identification and analysis of risks to the monitoring and control of them. The approach is finally validated through empirical evaluation by several studies including exploratory case study at NASA, characterizing case study at Hughes, Nokia and DaimlerChrysler study and method introduction study with IESE and Tenvois [JKL98, FHK$^+$01]. The results showed that Riskit method is practical, adds value to the project and its key concepts are understood and usable in practice. For instance in Tenvois case study, explicit process of the Riskit method is simple and understandable and cost of Riskit is 5% of the overall project management effort. Therefore the impact of risk management on the project was low and acceptable. However there are some limitations of the Riskit. There are no clear sources specified from where the goals are originated and how the identified goals are modeled. Risks are analyzed and prioritized by deriving scenarios which is a non-trivial task when a scenario depends upon more than one probabilistic element. Furthermore, developing a scenario is a time consuming task in hectic real software development project. There may be risk factors that are not necessarily directly obstructed a particular goal, expectation and constraint but that pose other problems within the development environment. This may also reduce the project success rate such as domain knowledge of the practitioners within the developing project. And it is always hard to formulate a scenario from this factors and attempt to perform a comparison among them. The approach prioritises risk scenarios rather than individual risks. However Riskit certainly contributes by introducing the goal concept for risk management.

Foo et al. [FM00] make use of comprehensive questionnaire to construct Software Risk Assessment Model(SRAM). A set of questions are chosen for nine critical risk elements, i.e., complexity, staff, targeted reliability, requirements, method of estimation, monitoring, process, usability and tools. Each question contains three possible answers and each element has different degree of impacts on different types of software projects. Therefore when the elements are combined, different weights are assigned to derive the overall risk value. However, determining risk element probability is a difficult task, which depends on project specific context and involved practitioner's belief. There is no set rule to use a common weight value. Further detailed implementation of the model is missing.

Roy's pro risk management framework [Roy04] is an extension of the AS/NZS [AS499]. The main attention of the framework is on business component in which the project is created and operational domain where the project is actually carried out. Business domain considers economic environment in which the project is being undertaken and organization knowledge and experience for the project. Operational domain considers risk assessment and implementation of control action within by following organizational views and policies. A risk tree model is constructed through risk factors and these risk factors are categorized into cluster. The model is calibrated mainly by cost and schedule perspectives but if required quality and reputation perspectives are also considered.

Enterprise risk management (ERM) framework proposed by [COS04] affect in every layer of the enterprise. The framework is applied in strategy setting and across the enterprise and is designed to identify potential events that may affect the enterprise. ERM initially focuses on achievement of objectives established by a particular entity and provides a basis for defining enterprise risk management effectiveness. The objectives are four categories which allow focusing on separated aspect of enterprise risk management. They are: Strategic - high-level goals, aligned with and supporting its mission, Operations - effective and efficient use of its resources, Reporting - reliability of reporting, and Compliance - compliance with applicable laws and regulations. ERM is composed of eight interrelated components, i.e., internal environment, objective setting, event identification, risk assessment, risk response, control activities, information and communication, and monitoring. Therefore ERM covers all steps for risk management however precise definition of each step is missing in particular how individual step is performed. More over it is considered as a separate enterprise module.

Prikaldnicki et al. in [RPJLNA06] proposed an integrated risk management process across three different organizational levels i.e., strategic, tactical and operational, in particular for the global software development (GSD) projects. It follows a reference model which focuses on two different dimensions, i.e., organizational and project and consists of four different phases including new project, project allocation, project development, evaluation and feedback [PNJE06]. Each phase contains individual process. Once the project is identified, risk assessment effectively assists offshore development decision from both the strategic and tactical level. Risk assessment continues further at the operational level and this integrates the result from the previous levels. This facilitates to communicate risk and decision, initiating from strategic and tactical levels, continuing to operational level. In the case study, several risk factors in particular from legal compliance, privacy, development infrastructure, security, requirements clarity, technology, project complexity and time frame are considered.

2.2.3 Risk Management Standards

IEEE std 16085-2006 [IEE06], system and software engineering- life cycle processes - risk management, specifies detailed guideline to perform risk management during system life cycle. The standard includes six main activities: plan and implement risk management, manage the project risk profile, perform risk analysis, monitoring and treatment and evaluate the risk management process to systematically address risk throughout the life cycle of the product. The activity should be performed at the beginning of the project and repeated when information needs to be changed. Furthermore it also outlines several artifacts including risk management plan, risk action request, risk treatment plan to manage well-documented risk management activities. Thus the standard complements the activities supported by the other literature in a systematic way.

There also exist other standards which address risks from different perspectives. ISO 27001:2005, Information technology - Security techniques - Information security management systems(ISMS) -Requirements [ISOc] emphasizes the importance of implementing and operating controls to manage an organization's information security risks in the context of the organization's overall business risks. The standard guides to identify, analyze, evaluate and treat risks to establish an information security management system. It also emphasizes monitoring and reviewing the performance and effectiveness of the control actions. The main focus is to protect assets that have a business impact upon organization and successfully security failures which can negatively impact on the confidentiality, integrity or availability of the assets. ISMS mainly adopts Plan-Do-Check-Control model for managing security risk.

However, available standards are too general and any detailed guidelines about how to perform the activities in risk management are not available. Therefore, one has to formulate his/her own process for performing risk management by following the standard. We can validate the standard by following it in an ongoing software project.

2.2.4 Current Practice of Risk Assessment

Risk identification in a software project relies upon two main techniques: checklists and brainstorming sessions with project documents. A brief overview of the techniques is given below

2.2.4.1 Checklists

Checklist usually comprises a set of questionnaires or a risk list based on the experience from past projects. Questionnaires generally determine the current state of the project through open or closed question and attempt to identify factors which pose risk in the development. The risk list composes of typical factors which pose potential risks in the software project. Both questionnaires and risk lists consider several areas of development such as project cost and schedule, practitioners, customer/user. For instance, SEI risk taxonomy by Carr et al. [CKM$^+$93] was organized into three major classes including product engineering, development environment and program constraints. These classes are further divided into elements and each element is characterized by its attributes. The complete taxonomy is a hierarchical structure of super and sub-types with 194 mostly open question described from the software development risk perspective. But the answers may not be concrete which make hard to quantify the risks value. More over, it is not clear when during the development the question response would be collected, who will be responsible to answer the questions and which artifacts can be reviewed for this purpose. Another risk identification questionnaire has been compiled by McConnell [McC96] covering mostly the coding issues, followed by a list of typical schedule risk factors.

The main benefit of the checklist approach for risk identification is that it provides a quick, easy to use and controlled way to identify and assess the risk exposure against the major factors. However there are several problems with this simple approach. The published risk lists are generic and may be not relevant to a particular project. Some are long checklists with unlimited scope through life cycle which makes it difficult to follow it in a hectic running software project. As stated, the perception of risk varies over time, over project life cycle and different cultures, between stakeholders. Therefore checklists may bias to particular state or may be limited in scope.

2.2.4.2 Brainstorming Session

Brainstorming session is a collaborative technique which depends on high human involvement for risk identification. It is a face-to-face meeting among project team members including practitioners, management representatives and customer/user representatives to share their experience to identify reasons of the risk of the running project. Konito studied the effectiveness of group work under brainstorming sessions for risk identification. The most beneficial feature of a brainstorming session is the interaction among several project stakeholders. Rather than that, there are several other advantages of using brainstorming session such as short duration, prompt effects, keeping project participants concerned about risk and improving interaction among project members. However this technique also has some limitations including low scope control, obtaining project stakeholder in a real situation is not always possible, heterogeneous output and the fact which result always depends upon the expertise of the participants.

Besides checklists and brainstorming sessions, there are some other techniques such as decision-driver analysis, scenario analysis and prototyping to identify the risk factors. However every technique in particular checklists and brainstorming sessions have distinguished features as well as limitations. Therefore combination of the techniques is more suitable for the risk identification.

2.2.4.3 Risk Analysis

Risk analysis determines the effects of the potential risks, categorizes as well as prioritizes the raw identified risks. It is the most challenging part of software risk management practice due to the difficulties of estimating the risk event likelihood and its impact. Both qualitative and quantitative based assessment is applicable to estimate the risk. Qualitative approach is simple and requires immediate actions to the prioritized areas for the improvement. But the approach does not provide specific quantifiable measurement of risk elements therefore supports limited risk estimation accuracy. On the other hand, quantitative approach provides potential support for decision making by quantified estimates of impact, frequency, effectiveness, and cost. However it is difficult to perform a comparison to a qualitative one but sometimes produces unclear results which require extra effort to interpret them in qualitative manner. Software project risk is most often estimated qualitatively for the sake of simplicity. The scale values of risk are typically ranked with three to five levels: catastrophic, high, moderate, medium, and low and probability can also be scaled as rare, unlikely, moderate, likely and almost certain.

However risk estimation, in particular obtaining specific risk probability value, is a difficult task. There are several sources to estimate the probability such as use of historical data, theoretical analysis and subjective value. Historical data is seldom available in software engineering and rarely provide trustworthy base to estimate the probability. This is because due to the course of time, project goal, situation, business process and technology have continuously changed, risks that are likely to occur in past do not guarantee to have similar probability in present. In software engineering, there is a lack of theoretical analysis in literature and practice to support risk analysis. Therefore, subjective judgment is commonly used to estimate the risk event likelihood and associated consequence. However sometimes, in an on-going project, there are often too many potential risk factors to analysis and manage. Therefore controlling individual risk factors would be unproductive due to causal ambiguity. Chapter 3 provides a brief overview of subjective probability in software risk management.

2.3 Study results on software risk management

There exists several survey studies on identifying risk factors and assessing its impact on the software development project. The participants of these surveys are mostly experienced practitioners who expresses their view about the risk factors and their impact in the software development. Some studies also focus on the challenges for implementation of risk management in software development projects.

2.3.1 Risk Factors

The well known *"top-ten"* list of risk factor is provided by Boehm [Boe91]. After that several lists of risk factors have been published [BRT93, KCLS98, SLKC01, WKR04, AMN07]. Among these contributions, Barki and Schmidt composed the most comprehensive ones. Barki et al. [BRT93] compiled a list of 35 risk variables which were represented in the form of a questionnaire consisting of 144 items based on the review of the existing Information System (IS) literature. The results of their survey based on the questionnaire showed five influential factors: technological newness, application size, lack of expertise, application complexity and organizational environment for the most interpretable solution of software risk. Initially nine factors are considered but judged as uninterpretable and proposed these five factors as an interpretable solution. Based on the result 23 uncertainty variables were retained related to uncertainty and measured by 83 items. However the measurement scales were complex and large number of single item measure puts the whole measurement process into an unmanageable state. Moynihan [Moy97] focuses on the project constructs, i.e., personal constructs and application, needed to be considered by the project manager. The study observed that risk variables identified by Barki et al. [BRT93] lack client's apparent knowledge. There are also many areas addressed by the SEI taxonomy [CKM$^+$93] i.e., product technical aspect, management methods and process which are not directly reflected by the personal construct. A project manager should focus on all issues related to personal constructs since early stage of project. Schmidt et al. [SLKC01] published a comprehensive list of risk factors by conducting Delphi survey study to the experienced project practitioners through three different panels located in three different countries. The study categorized the risk factors into several different areas such as corporate environment, sponsorship, relationship management, project management, scope, requirements, funding, scheduling, development process, staffing, technology, and external dependencies. Arshad et al. [AMN07] published a list of important risk factors and ranked them by following the survey response.

2.3.2 Risk Factors in Global Software Development

Some research specifically considered to identify the risk factors in Global Software Development(GSD) context. Iacovou et al. [IN08] summarized a risk profile by the top ten risk factors for offshore-outsourced development projects. The risk factors are ranked as very important, important and less important by focusing on three main areas: communication, client's internal management and vendor capabilities. Nakatsu et al. [NI09] further investigated and compared the risk factors between offshore and domestic outsourcing. The result showed that many risk factors related to project management commonly appeared on the top of both domestic and offshore risk lists such as lack of top management commitment, inadequate user involvement and failure to manage end user expectation. Some are unique for the offshore context such as lack of communication, poor change controls, lack of business know-how and failure to consider all costs. Aspray et al. [AMV06] provided an ACM task force report that considers risk from both technical and nontechnical issues. Tsuji et al. [TSY$^+$07] proposed an questionnaires assessment scheme considering software, vendor, and project properties to quantify risk of offshore software outsourcing. Their survey result showed that the degree of importance among these properties and concluded that vendor properties such as communication and project management ability mainly affect the result of development whereas software properties such as requirement volatility did not affect the result. Some studies also outlined difficulties due to development process, project management and project complexity for the GSD projects [Car99].

2.3.3 Risk factors impact on software project

Some works have been undertaken to investigate the potential effects of risk factors on software development projects. Wallace et al. [WKR04] focused on risk factors from six different dimensions and their affect on a project. The study considered low, medium and high risk projects and observed the impact of risk dimensions on the projects. The six risk dimensions are: user, team, requirements, planning and control, complexity and organizational environment. The result showed that risks associated with requirements, planning and control and organizational environment are more obvious for the high risk projects. Project scope affects all dimensions of risk and project complexity is more obvious for the low risk project. Keil et al. [KCLS98] identify the risk factors and their relative importance in software project. The result showed that very important risks are most often not direct controlled by the project manager. Risk factors are organized into four quadrants based on the perceived importance of risk and perceived level of control by the project manager likely to have. Quadrant 1, *customer mandate* focuses on risk factors related to customer and user. Quadrant 2, *scope and requirements* focuses on risk factors related to success criteria, requirements, project scope, and gaols. Quadrant 3, *execution* focuses on risk factors related to staffing, project constraints, roles and development methods. And quadrant 4, *environment* focuses on risk factors related to internal and external environment. Wallace et al. [WK04] further investigated the relative importance of these quadrants and outlined that execution risk including project team, complexity, planning and control, management is twice as important as scope and requirement risk for the process outcome. And customer mandate, scope and requirements, and execution risk have significant affect on product outcome.

Ropponen et al. [RL00] empirically identified six components of software development risk i.e., scheduling and timing, system functionality, subcontracting, requirement management, resource usage and performance and personnel management. Several influential factors are considered for the individual components. The result showed that awareness of the risk management impact and systematic risk management practice has an effect on schedule, requirements and personal management risk. Software risk management is affected by several environmental factors such as target platform selection, development process, leveraging on experience, experienced and well educated practitioners, and proper project scoring. The study also recommended, that organization must tailor risk management efforts to the development environment.

Jiang et al. [JK00] examined the risk impact on different system development aspects in particular the study looked at the 12 most common software development risks proposed by Barki [BRT93]. Empirical evidence on the relationship between risk and project effectiveness is considered here. Two common risks observed are: lack of expertise and clear role definition. Practitioner expertise is further refined with ability to work with uncertain objectives, top management and as a team, and ability to carry out tasks effectively. However project management guidelines allow the skill matching and team skills should be conducted throughout the development. On the other hand, member selection for the project must comprise clear definition of specific tasks, expected outcomes, rewards, timing, responsibilities, and reporting relationship. The result concluded that, controlling these two risk dimensions can effectively contribute to the project success.

2.3.4 Software Risk Management Barriers

Ropponen et al. showed that [Rop99] 75 % of the project managers did not follow any detailed risk management practice and did not have adequate knowledge about software risk management. However for a successful project, it is difficult to prove that any part of the resulting product was influenced by the software risk management [Ban08]. Kwak et al [KS04] observed that project managers and practitioners perceive risk management activities as extra work and expenses. It may inhibit creativity work during the development. Nyfjord et al. in a survey study outline [NKM08] several problems to integrate risk management into the development. They are: resource problem (i.e., expensive training cost, lack of resource and time), organizational problems (i.e., different roles, lack of competence, overloaded work), scope problem, and process problem (i.e., lack of coordination, integration and planning). Another recent survey study of experienced project managers [OG09] on their perception of software risk management outlines tangible development cost, lack of resources and efforts from organization or process change due to risk mitigation are top the three identified barriers of software risk management. The result also emphasizes on other barriers such as too much risk to control, reward for problem solving not for risk mitigation, overconfidence on individual measurements which pose challenges for the implementation of risk management in the development. Risk management is the most problematic task of software risk management. Pfleeger [Pfl00] emphasizes on false precision (i.e., lack of risk probability distribution data and obscure value), questionable science (i.e., relevancy and quality of the data being studied, ignorance or giving less credence to qualitative data), and confusion of facts with value (i.e., risk quantification without knowing the consequence).

Despite of these barriers, risk management is one of the most effective tools which effectively contributes to increase the likelihood of project success. Therefore integration of risk management in software project effectively contribute to prevent the risks and contributes for quality product with estimated cost and expected quality. Literature [BP01, FCD02] emphasizes the importance of educating risk management process and techniques to the practitioners involved in software development.

2.3.5 Study Result on Software Project Success factors

To demonstrate a software development risk, it is necessary to understand the issues related to the project success. Project success factors have great importance to software development project effectiveness and are obstructed by the software development risk. Therefore success specifies what should be done for the project. Nevertheless it is highly dependent on the context and difficult to describe in absolute terms. Study results published factors related to the project success.

Procaccino et al. [PVOD02] identified seven major factors which contribute to the success or failure of a software system. They are: management, customer and users, requirements, estimation and scheduling, the project manager, the software development process and development personnel. This study investigates some of the most influential success factors early in the development process from the perspective of developer. Developers perceived that presence of a committed sponsor and customer/user confidence on the project manager and developers are the influential factors to the project success. However this perception differs significantly compared to the management perception about success because management takes politically oriented view to manage the customer and user. Change of project scope during development does not change the perception of project success. The result also shows that neither project manager's full authority to manage the project nor customer/user involvement in schedule estimation highly influence the project success. Procaccino et al. [PV09] in another qualitative assessment report the end user expectations towards a successful project from the developers perspective. The most frequently mentioned responses related to process are : communication between user and team in particular importance of keeping the customer informed, user involvement in the development process specifically user feelings as a part of decision making and defining specific and realistic requirements. The most frequently mentioned responses related to final system/outcome are: requirements/ functionality/ performance, ease of use and learning and system completion which delivers functional needs and within budget and time.

Linberg noted that software project success is narrowly defined due to the deep dependency on cost. Jiang et al. [JKD02] used metrics to measure system success from providers and users perspectives. The multidimensional measures on system success allows to improve the link between IS professional and those who use the results of their efforts. In literature, user satisfaction is considered to be the most widely used measure for the system success. But perception of user satisfaction on service delivery and performance are significantly differed between users and practitioner. For instance If users have higher expectations than IS professionals, a problem may develop in the reception of the final product. On the contrary, if professionals have higher expectations, then there may have ineffective resource utilization in particular too much effort may spend for the less important issues. Knowledge gap between the groups is another critical problem [JK00]. Users commonly evaluate final product based on the specifications. But there exist external issues such as appropriate level of technology for the product or adequate user knowledge to handle the production in particular when the product is innovative or unique and this knowledge gap of technology is difficult to control and highly affect to the usefulness of the product. User education may be the best control for this gap. Wohlin et al [WvMHR00] use subjective measurement factor based on project characteristics and success indicators to evaluate project success. There exists positive correlation between the project variable, i.e., problem complexity, requirements stability, staff turnover, time pressure, information flow, top management priority and project management with the success variables. Results showed that requirements stability and priority secure the two highest correlations. Khan et al. [KNA09] by a systematic literature survey noted that cost saving even though the highest prioritized success factor should not be considered as the prime one for the offshore software development. Vendors should also focus on competence of human resources, appropriate infrastructure, and quality of product to be very important success factor in offshore context. Prikaldnicki focused on managing human factors as critical success factors for distributed development team [Pri09]. The approach used the PDI model to quantify the perceived distance which is not necessary physical distance but is a subject of feelings among the distributed development teams.

2.4 Requirements Engineering and Goal-Orientation

Goal-Oriented Requirements Engineering (GORE) refers to the use of goals has long been recognized by the RE community in recent years for elicitation, evaluation, negotiation, elaboration, structuring, analysis, and documentation of requirements. GORE addresses the problems associated with business goals, plans and processes as well as systems to be developed or to be evolved in order to achieve organizational objectives [LK95]. Goal model generally shows the system's functional and non-functional goals contribute to each other through refinement links down to software requirements and environmental assumption. Requirements are the lower level goals under the responsibility of a single agent of the system to be [vL09]. Goals play a crucial role and identify all phases in particular during requirement elicitation, evaluation and consolidation in the RE process. They provide the rationale for requirements and a foundation for showing the alignment of the system-to-be with the organization's strategy objective. Several methodologies in particular i*/Tropos and KAOS are commonly used for GORE in RE process. Here we present a short overview of i*/Tropos and a detailed of KAOS since it is the main foundation for this work.

2.4.1 i*/Tropos

i*/Tropos is a requirement engineering methodology, which provides a description of work organization in terms of dependency relationships among actors [Yu97, YLL01, BPG$^+$04]. The approach pays a great deal of attention to the early requirements, emphasizing the need to understand not only what organizational goals are, but also how and why the intended system would meet its organizational goals. In particular, Tropos mainly adopts the i* modeling framework and focuses to build a system model which is incrementally refined and extended from a conceptual level to executable artifacts, by means of a sequence of transformational steps. Several concepts such as , actor, goal, task, resource and social relationships are used to capture stakeholders' intentions in an organizational setting. An actor, within the multi agent system, is an active entity that carries out actions by using some resource to achieve intentionality and strategic goals within the multi-agent system or within its organizational setting. These goals can be hard or soft. A hard goal represents a condition or actor's strategic interest in the world that an actor would like to achieve. A soft-goal is used to capture non-functional requirements of the system, and unlike a (hard) goal, it does not have clear criteria for deciding whether it is satisfied or not and therefore it is subject to interpretation. A resource represents a physical or informational entity which is generally required by an actor to perform some action. The main concern when dealing with resources is whether the resource is available and who is responsible for its delivery. Once the concepts are defined then an actor model can be constructed by establishing a dependency link among the actors, performed tasks, required resource, and related goals. This goal-oriented approach explicitly links business needs and objectives to the system functional or non-functional components.

Several works consider an extensions of Tropos methodology in particular for security as well as other non-functional domains. Among them, Secure Tropos presented a successful extension of the Tropos methodology. Therefore agent oriented software engineering paradigm presents a feasible approach for the integration of security into software engineering. Security requirements are mainly obtained by analyzing the attitude of the organization towards security and after studying the security policy of the organization. Liu et al. [LYM03] propose a methodological framework for security requirements analysis founded on i* and the Non-Functional Requirement NFR) framework. In particular, their analysis explores alternative designs and evaluates them on the basis of threats, vulnerabilities and countermeasures. Secure Tropos [MG07, IMJ10]introduces security related concepts (e.g. security constraint, secure dependency, secure goal) to the Tropos methodology, to enable developers to consider security issues throughout the development life cycle. A security constraint is defined in Secure Tropos as a restriction related to security issues, such as privacy or integrity, which influences the analysis and design of the software system under development by restricting some alternative design solutions, by conflicting with some of the requirements of the system, or by refining some of the system's objectives. These constraints represent in Secure Tropos the initial high level security requirements which elicited from a number of sources including the stakeholders and users of the system as well as domain and security experts. In actor model, Secure Tropos introduces secure dependencies where an actor must be fulfilled the constraints to attain the goals. Secure Tropos uses the term secure entity to describe any goals, tasks and resources related to the security of the system.

2.4.2 KAOS

KAOS (Keep All Objective Satisfied) is a goal-oriented requirements engineering methodology, which aims to model not only what and how aspect of requirements but also why, who, and when [DvLF93, vL09]. A goal in KAOS, is a prescriptive or descriptive statement of intent that the system should satisfy through the cooperation of its agents. Goals can be behavioral which specify maximal set of admissible system behaviors either required to achieve or maintain/avoid depending on the nature of the goal. E.g., car behaviors when driver drives the car. Goals can also be soft which prescribe choices among alternative system behaviors, may require to improve, increase, reduce, maximize, and minimize. E.g., user interaction, cost, development, and some performance goals tend to be soft goal. A goal can also be categorized as a functional and non-functional goal. A functional goal specifies the intent basis of the system service, whereas a non-functional goal states the quality or constraints on a service provision or development. Therefore functional goals can be satisfaction, information and stimulus-response goals. And non-functional goals can be quality of service in particular safety, security, reliability, performance, interface, and accuracy, compliance, architectural, development goals. However goal partitioning through behavioral or soft is completely different from functional and non-functional ones. Because the former one deals with goal satisfaction in clear cut sense and the later one on classification criterion. Soft goal is more fuzzy that deals mainly with intended system services or quality constraints of such services. In KAOS goal model, each of these goals is represented graphically with possible prefix by its type, is annotated by a number of features to characterize the goal individually. The core of the goal model consists of a refinement graph of higher-level goals to lower-level goals and conversely lower-level goals contributed to higher level ones as well as represents the potential conflicts among the goals.

The initial model has been extended, by defining obstacles and constraints that can be seen as boundaries in requirements analysis [vLL00]. A goal is obstructed by an obstacle if the satisfaction of the obstacle prevents the goal from being satisfied. Goal obstruction yields a sufficient condition for a goal not being satisfied and the obstacle must be compatible with known valid properties of the domain related to the goal. An obstacle can be of several types, i.e., hazard obstacles which obstruct the safety goals, threat obstacles which obstruct the security goals, dissatisfaction obstacles which obstruct the satisfaction of agent requests, misinformation obstacles which obstructs the goals of making agents informed, inaccuracy obstacles which obstruct the consistency between state of variables controlled by software agents, and usability obstacles which obstruct usability goals. Obstacles are also modeled and refined from a higher to lower level based on its category to specify which single or several goals a specific obstacle obstructs. In KAOS, both goals and obstacles can be refined through AND-refinements or OR-refinements. However, if an obstacle O directly obstructs a goal G, then OR-refinement sub-obstacle of the parent O also directly obstruct the goal G due to the transitivity relation. Additionally, there is a dual structure between goal's AND-refinement and obstacle's OR-refinement. For instance consider a goal G is AND-refined in two sub-goals G1 and G2. An obstacle O that directly obstructs the goal G can be OR-refined into O1 and O2 that also directly obstruct the sub-goals G1 and G2. This is the direct consequence of Demorgan's law of propositional logic not(G1 and G2) is logically equivalent to not G1 and not G2

This representation allows assessing the severity of the obstacle when one single obstacle obstructs single or more than one goal. Therefore, obstacle analysis supports for a more robust goal model by following risk management process through obstacle identification, assessment and resolution. Higher level goals yield higher level obstacles that need to be refined significantly so that likelihood of an obstacle can be assessed easily and accurately. This refinement should support the duplication of goal refinement by goal AND-refinement and obstacle OR-refinement. Obstacle identification process should be fairly systematic based on the goal category and priority and may require domain experts to obtain the task. All refined obstacles should verify its consistency with the relevant domain and estimate the likelihood of the refined sub-obstacle as well as the related parent obstacle and their severity. Finally, the identified obstacles, if likely and critical, must be resolved through suitable control actions, in general at RE time.

2.4.3 Summary of the Current Approaches

This section summarizes our investigation on the state of the art in practice of software risk management by following the existing risk management frameworks and study results:

- We noticed that most of the existing frameworks follow more or less the same process to asses and manage risks. Practitioners and researchers emphasize on the initiation of risk management activities as early as possible. However details to integrate risk management activities into software project is still missing. Some works consider risk management during requirements engineering [Bor05, BFH08] and software design [VM04]. But in case of considering risk management activities into software design, counter measures may introduce revision of the whole design or alteration of the elicited system requirements and related artifacts. These may lead to unanticipated problems during the development and jeopardy to the project success. Therefore considering risk management since the early phase in particular within requirements engineering phase can avoid such problems and contributes to mitigate these risks. On the other hand, approaches which consider risk management into requirements engineering mainly focus on the elicited requirements and late requirements analysis. This implies that risks are analyzed only from the technical perspective. Recent work from Asnar et at. [AG06, AG07] explicitly consider risk management activities into early requirements engineering by considering stakeholder needs within the organizational setting. They extend i* / Tropos and SI and include a goal-risk model. However, actual integration points are missing regarding the integration of risk management into requirements engineering. Hence there is a great need for process integration in particular criteria such as activities, resources and roles to understand the integration.
- Software risk management generally focuses on limited goals related to schedule, cost and quality. Nevertheless, certain goals such as coordination projects work within different cultures and locations, supporting critical business processes, compliance with the demanded regulations, product

maintenance, safety, safety and privacy have gained importance recently. Indeed, software projects should not be restricted with limited goals. Additionally, supporting limited goal may ignore the important risk or may rank the risk lower priority.

- There are many study results on risk factors as well as different software risks checklists available in the literature. But relatively little effort has gone into assessing the impact of these risk factors on the software development projects. In particular little work has been undertaken to understand the impact of overall risk management framework into software development. Furthermore selecting and using a specific checklist is a difficult task as projects also suffer from project specific risks. Checklist only supports to identify generic risk. To identify the project specific risk factors, we need to understand the project domain context, goals, local environment and other related factors. We also need to study the results of risk management impact on the software development project. It gives empirical data points about the benefits of the risk management and motivates the practitioners for an effective use of software risk management.

2.5 Thesis Contributions

This thesis work contributes to overcome the existing limitations for an effective risk management practice and its integration in software projects. For our work, we have decided to extend the KAOS methodology to integrate the risk management activities. This research introduces goal-driven approach for a comprehensive risk management framework. The thesis mainly contributes:

1. A goal-driven modeling framework to support more effective, systematic and straight-forward methods and techniques for software risk management. Risk management cannot solely depends on technical aspects. It needs to focus on the organizational, practitioners, user and business perspective as well as user organizational environment where the software would operate and maintenance. The proposed model considers a holistic view that spans toward both technical and non-technical dimensions of the development components. The modeling framework is organized into four different layers: goal, obstacle, assessment and treatment. At the top, goal layer, it identifies and elaborates the goals of the project stakeholders and other development components such as human, organization, product and process. Obstacle layer considers the risk factors which obstruct the goals and elaborate through assessment layer. Finally treatment layer selects the potential control actions to fulfill the goals. Goal-driven approach systematically integrates project generic and specific risks to assess and manage throughout the product life cycle.
2. The model explicitly integrates into early requirements engineering phase. We consider integration points from the process and artifact perspective of both requirements engineering and risk management to understand the necessary criteria for the integration. This integration eases to consider issues related to a practitioner, customer/user, project context, development environment even before elicitating the system requirements.
3. A systematic risk management methodology with sequence of activities to identify goals and obstacles and to model them to support risk assessment and treatment. The methodology adapts the foundations of risk management practice from existing literature in particular Boehm's Spiral model, Karolak's SERIM and the Systems and software engineering - life cycle processes

- risk management standard ISO/IEC 16085: 2006 as well as basic concepts of KAOS's goals and obstacles. The methodology supports modeling of goals and obstacles i.e., goal-risk model and causal relationship from the risk factors to the risk events and consequences. The process, techniques and roles are clearly specified for an effective risk management practice. Furthermore, we also define the artifacts produced by the activities.

4. A questionnaire has been developed to identify the risk factors within the software development projects. It consists of a comprehensive number of close questions which are arranged by following the software development components and uses as a taxonomy to identify the causes of anticipate and unanticipate problems in software projects.

CHAPTER 3

Subjective Expert Judgement and Bayesian Belief Network

Contents

Accurate estimation of project risks is always important for an effective risk management practice. However in software engineering domain precise estimation of risks is difficult, in particular, when risk event likelihood is measured through probability. Software risk event likelihood estimation contains much fuzziness and uncertainty. But the estimation should be simple, reasonable and practical to implement, otherwise it would not be formally followed in software projects. One way to improve the situation is to integrate an experienced practitioner or expert judgement to estimate the risk event likelihood. Subjective expert judgement is an alternative and effective way for the software risk estimation. We integrate expert judgement with Bayesian Belief Network(BBN) for the risk estimation. This section provides an overview of the subjective expert judgement and BBN.

3.1 Subjective Expert Judgement

Probability can be purely subjective when a true direct value does not exist, i.e., it has not yet or cannot be observed directly. In such a case the value considers as degrees of confidence or credence to partial beliefs of expert within a specific domain. The main focus is to put on observable values that are not known at the time of the assessment but can be observed some time in the future. The future observable values are expressed as the subjective estimate of what the expert thinks will be the outcome of a specific context. These values are from relevant experience, knowledge and recommendations provided by one or more external sources that the expert trusts. A subjective measure depends only on the knowledge and expertise of the people involved. The potential advantage is that it is fairly easy to collect the information through interviews, questionnaires or brainstorming sessions. Such a judgement commonly does not require an extensive measurement program. However the measurement may be less precise which makes it hard to draw any conclusions. But when objective measurement value is not available or limited data points are available to measure any context then subjective expert judgement is a useful technique.

Subjective judgement is applied in limited areas of software engineering domain such as cost and effort and risk estimation. A few studies attempted at a better understanding of the expert estimation process within software project. For instance, Connolly et al. [CD97] showed that the expert estimation process turns out to be more effective when risk analysis is included. Gray et al. [GMS99] observed a close relationship between software task characteristics such as the size and module with expert judgement relevant for software effort estimation. Wohlin et al. [WvMHR00] considered subjective factors from the project characteristics and project success indicators to evaluate the software project success. They considered correlation analysis, principal component analysis, ranking, classification and agreement index to study the variables of several software projects. Ropponen et al. [RL00] also considered subjective judgement to estimate risks in software projects. Similar to the existing approaches, we rely on expert or practitioners belief to estimate the risks. An Expert is someone who has knowledge about either the application area or specific domain such as risk management, security or safety. Studies from other domain provide some common finding on subjective judgement. The results show that experts perform better than models in highly predictable environment but worst in a less predictable environment [SS96]. Experts systematically have too high confidence in their judgement, if proper learning was not achieved [EH78], expert when comfortable with his/her current values or beliefs is reluctant to change opinions when faced with new knowledge. He/she may always be biased on own judgement [Sch08].

We describe the uncertainties involved in software development projects by a subjective probability distribution which maps the expert opinion. The Triangular distribution is the most commonly used distribution for modeling expert opinions [Vos00]. This is defined by three different values, i.e., i) its minimum value, ii) most likely value and iii) maximum value. For instance, likelihood of schedule overruns can be estimated into three different scales. Ideally, probability distribution should be available to the experts when expressing their belief. However, experience has shown that simple probability distributions are more tractable and even more importantly understandable for the experts. The latter is important as it has been observed that poor performance by experts is often not due to lack of expertise but rather to the lack of training in subjective probability assessment [GHKM00]. The main structure of such procedures is usually a variant of the following three steps: (1) preparation for elicitation, which includes identifying and selecting experts and describes the area of interest, (2) elicitation, which is the actual collection of expert judgements and (3) post-elicitation, which includes among other things the aggregation and analysis of expert opinions. There are three factors, i.e., scrutability, neutrality and fairness, that must be taken care of during elicitation of expert judgement. Scrutability or accountability is of great importance which concerned with the generalization of the result. Therefore results must be reproducible by competent reviewers. Neutrality is also important. This means that all elicitation and evaluation methods must be constructed such that they encourage experts to state their true opinions. Underestimation and overestimation should be taken into account in relation to neutrality. Underestimation means that the expert has little confidence in his or her experience and knowledge and consequently gives pessimistic information. Overestimation is the opposite and means that the expert has great confidence in his or her expertise and knowledge and consequently provides optimistic information. And fairness considers the overall judgement quality which support both scrutability and neutrality.

3.2 Bayesian Belief Network (BBN)

Bayesian Belief Network (BBN) [Jen96], based on the Bayes theorem, has already proven to be an effective technique for reasoning under uncertainty. BBN is successfully applied in several disciplines in particular medical and safety domain. Some recent work employs BBN for the software risk management and software quality [FN09, Wag09]. BBN is a mathematical model to depict interrelation of several events by defining the conditional probability between events. However BBN is not a model that is specifically built for risk modeling [Pea01].

BBN [Jen96] is represented as a directed acyclic graph (DAG) together with an associated set of probability tables. The graph consists of two portions: nodes representing the variables involved and arcs representing the causal/relevance dependencies between these variables. Nodes are of various types, i.e., parent or observable, target and intermediate nodes and are denoted as stochastic or decision variables where multiple variables are often used to determine the state of a node. Each state of individual node is expressed using probability density functions(pdf). Probability density specifies the confidence in the various outcomes of a set of variables connected to a node and depends conditionally on the status of the parent nodes at the incoming edges. For instance, user passive involvement and practitioner lack of domain knowledge increase the likelihood of requirements errors. User's passive involvement and lack of knowledge are considered as root causes and are treated as parent nodes. These parent nodes, as shown in Figure 3.1, influence the likelihood of requirements errors which is the target node. Therefore, requirements error causally depends on two parent nodes and may further influence the poor quality of product. These nodes values can be obtained by observing the elicited requirements and practitioners expertise level in the project. However statisticians argue that the probability value should be obtained from long-observation of random process. But in software engineering risk management domain, it is difficult to obtain long observations of data points due to uniqueness of project, technology evolution and lack of historical data. Furthermore, assigning a precise node value is also difficult in software risk management. The problematic part of modeling uncertainty is that the outcome of an event is dependent on many factors and on the outcome of other events. Hence, there are complex dependency relationships that need to be accounted for. This is handled by the underlying computation model of BBN, which is based on Bayes rule.

The main advantage of BBN is that it visually represents the causal relationship among the factors related to uncertainty. The relationship can be serial which refers to the situation where belief propagates through the nodes to increase confidence of the occurrence of a certain event. Or can be diverging which distributes a certain belief through several nodes and finally converging a connection, which combines certain beliefs to increase the confidence for the occurrence of an event. The BBN method is applied whenever using BBN as a decision making tool. The application of the BBN method consists of four main tasks, i.e., i) identify the interesting variables, ii) construction of the BBN topology, iii) elicitation of probabilities to nodes and edges and iv) making computations. Each of this steps is important and non-trivial. Errors in each of these steps can have a large effect on the outcome. The first task includes the assumption that the model builder can decide on some basis what is important for a specific domain. Construction of the BBN topology is done by examining the variables involved and their relationships and by modeling these explicitly as nodes and arches in hierarchy of DAGs. Elicita-

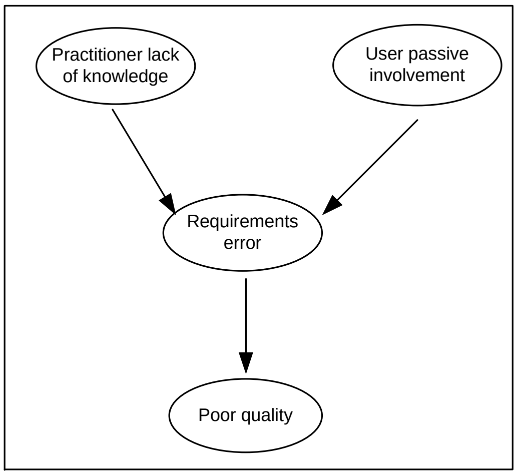


Figure 3.1: Bayesian Belief Network Example

tion of probabilities to nodes and edges involves the collection and aggregation of evidence/information, such as elicitation of an expert opinions as described in the previous section. The probabilistic relationship between the nodes in a DAG is described using node probability tables (NPT) where variables can be independent or dependent. Making computation involves propagating information and evidence entered into the observable nodes of the BBN topology. These pieces of information are propagated to the target node of the topology through the intermediate nodes. Propagation is done by updating the conditional probabilities on the edges of the topology taking the new evidence into consideration. There exist evidence propagation algorithms, such as Bayesian evidence propagation [Jen96] and the HUGIN propagation algorithm [Lit].

The key feature of BBNs is that it allows to model and reason about uncertainty. BBN forces the assessor to expose all assumptions about the impact of different forms of evidence and hence provides a visible and auditable dependability or safety argument. We need to provide the evidence of a software risk event occurrence based on the factors as causes responsible for the event. Some of the factors are serial, where one factor X influences another factor Y and then Y further influences Z. Otherwise there can be diverging connection, factor X influences Y and Z and these factors are more important for the software risk. Factors can also be converging such as several evidences influence to one single consequence, i.e., shown in Figure 3.2. To develop causal relationship among the risk factors and their influence into software risk management, some contributors focus on the Bayesian Belief Network to estimate the risk probability. Fan et al. [FcY04] use BBNs to support decision making of software project risk management. The approach combines BBNs in an algorithmic way to identify, predict and estimate risks in a probabilistic fashion. Risk management mainly considers ensuring the project activities in a cost-effective way by dealing dynamic risk management, where activities are adjusted dynamically and varied costs are the major concern. The risk management process should be continuous and BBNs are updated in each iteration with new project data to generate new estimations. The tasks for the process by following the IEEE standard 1540 are initialization, risk profile, risk analysis and monitoring and risk treatment. The risk analysis step considers the risks beyond the schedule, budget and quality. Saturation module is used to adjust the resources utilized for a specific activity in development. Hui et al. [AL04] contribute with a mathematical

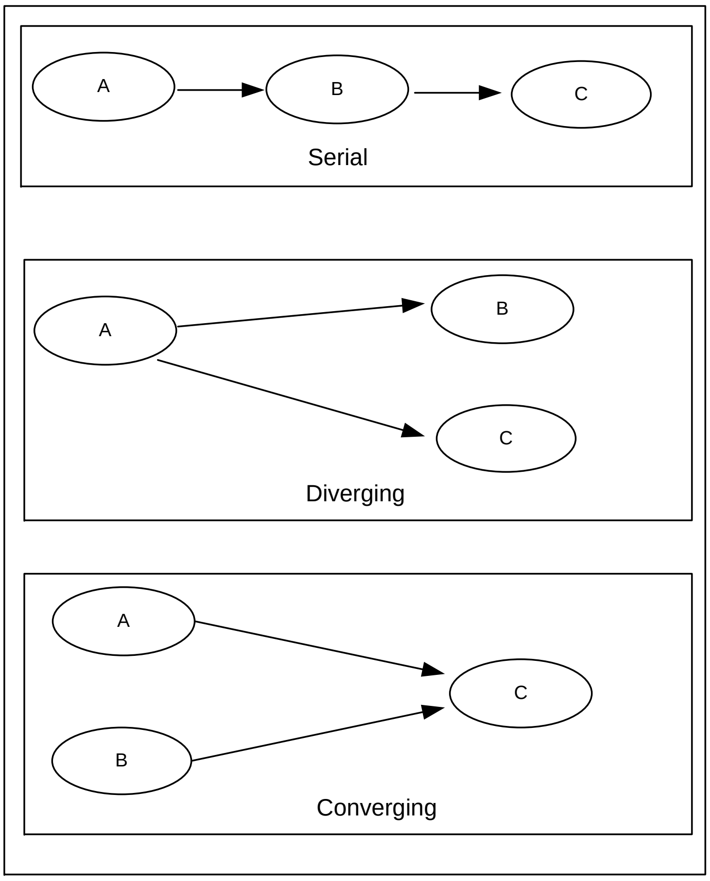


Figure 3.2: Node Propagation in BBN

model based on BBNs to prove that software development team can rely on it to accurately predict and calculate the software risks and their impact on a software development project. The proposed model is conceptualized into a tool that eases to understand and calculate the risks of development project. The tool can be used at any phase of the development and only requires to input the initial probabilities of the risk factors or default probability value when the real value is not available. Then it generates the impact weight level in terms of a numeric value. Fenton et al. visualize software risk by turning the risk into causal model [FN08]. They consider causal taxonomy of a risk by specifying causes, consequences and control actions or a mitigations. The main concept is based on BBN and further used to measure the risk [FN09]. Wagner [Wag09] considers activity based quality model to derive Bayesian networks for software quality assessment and prediction. The approach uses activity, fact and indicator nodes by following activity, fact and metrics of an activity based quality model. BBN is successfully applied to assess the safety level of a safety case [FLN$^+$98, WK07] and calculating return of security investment of developing a secure system [HGBJ05]. BBN is also used to aid decision makers in domains such as medicine, software, mechanical industry, economy and military. In the medical domain BBN is used for supporting decisions in the diagnosis of muscle and nerve diseases, in antibiotic treatment systems and in diabetes advisory systems.

3.3 Conclusion

There exist uncertainties throughout the life cycle of the software product. These uncertainties involve a wide variety of factors, such as incomplete understanding of project context, inaccurate estimation of effort, change of project scope, market demand and deployment. On the other hand, assessing risks related to these uncertainties is difficult as the observation needs to last for a long duration despite of any evolution such as evolution of technology, scope and business process. Therefore subjective judgement aids such type of estimation. We use subject judgement through Triangular distribution scales and assign it into the BBN's node to model the uncertainty and estimate the risk event likelihood and priority. Risk management paves the way for project management. All obstacles should be removed or reduced at a desired level so that project has a successful journey until completion. For this, a systematic risk management process is important in order to manage the risk. Next chapters provide detailed about the proposed risk management model and use expert judgement and BBN.

CHAPTER 4

Holistic View of the Software Development Risk Management

Contents

The proposed software development risk management model is grounded on the holistic concept, where more than one perspective or vision of the software risk is considered. By the term holistic view, we consider risks related to technical and non-technical issues. This section illustrates details of the software development components and their refinement through component-element-factor hierarchy so that goal-driven risk management can be considered from a holistic view.

4.1 Holistic View of Software Development Risk Management at Early Development

To develop an effective risk management model, it is important to consider risks from all technical and non-technical dimensions of the development. Note that, technical issues are directly associated with the hardware and software of the system such as tool support, development platform, project technical complexity, specific device or hardware and operational specification of the product to-be developed and deployed. Non-technical issues concern with the organizational environment, project execution, development process, methodological and managerial issues. We consider goals as objectives, expectations and constraints of technical and non-technical issues and risk factors that obstruct these goals. A details of early development components and project success factors are the fundamental sources, which ease to understand the background foundation for the construction

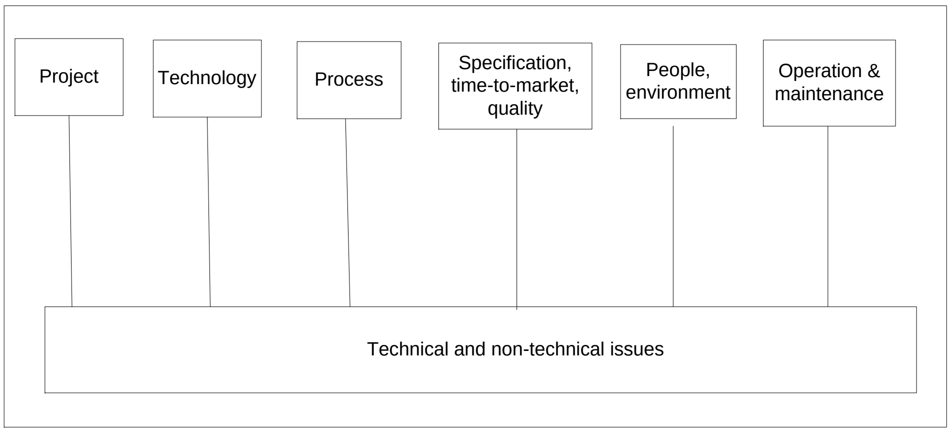

Project
Technology
Process
Specification, time-to-market, quality
People, environment
Operation & maintenance
Technical and non-technical issues

Figure 4.1: Holistic View of Software Development Risk Management

of the goal-driven risk management model. The approach focuses on the components from the early development phase, i.e., pre-project planning, project initiation and requirements engineering phase. The focus is more on the issues related to requirements engineering phase. The main task of requirements engineering is to describe the problem space completely and correctly and to produce artifacts concerning a comprehensive requirements specification document. The integration of risk management into requirements engineering facilitates to manage any change related to cost, schedule, scope, requirement and quality rather easily. For example, a study found that cost related to fixing errors during the testing phase is 20 times more than the cost of fixing these in the requirements phase [Boe81]. Furthermore, this integration also allows to identify, manage and trace the critical software risks from early development phase.

However, there exists numerous errors in the requirements engineering phase, which pose the most expensive software risks. Research results concluded that poor requirements are the main cause of the project failure [Gla98, vL09]. Developers fail to address requirements because they consider requirement specification is the responsibility of customers. But customers rarely have a clear conception of their problem domain and often not being able to state their requirements explicitly. Although they still expect that the end-product meets their needs and supports the business purposes. Sometimes customers are also not actively involved during the requirements elicitation process, which makes the specification erroneous. Developers when involved in requirements engineering activities may not have adequate project domain knowledge or poor understanding of the project scope [Gla99]. Requirements engineering process may be ineffective. Practitioners commonly focus more on the solution oriented view rather than understanding the problem space. Projects may not allocate adequate schedule for the requirements engineering. These are the causes of the requirements problems. Therefore, if factors related to the requirements problems are addressed up-front, even before the actual elicitation of requirements, practitioners in development team can effectively contribute to reduce the requirements errors.

Besides requirements problems, one should also focus on the other issues related to the project execution, practitioner, user and environment that are responsible for software risks. There is a tendency of the project managers to emphasize more on the technical issues. Research results suggest that the cause of most project failure has little to do with the technical issues. Failed projects suffer from the poor management of people related problems [McC96, Lin99]. Non-technical issues are more difficult to manage compared to the technical ones. It requires certain expertise such as ability to estimate accurately, motivate practitioners and well planned project execution activities. Issues like management support and leadership quality, individual productivity, customer/user involvement and user-practitioner relationship are also important from the non-technical perspective. Non-technical issues are not straight forward to manage. Education or training are not adequate to achieve these qualities and individual needs certain experience. Therefore risk management framework needs a holistic approach, as shown in Figure 4.1. Risk management should consider issues related to project execution, technology, development process, product specification and quality, people, organizational environment, operation and maintenance. These dimensions combine both technical and non-technical issues and provide a systematic way to consider risks.

4.2 Project Success and Failure Factors

Before going to the details of the development components, this section specifies factors related to project success and failure, so that expectations and obstructions of the components can be systematically determined. Thus we provide a greater understanding of the early technical and non-technical software development components from the perspective of both project success and failure. For any project, it is necessary to know what generally constitute a successful project and which are relevant for a specific project context. However, what constitutes a successful project, i.e., what is the definition and perception of success in this context is ambiguous. Defining software project success or failure is a complicated undertaking and is rarely described in absolute terms [PV09]. But the factors related to the project success and failure are necessary to identify the goals and risk factors.

Project Success The perception of a success and successful project differs significantly among the various project stakeholders, i.e., customer/user, practitioner and senior/executive management. The reasons are that each of these groups has different backgrounds, responsibilities, expectations and understanding for evaluating the project success. Generally well accepted industry standard and organizational managerial definition of projects success are cited as: meet agreed upon business objective and complete on time and within budget [Lin99, WvMHR00], meets all customer/user requirements and achieve user satisfaction [JKD02, PVSG05]. The management position appears to be more political oriented to keep customers and users happy rather than the developers. Project practitioners and other stakeholders add additional factors and associated complexities to define the project success. User satisfaction is the most widely used measure in the literature for the project success, it is measured by four major constructs: information quality, system usefulness, system usage and system complexity [CDN88, JK00]. The factors related to the customer/user perspective are: realistic expectation and requirements, active customer/user involvement, review and feedback. User involvement supports the developer of better system design and contributes for a more usable final product [PV09]. Practitioners also concern values about the achievement of the business objectives [Lin99, PVSG05] and do not blindly judge only the budget and schedule issues for the success. Specifically, project practitioners always focus more on the micro-level inward-looking project view that is due to their intensive involvement during the development compared to the management [Gla99, Pre96, McC96, Lin99, IHMFJ09, IH10]. They distinguish a project to be a success whether it is *completed* or *canceled*. Some of the factors related to the success are:

- Technically realistic requirements
- Realistic estimation of schedule, cost and effort
- Product works as per specification
- Effective leaders
- Effective team performance, specifically small and high performing team
- Clear and complete project scope
- Diverse and synergistic development team
- Adequate development facilities
- Effectiveness of project management
- Participation and support of senior/executive management
- The practitioners' individual issues such as motivation, team work, sense of achievement, and perception. Motivation has the single largest impact on practitioner productivity.

Researches consider a project success from multi dimensional concept, i.e., Delone et al. [DM92] consider success through the system use, information quality, system quality and individual and organizational impact. Project success depends on issues related to the development components. The project goals must support the stakeholder and business needs. Development team and development process confirm the estimated cost, schedule and desired quality for a successful project.

Project Failure Project failure opposes the project success. Factors related to the failure directly or indirectly obstruct the project success factors. Similar to success factors, there exist several study results on project failure factors. A survey result has showed, that project management and organizational issues are the two main perspectives of the project failure [S.N09]. Linberg [Lin99] also obtained experienced project participants' opinion about the least successful project they have worked on. The participants have emphasized on poor project management and poor marketing research as the main factors of failure. Lack of project management also fails to estimate the human and technical resources necessary for the project. Poor communication between end users and developers and level of customer and user involvement also contribute both for the success and failure [PV09]. Lack of general expertise on the team and lack of clear role definition are considered as two common obstacles, that have more significant impact on project effectiveness [JK00]. There are other influential factors contribute to the project failure: overly optimistic estimation, inadequate project planning and inadequate change management [Bro95, McC96], lack of resources, poor understanding, customer /user passive involvement, lack of domain knowledge of the customer business process and inadequate requirements gathering [KS04, AMN07, IN08, IJH09].

Both success and failure factors are mainly identified from the stated survey studies of the experienced practitioners. A project success factor specifies a number of success indicators, contribute to the project outcomes and vice versa for the failure. Furthermore, these factors combine both technical and nontechnical aspects of the development. Project success or failure is a question of perception and the factors related with them vary from project to project context. It is indeed a multidimensional concept. We carefully analyze the factors from the development components as base line foundation for the goal-driven risk management. We believe that if these components receive more attention at early stage then the development activities can avoid or minimize the problems significantly towards a successful software development project.

4.3 Software Development Components

To develop a goal based risk management model, it is important to understand the basic components of the software development, in particular, what it takes to succeed within development and future use and maintenance of the system. This section contains detailed elaboration of the development components that are essential for any project.

According to Boehm [Boe91] and McConnell [McC96] effective and efficient software development and a ultimate project success can be framed in terms of people, process, product and technology. These four dimensions are related upon each other. For instance, process guides people to perform development activity and technology assists the development effort for a successful product. Saarinen [Saa96] also considered four dimensions: system development process, use, quality and organizational impact as measures of the system success. These dimensions illustrate that a system can be both successful and unsuccessful depending on the metric selected. Procaccino et al. [PVOD02] further categorized seven factors such as management, customers and users, requirements, estimations and scheduling, project manager, software development process, and development personnel, which contribute to the success or failure of software systems. Wallance et al. [WK04, WKR04] considered software project risks into six dimensions. They are: team risk, organizational risk, requirement risk, planning and control risk, user risk and complexity risk. The nature of the project has high influence to these risk dimensions. For instance, strategic application observed higher complexity risk and outsourced project exhibited greater team risk. Different risks have different level of impact on the project, specifically on the development components. And elaboration of these components supports the link between the factors related to the project success and failure.

We summarize the development components into five important dimensions. They are: project execution, development process, product, human, and environment (internal & external). These dimensions are multidimensional and consider fundamental issues related to the development and future use of the system. Individual component rather provides an abstract view which is generally comprised of single or multiple elements. Elements are the essential parts that describe a component. The elements may further be characterized by single or multiple factors, if necessary, also refined into sub-factors. Factors are the lowest level refinement of the development component and represent a concrete aspect of the development. Elements and factors together represent the components, by following development activities, project execution, product specification and quality, human and environmental issues and the resulting artifacts. GSRM defines this as a component-element-factor hierarchy, as shown in Figure 4.2. E.g., project execution component are described by elements, such as planning and control, scope and tool support; where planning and control are further refined into factors such as budget, schedule and milestones, monitor, complexity and change management. Generally, the elements are intertwined, interdependent and contribute to attain single or multiple development goals. GSRM focuses on the expectations, objectives and constraints of the development components that directly and indirectly relate to the project success. This hierarchy includes both technical and non-technical development issues, which facilitate to consider a holistic view on software development risk management. An overview of the components is given below:

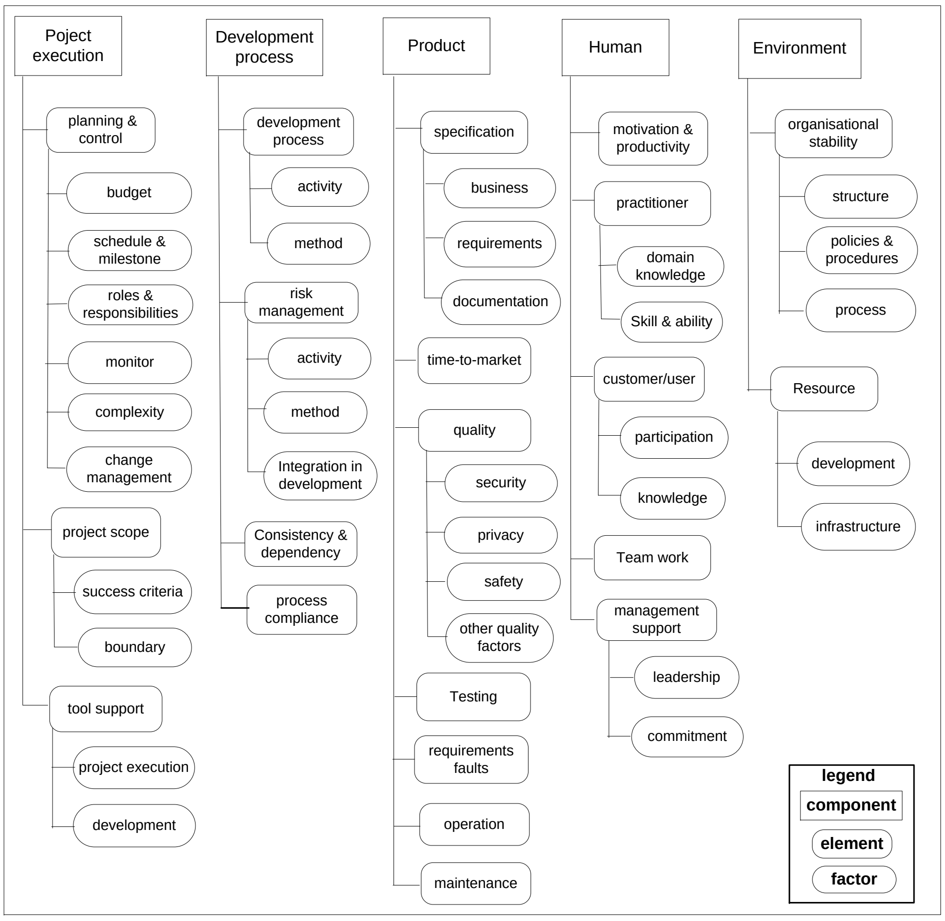


Figure 4.2: Component-Element-Factor Hierarchy

4.3.1 Project Execution

This dimension focuses on the issues related to the execution of the software development project and its economic benefit. It is highly dependent on the human factors in particular project manager's skill and ability to estimate, execute and manage the project. Project managers also have the important responsibility of defining specific roles for members of the development team [EM97, JK00] and tracking progress through their support for on-going status meetings and reports. The development component contains several elements, which further are characterized by the factors and sub-factors. Figure 4.3 depicts the details of this dimension.

4.3.1.1 Planning and Control

Software projects are inherently complex, without careful plan projects can never be succeeded. This element contains certain project execution characteristics such as project plan, budget, schedule, roles, change management and monitor of the overall status of the project. A well planned project will be actively controlled, visualized its progress and people will be given the support to accomplish specific

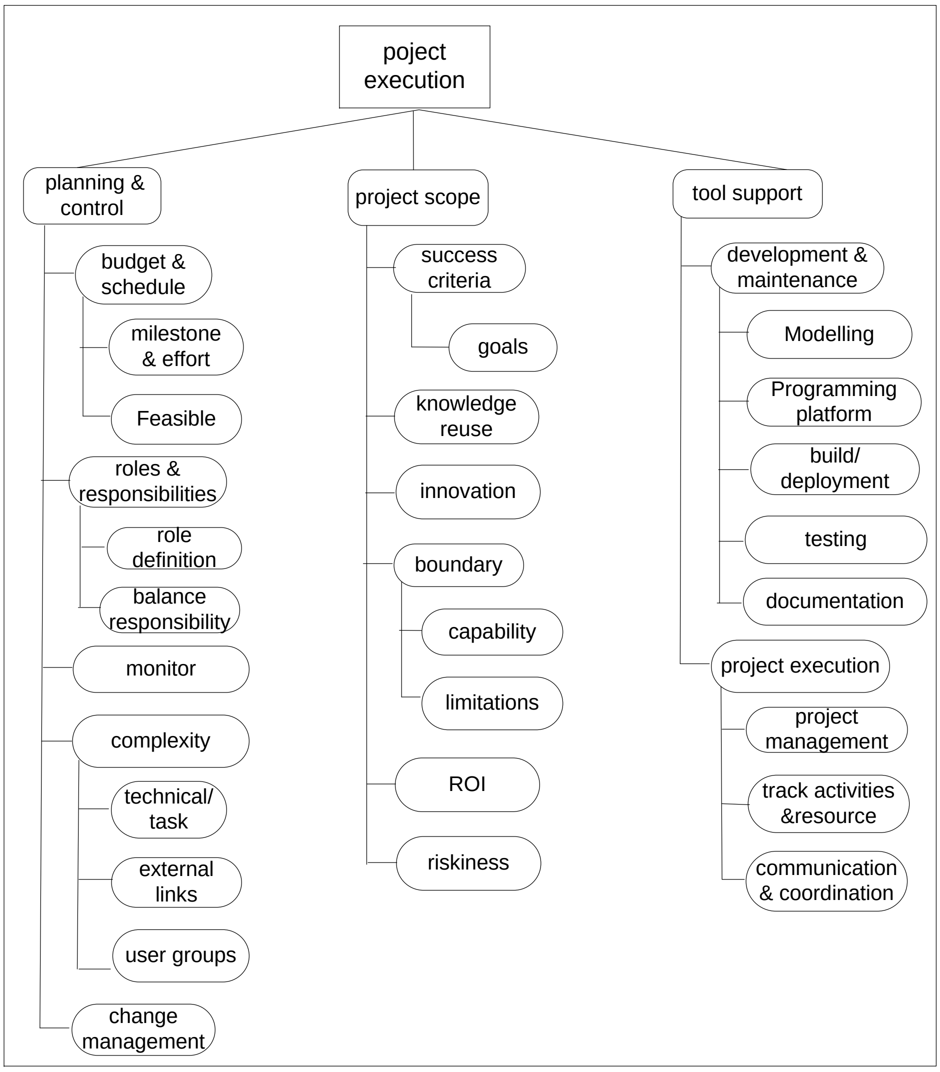


Figure 4.3: Project Execution

project task. Failure to plan is one of the most critical mistakes of any project context [McC98]. The factors and associated sub-factors of the project planning and control are given below.

Budget, schedule, roles, and monitor Budget, schedule, resource and effort estimation should be realistic, feasible and agreed with the main stakeholder. An experienced project manager can assist for the accurate planning and controls of the software development project. He also needs to be aware when it is beneficial to add people to the team [McC96] and recognize when something has gone wrong in the project. There are many roles needed to be filled for performing the project management, development, risk management, deployment and maintenance activities based on the specific responsibilities. Clear definition of this role and balanced assignments of responsibilities to the involved stakeholders is critical for the project planning and control. The roles and responsibilities of the individual task should be clearly defined during the project planning and control. Typical project roles include project owner, customer/user, project manager, team leader, business analyst, requirement engineer, developer, designer, tester, documenter, trainer, release manager and risk manager. Note that, the nature of the roles differs according to the process and methodology that is used. For instance, in a distributed environment, there exist addition roles, such as coordinator of the distributed sites, site manager and infrastructure manager. Estimation of efforts, costs, and duration as well as allocation of resources should be accurate and realistic. However, realistic estimation is not always possible ,in particular, at early stage of the development. For instance, requirement elicitation and coding may take much more time than the initial estimated ones. Some factors are difficult by nature to be estimated, i.e., project complexity, riskiness of the project, practitioner motivation and productivity. Subjective judgment and previous project data assists the overall estimation process. But over-estimation or under-estimation can pose potential problems. Over-estimation might cause the project to take longer time and allocate more staff that the actual one and under-estimation is always danger to the product quality [HC99].

Once the project planning and control factors are defined and the project is under way, then one of the main focuses is to monitor the project progress. A continuous monitor is necessary to track what is occurring and to compare the actual achievement against the estimated budget, schedule and deliverables. It includes the revision, if required, to bring the project back on its target. Project managers in case of small project or project steering committee are responsible to monitor the project progress. Furthermore, practitioners of individual teams are also monitor specific part and report to the team leader or project manager. Formal meetings after certain time interval generally take place to update the status of the project activities. But frequent monitoring is not always possible because it takes time and uses resources.

Complexity This element considers the inherent complexity of the software project. Complexity in a software project arises from various aspects. It depends on the project context, size, user groups, subsystems, external interface, development environment, operational and maintenance constraints. Common factors of the complexity are: highly complex task being automated, immature technology, large number of external links, high number of user groups who will use the final product, new technology and environment where the software will operate, hardware with which the software must associate, data migration and product volatility. These complexity factors also determine the riskiness nature of the project. Large project size is by nature complex. In a survey study, practitioners mentioned the difficulties of managing changes to large projects with many requirements [PV09]. However, at early stage it is also difficult to address the issues related to the project complexities. We advocate to determine the main influential areas of project complexity at early stage, so that issues related to the complexity can be addressed throughout the course of development.

Change management Software under development often goes through numerous changes throughout its life cycle. These changes take place because the requirements and system change continuously, people make mistakes, hardware manufacturers make changes to the system, engineers introduce improvements or change of laws and legislation. Customers, users or even the project manager may demand for any change at any stage of the development. For instance, at later stage users get a clear idea of what is really needed. Wide varieties of changes at later stage or inability to control the changes can expose in many cases insurmountable risks including expensive rework, increase time and schedule. Thus certain changes link with risks. And a systematic change management is critical for adoption of the changes. We focus on analyzing the changes through the goals, in particular, by reasoning why a change is required and what are the factors that obstruct the change and impact of the change to the product artifact. This research attempts to link the critical change with the goal-driven risk management, so that changes are properly analyzed. Nevertheless, there exist factors, which require adequate attention to control the risks related to the changes. They are: key user active participation from early development, change analysis through major project goals such budget, schedule, quality, feature, project complexity. A complete change control procedure (i.e., practice of evaluating, controlling and approving important changes in a systematic manner) is also necessary. A well defined change control procedure protects the project from unnecessary changes, increase accountability of accepting or rejecting the changes and manage changes in an effective way.

4.3.1.2 Project Scope

For any software project, the project scope needs to be clearly agreed with the main stakeholders. It provides a common vision, before the project gets rolling. The scope contains criteria, which are committed to the success of the project and certain boundaries, which limit the project scope. The boundaries define the inside and outside issues of the application and their interactions within the environment. Project boundary concerns with the project capabilities and limitations. It ensures alignment of all the stakeholders' expectations into a common direction and decreases the possibility of scope creeps. The scope should also determine the level of information reuse from the previous project and inherent novelty of the project. If the project can reuse the practitioners' knowledge from previous project due to similar type of project and less innovative then certainly scopes are achievable. A high innovative project increases challenges in the project. The scope should also confirm the economic feasibility of the project. By economic feasibility we mean whether it makes sense to undertake the project. Project scope also supports to determine the main scope of risk management. It is the entry point, which guides to initiate the development activities and allows to reason how the system under development will fit into the future customer environment.

4.3.1.3 Tools Support

Tools support contributes for an effective development project. It can be categorized into two different groups, e.g., tools to support development and maintenance and project execution.

Tools to support development activities are generally used for modeling, programming, testing, build/deployment and documentation. For instance, requirements are sometimes modeled using UML use case diagrams or simply documented as plain text. Once the requirements are specified, then the desired features are accommodated by design modeling tool. Models provide initially abstract descriptions of a system, they need to be realized into a product by programming or coding. Since programming is done using some programming language, the tools involved need, at least, to support the actual coding. Testing is also done on the end-product, it is also important to look into what type of input the testing tools can operate on and how these tools relate to the modeling and programming tools in terms of re-engineering and similar. Build or deployment tools are used to package services or applications for release, which includes making a self-executable instance of the service/application. It needs to consider the issues related to the product operation. For every stage of the development, documentation is crucial. Manually documenting a system is time and resource demanding. Hence, any type of automatic documentation ability in modeling and development tools is important for an efficient maintenance and evolution of the product. Manually adding information in automatic generated documentation is not always straight forward and often avoided or forgotten. Documentation also needs to provide maintenance support of the product. If the project mainly focuses on the maintenance or partially considers maintenance, then similar development tools can support the maintenance activities.

Tools to support project execution mainly focus on the project management, communication and collaboration within the project. Project management tools govern the administration of a project and hence support the project leader and participants in terms of schedule, cost, milestone and activities. Such tools should also aid in the planning and evaluation of a project and in the allocation of resources to the different tasks. However, if the project leader has no other means to gain any knowledge of the people involved, such as people involve in distributed development sites, he needs to rely on tool-support to keep track of abilities, productivity, schedule, cost and other estimation variables. The communication and collaboration tools are used to support handling check in and check out of documents, as well as simultaneously working on the same document and graceful integration of changes. If the development site of the project is distributed, then the communication and collaboration tool is the main way to exchange information and update the artifacts and project reports. However, the tool selection depends on several factors. Tools can support effectual collaboration, proper project planning and progress monitoring. When it comes for the development tools selection then we need to focus: modeling technique, traceability facility, programming platform and team location. Factors and sub-factors for the project management tool are: tracking people, activities, schedule, budget, communication and coordination, document exchange, any particular information relating to the project, productivity and correctness. Finally, project participants require expertise regarding the usage of the selected tools.

4.3.2 Process

The software development process commonly deals with the activities and methods that people use to develop the project artifacts. The activities can be organized in different ways, which are known as process model, if required, tailored by following the project context. Adequate development process is essential for a software project [PNJE06]. This component includes processes related to development, deployment, maintenance and risk management. Therefore, it includes both management and technical methodologies. Generally, an individual activity contains single or multiple tasks and task may further categorize into sub-tasks and steps responsible to perform certain actions. Some people see software process as rigid, restrictive and inefficient and often ignore of using the complete process. A project with little attention to the process may spend more time in meeting or correcting defects, which takes more time [McC98]. Therefore, an investment of well-defined and effective process at the beginning of the project produces large return later. Figure 4.4 depicts details about the process and the associated elements and factors.

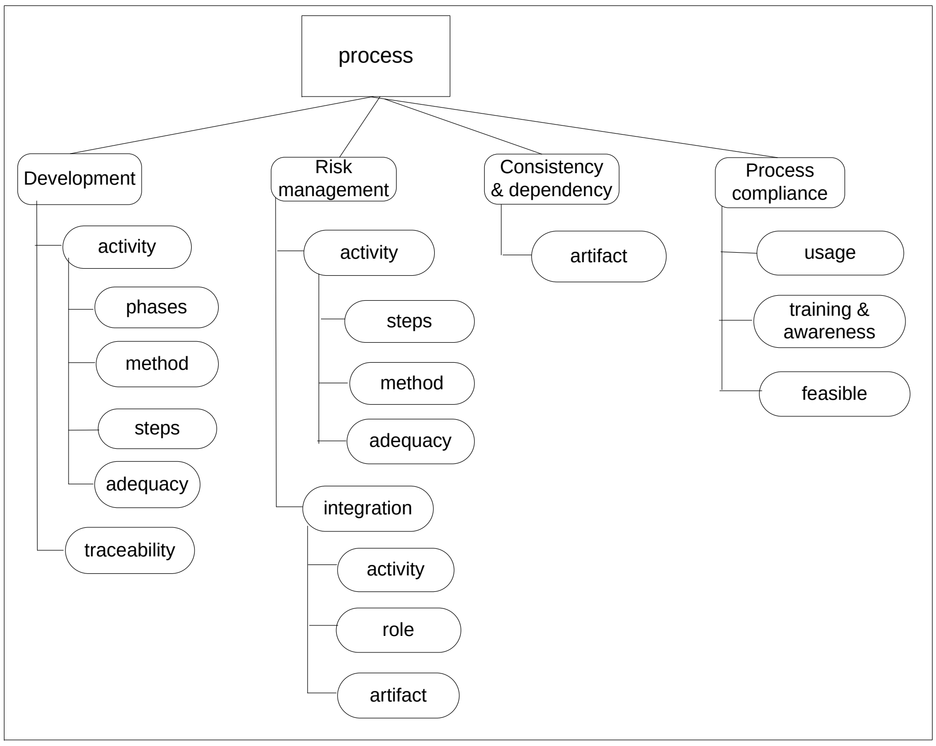


Figure 4.4: Process details

4.3.2.1 Development Process

The dimension of the development process focuses on the activities of the life cycle phases, specifically requirement, designing, coding, testing, implementation and maintenance. It represents an area of high leverage for improving the development speed [McC96]. Process related factors such as effective method and complete definition of steps within each phase, adequacy of the development activities, roles definition and strong coordination among the roles are necessary for the development process. The development process should have a strong focus on the customers' needs and desires. Requirement determination is one of the most important steps in software development process [SS97] and uses a base line for downstream development activities. Identification and management of complete system requirements helps to alleviate costly rework [BB01]. If the identified requirements are appeared to be unrealistic, then developers may become discourage and not fully committed to the project goals. Survey results have showed that requirements specification and managing user requirements were the two biggest problems associated in the software project [LW00]. Requirements management includes relative changes of project scope and understanding between the development team and customer/users regarding requirements and the functionality of the final product. Factors like adequacy in modeling and managing of requirements, realistic/ achievable requirements and traceability of requirements to design decision and further phases of the development as well as product operation and maintenance are essential for an effective requirements engineering practice.

Once the requirement are finalized then the development process continues with the architectural and detailed design, coding, debugging, testing and creating of potential releasable product. It is important to complete most of the requirement work before continuing with the design. The architecture provides a technical structure of the project for the conceptual integrity. Bad architecture and design make the rest of the project difficult to complete. In reality, development activities are overlapped to some extent, done about the same time over the life of the project. At early stage, factors like overall system overview, subsystems and their interactions, inclusion of stable requirements, user interface, traceability among the artifacts, subsystem for design and design document are important. At later stage factors like coding standard, quality coding, user training, operative manual, data migration, maintenance plan, consistence among the code, data, quality due to the maintenance are important. These factors have a tremendous impact on the lifetime system costs. Testing ensures that all requirements have been implemented at an acceptable quality level and product will successfully operate in customer premises. Factors like tests plan and adequate test cases, that focus entire system scope are essential for performing the testing.

4.3.2.2 Risk Management

For an effective and successful software development, risk management must be an integral part of the overall system management. It allows to avoid disaster and can significantly improve the software project outcomes [JKD02, Cha05]. Active risk management practice is necessary in the management of technological systems, such as software intensive systems, where system failure can be caused by the failure of hardware, software, organization or its people. Therefore, risk management should be integrated as early as possible in the software project.

In summary, there exists no well-established set of protocols and commonly accepted procedures, which a software development project can follow directly. Most cases, each project is envisioned as a unique and distinct product. Every organization desires its own development and risk management process based on its objective, policy and overall environment. Lack of a well-developed and acceptable protocol causes major implications on software risk management. But organizations tailor any existing standard which the project can follow. Adequate and effective development process can support alignment of the developed artifacts. This also motivates the practitioners in terms of process compliance and can spend most of their time on productive work. Inadequate processes incur rework, excessive cost and the practitioners spend a lot of time correcting mistakes.

4.3.2.3 Consistency and Dependency

A well defined development process does not provide full guarantee for the end product quality. In particular, when the activities of different development phases are distributed in several sites, then the consistency and dependency among the activities and resulting artifacts are necessary. In software project, development and risk management activities are generally sequential in a sense that later stage activity depends on previous activity. For instance, risk mitigation needs the correct risk identification and assessment result as well as architectural design of the subsystems needs the complete specification. Synchronization among the tasks and methods within the activities and consistency among the artifacts are necessary for a complete and correct artifact.

4.3.2.4 Process Compliance

Process compliance concerns with the implementation and usage of the organization's specific process during the actual development. Process orientation is a straightforward way to save time in software project in particular to avoid rework of the artifacts. Practitioners need to be aware and trained the existing processes and underlying techniques. A process is only effective if used properly and if the developers are able to make them familiar with the tool. Process assessment is also important to validate the process for its completeness and effective contribution to meet the specific purposes. ISO standards such as ISO 9001:2000 specifies the adoption of process approach for the quality management system [ISOb] and ISO/IEC 15504 introduces measurement framework for the assessment of process capability [ISOa]. These standards can be used as a guidelines to ensure the process compliance. Process needs to be feasible, so that it can effectively contribute to achieve the project specific goals. Practitioners have a common tendency to neglect the process, specifically when process can be overly rigid, overly bureaucratic or ineffective to attain specific development goals. They believe that process may limit their creativity, nevertheless effective process supports creativity and morale. Therefore simplified and effective process construction is essential.

4.3.3 Product

This component considers the product artifacts such as specification, quality, time-to-market, requirements faults, operation and maintenance.

4.3.3.1 Specification

Business and requirement specifications are the two main artifacts at early stage. Clear and complete specification document by specifying to which extent it meets customer/user expectation and organizational objective support a quality software deliverables. Deliverables may include system functionalities, screens, reports, operational specification and system utilities. Realistic/achievable requirements are desirable. However, generating a complete requirement specification is a challenging task, which depends on many factors i.e., accurate level of abstraction, requirement modeling techniques, complete process, customer/user active involvement, adequate domain information and correct assumption. We follow *artifact oriented view*, i.e., artifact concept, attributes and dependencies to elicit, manage and document the requirements. Furthermore, requirement priority, traceability and validation support the accurate design decision and the implementation of the requirements.

4.3.3.2 Time-To-Market

Time-To-Market(TTM) of a software product is necessary for its competitive existence in the market. The factors related to TTM include theoretical and technical aspects of the software development work as well as work environment and team dynamics of the project [JJ02]. It allows to understand the market demand of a specific product that helps to produce a complete and correct specification of the upcoming product. Software projects always suffer market risks [Sch04].There is no standard parameters to measure this element but factors like product complexity, innovation, market and competitor analysis result, customer expectations, overall development time, quality, technology, and organizational complexity are always influential to determine the time-to-market. The competitor analysis should focus on transition cost, brand quality, product differentiation and total purchase cost of similar product. One study result showed, that requirements stability, time pressure, information flow and individual competence are highly correlated factors with TTM [WA95]. If TTM is not the major goal of a product then some of these factors can be ignored otherwise these factors need full attention.

4.3.4 Quality

Quality builds the backbone of the system. It describes a comprehensive view on the demanded properties of an application, its architecture and environment, which must exhibit within the system landscape [Gil92]. For an effective project, quality assurance activities must begin during the requirements engineering activities or even before, so that early stage errors can be identified , accumulated and resolved. There exist quality models to support the quality assurance activities. For example, hierarchical model is based on a set of criteria or attribute and each of which has a set of measures and metrics associated with it [MRW77]. Recently activity based quality model [DWP07, WDW08] introduces to precisely specify the quality properties by following the activities. The quality model is the central means for specifying quality requirements, planning, quality assurance and evaluating quality. This model specifies the quality requirements based on the system context. McCall grouped software quality into three sets: product operation, revision and transition. The individual set contains several attributes such as operation consists of correctness, reliability, efficiency, integrity and usability, revision consists of maintainability, testability and flexibility and transition consists of portability, reusability and interoperability [MRW77]. Quality attributes are represented through quality requirements during the development, often difficult to implement, but at the same time they are decisive factors for the success or failure of project [Gli07]. The activity model was built based on information about the characteristics of the system and other important facts and their respective influence on quality activities such as maintenance [DWP07]. A continuous decomposition of business goals that steer the description of behavioural aspects and characteristics of the business and the underlying systems is required to identify the requirements from the quality attributes [SFI09]. Well quality management planning w.r.t. project context is necessary. A software project generally focuses on an information system or embedded system development. This distinction is important, because what is appropriate in one context might not be in another. Quality factors such as usability, maintainability, security, privacy, and safety are always important. A brief overview of security, privacy and safety are given below since these factors were considered in different projects during the course of this dissertation.

4.3.4.1 Security and Privacy

Security and privacy are properties, which if violated, may threaten the system for a potential attack. Software development must consider security and privacy issues from the early development stage [IMJ10, HIK$^+$10]. Identifying and analyzing security and privacy requirements by considering the threats, vulnerabilities, privacy harms and relevant legislation at early development is now well recognized by the industry and research community. Furthermore, elicited requirements are also needed to be traced through out the development for a security and privacy aware software development. Security properties such as confidentiality, integrity, availability, authenticity and non-reputation and privacy properties such as consent, enforcement, notice, awareness and participation are required to consider in the elicitation process [AER02, MG06]. Accurate security and privacy level and pay off benefit of the security and privacy investment is necessary for the successfully implementation of security and privacy aware solution. We advocate to focus on the goals and threats from the security and privacy perspective by considering the project context from early stage of the development. However, detailed investigation will only depend on the project context and stakeholder desires.

4.3.4.2 Safety

Safety is a system property which ensures predictable performance of the product under normal and abnormal conditions [Her99]. It is a critical property of a safety-critical system, i.e., space, military, aerospace and automotive systems. It minimizes the likelihood of unplanned event occurrence so that event consequences can be under control. It is also difficult to confirm that a system is fully safe under all conditions. Depending on the project context certain safety factors such as complete error detection and recovery, system states and flaws and safety requirement errors [Neu95] are required to be assessed. It is also necessary to consider and account all environmental factors and conditions under which the software is deployed. Humans factors are important for the safety systems because any mistake can pose severe potential accident once the system is in operation or maintenance. Similar to security, safety requirements need to be identified from the early stage of the development. Safety and security are closely related properties, as both deals with threat and risks to life, service and information. Therefore safety and security requirements are co-related and play an important role in deciding whether the software can and should be used. These requirements should consider the result of hazard analysis and other system failure conditions.

4.3.5 Testing

Testing validates a software product with respect to the project scope, user's needs and requirements. It executes partial or a complete program based on predicated and observed inputs and outputs. Therefore testing serves to identify the errors existed in the product throughout the product life cycle. There are several types of testing such as unit, integration, system and acceptance [Tha02]. Depending on the availability of resources and budget, a complete test specification should be planned at the beginning of the project. In a project, it may not be possible to consider every unit testing due to budget constraints or large number of tests, but unit testing is necessary ,in particular, for the critical system modules. Integration testing combines system components incrementally or non-incrementally and checks the combined system as per the specification. Finally, system and acceptance testing are used to determine whether the produce is ready to release to the customer. Generally system testing is highly technical in nature and marking the end of the development process. Acceptance testing makes the final product ready to delivery. Whatever the types of testing during the development, it should be well-planned, includes documents test cases and test results should be analyzed and addresses before the product delivery.

4.3.6 Requirements Faults

Elicit *defect free requirements* is a vital but difficult part in the requirements engineering. Any error or fault that originates from a requirement can pose a potential

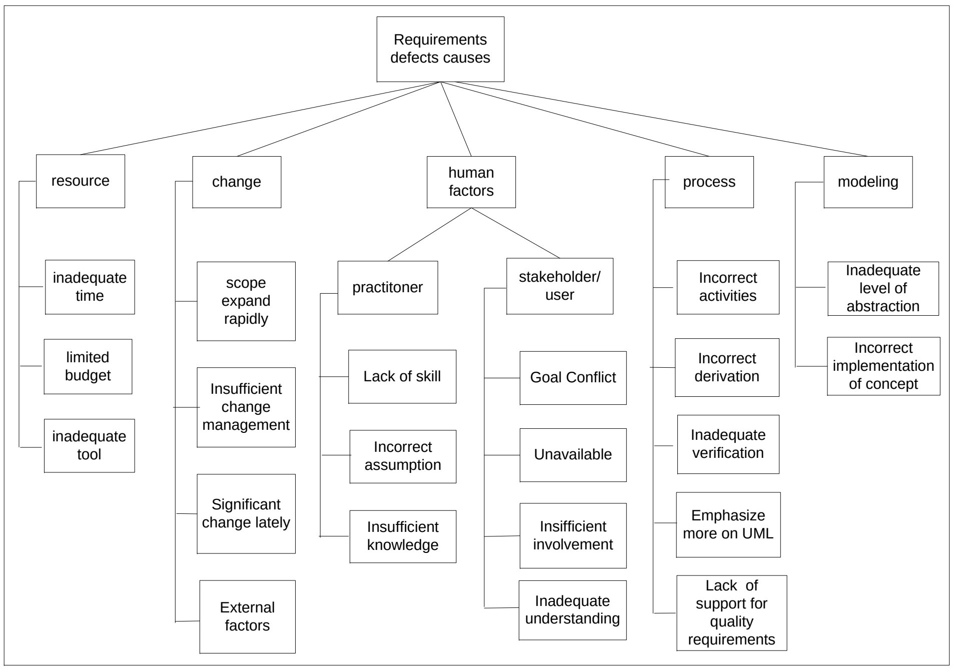


Figure 4.5: Requirements Defects Causes

risk to the development. The identified requirements may be under-specified (e.g. the system shall be maintainable) that makes it hard to interpret the requirement. Sometimes requirements are over-specified which makes difficult to rationale the requirements. Figure 4.5 shows the causes of the requirements defects by considering 5 dimensions: resource, change, human factors, RE process and modeling. Human factors such as practitioners' lack of skill, insufficient knowledge and assumption and users' involvement, inadequate understanding are the common causes of requirements defects.

These defects threaten the overall project success and impact on business, product and project execution. Figure 4.6 shows the impacts of these dimensions on the business, product and project context. For example, impacts can be delayed business benefit, expensive rework, volatility in maintenance and cost and schedule over-runs. In the worst case requirement defects can even cancel the project. It has already been proven in the literature, that to fix a requirement defect discovered at the test phase require much more time as compared to fix it during RE [Boe81]. The most cost-effective point to detect and mitigate defects is during requirements engineering. This task is challenging due to variations of the application and project specific context. Requirement types, risks and assessment strategies vary a lot based from application to application. There is no silver bullet (i.e., single tactic) for managing requirement defects. The activity is a human-based that is always prone to errors. Even if the project size is medium or small, the schedule pressure forces to overlook the requirement defects. It also explicitly depends on the domain expert and the individual perception.

We consider the factors related to the requirement defects. The main focus lies on the business and application specification and user and system requirements. Business specification considers business goals, rules, vision, domains and process. User requirements include use cases and scenarios that present the users' ex-

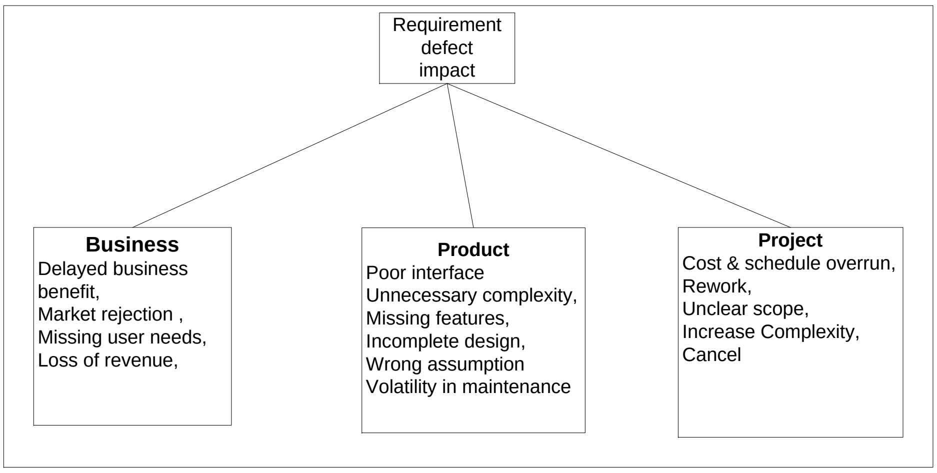


Figure 4.6: Requirements Defects Consequences

pectations derived from the business specification. And system requirement covers functional, quality and architectural requirements. These requirements are reviewed to identify the errors and faults. We follow requirements quality attributes: complete, correct, stable, feasible, measurable, traceable based on the IEEE std 830-1998 [IEE98], requirements implementation, stakeholder goals and product quality to identify the errors.

4.3.7 Operation

Product operation makes the developed or updated system in live to a prepared set of users' environment. At this stage the project ownership moves from the project team to the customer organization. Therefore all project activities should end in a safe, protected and secured way so that the software does not have any negative impact to the day-to-day business operation [Tha02]. This task is challenging as today's customers demand the software to be delivered on time at the lowest possible cost to the operation. The developed solution would not create significant complexity to the overall system environment. Any fault at product operation will almost certainly translate into direct operational and/or financial impacts on the performing organization. The risks may impede that the implementation covers a much broader spectrum than that of the development. Operational failure may include unpredictable cost to the project, delay schedule, loss of revenue and customer/user dissatisfaction.

Several issues needed to be confirmed before delivering the system to the customer. We divide the issues from two different perspectives, i.e., system deployment and transition. System deployment accumulates all preceding development efforts and the resulting artifacts. It includes the execution of all steps to educate the users on use of the system, availability of the required resource, create the environment where the system is going to operate, confirming all data required are available, migrate, initialize and accurate. It also confirms that the system components properly support the overall business functions. Supporting document such as user/technical document and training manual should be available before training the user as well as deploying the system. The training should be carried out as professionally and competently as possible. The first training experience is vital for the successful usage of the system. All training should be performed before the physical migration of the system. System transition mainly switches the system as whole from development mode to the operation and maintenance mode. The transition plan should be aware and synchronized among all the involved parties. It triggers from the deployment activities, which may be beyond the control of the project manger as the product is going to be implemented into the customer premises. Problems may occur at any time during the course of deployment. Expected things may not happen and some thing happens that was not planned. The more the system is complex and extensive the higher is the likelihood that something can go wrong during the operation. Users support activities apart from training such as provide direct assistance during operation, data management and problem reporting and solving during the operation are also important.

4.3.8 Maintenance

This element considers the issues related to the future evolution of the product. Several studies have shown that more than 60 percent of the costs associated with a software product are expanded due to the maintenance, once the product is already delivered [Tha02]. This task is challenging and often technically and managerially more difficult compared to a project which considers only the development. Risks related to the maintenance differ from the risks related to the development [WdOA05] because in maintenance generally the software is in operation that increases difficulties to alter the system. Risk management is critical for the project focused on the maintenance. It can be a costly and risky job, once the product is in operation and initiates the maintenance activity. Maintenance harmonizes the system with the reality and adopts the continuous change issues with the existing software. It can not be avoided. Perfect maintenance of product may or may not increase the value of the software from an economic point of view [SB03]. But it is very subtle, since maintainability to a specific functionality may increase the cost but probably have no effect on the income. The goal of maintenance is to keep the software system actual and useful. It can be event driven, for instance customer needs to modify their existing software to support their business needs, product needs to compliance with the legislation or the technology may be obsolete. Maintenance activity analyzes in-depth the existing system, before any modification can take place. We consider several factors related to the development and management, by investigating the existing literature [Sne96, CAW97, SB03, WdOA05]. These factors highly influence upon the success of the maintenance activities.

The factors are classified into two different groups: development and management, as shown in Figure 4.7. The development group considers the issues related with the artifacts from the development. These issues arise from requirements, design, coding, testing and quality, which trigger the software maintenance. There are several factors of the development responsible to consider maintenance issues. Requirement errors such as incomplete, unstable, incorrect requirements or new requirements from the customer premises to support additional functionalities are important to consider. These factors may increase volatility of the product. The maintenance operation should preserve, if not increase the quality of the system under maintenance. Therefore product error rate, performance, productivity and consistency between documentation and code need to be measured. Complete and correct maintenance specification is important within this context. The sources for the change can be from organizational problems, supporting new business goals, market demand and so on. Therefore management, customer support, marketing and development staff should interview to identify the need for the desired change. Quality factors like performance, resource consumption, portability,

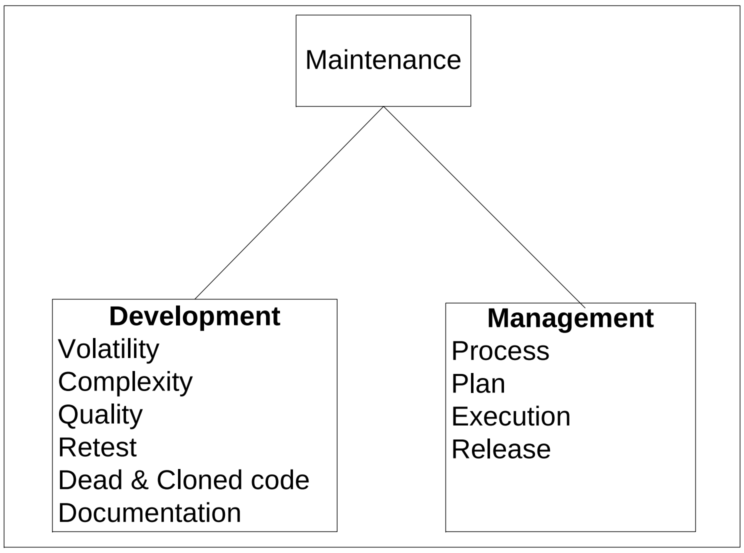


Figure 4.7: Factors related to Maintenance

safety and security need to be analyzed during the maintenance. The maintenance should not violate the existing coding standard and retain at least the similar standard as considered during the development. Cloned or dead code also need to be eliminated. Moreover, it should also not increase the overall product complexity [TBK99]. Modified system must be retested through unit, integration or system test before releasing the updated version. Finally all documents including technical specification, design and user manual should be updated once the maintenance activity is carried out.

The management part considers issues related to maintenance plan, process, execution and release management. Generally maintenance plan and control activity is more intensive than other modes of software engineering project [Tha02]. The plan should include maintenance efforts, required process and specification [IEEa]. In general maintenance should be a controlled process to ensure that the software-to-be maintained continues to meet the end user needs. The process should consider planning, documentation, policies and methods used to support the maintenance. Process should also adequately addresses the issues related to quality, complexity and volatility of the product and adapt the rapidly changes of the customer business environment. Maintenance project or maintenance part of a development project need a separate plan, schedule and budget estimation, role allocation, maintenance contract in terms of what should include or should not include in maintenance and acceptance criteria.

4.3.9 Human

Human plays the central role of software project. Issues related to human have more impact on software productivity and quality than any other factor [McC96]. Therefore, all kinds of human factors can deeply affect the results and efficiency of a project. Humans involved in software project fall into four categories, which are practitioner, customer/user, team and management support [ID08]. It is important to identify the people, in particular, who have a stake or interest in the project as early as possible [HC99]. There always exists inter-relations and overlaps among these four categories. Every category contains multiple factors, which include desirable properties that are essential during the development. But motivation and productivity is more important for every role played in software development. Handling human factors is non-trivial and challenging for the software development. For instance, increasing productivity of a team should first look at the team assigned work, background knowledge, motivation, staff selection, training or focus on the potential benefit of the team. Certainly, it is difficult to achieve satisfactory level of these factors.

4.3.9.1 Motivation and Productivity

The motivation and productivity of individual and teams are necessary in a software project. Motivation is considered as the single most influential factor on the productivity during the development [Boe81, McC96]. Productivity is directly related with motivation. Motivation is an internal engine and its benefits show up over a long period of time. It requires work type, personal sense of achievement and reward [Gla98, Lin99, PV09]. Without motivation, it is very unlikely that a person work hard. Motivation and morale are enhanced, where targets are achievable. Linberg identified sense of involvement, celebrations, positive feedbacks and autonomy; as the main influential factors for the motivation from the developer perspective [Lin99]. Everybody who is involved in the developing project has not the same motivation, productivity and objectives. Project managers need to carefully construct the teams in order to achieve the high productivity.

4.3.9.2 Practitioner

Individual practitioner's factors, such as knowledge, skill, programming or design ability, cooperation, availability and training are essential during the development. These factors increase the overall productivity of the individuals. Practitioners also expect realistic estimation of budget and schedule, technically feasible requirements, interesting and challenging work from the project. These factors motivate the practitioners and collectively offer the greatest potential benefits. They support overcoming problems of the other dimension such as process, product or project execution. For instance, knowledgeable and experienced project management effectively contributes to the project success [Pre96]. There is also a need for negotiation skills in working with the customer and users on scheduling, budgeting and communicating the development team's activities to the users.

4.3.9.3 Customer/User

Customers and users are the people who pay to have the software developed and maintained. They are also responsible to use the product and to accept or reject the final deliverable product. User's adequate knowledge about the project context and IT competence are required for identifying and managing the requirements as well as for the overall development, use and maintenance of the product. User's active participation and close collaboration with the main project participants such as project manager, requirements engineer and architect are critical in several aspects. First, users will mainly use the product and second, incorporating user expectations to the final product increase the chance of project success. User participation from early stage saves time because it eliminates one of the main sources of requirement changes and reduces defect for a quality product. User should also effectively participate during the system deployment because at the end they will use the product. Typically in large software project, there exist various number of user groups. Active participation of the key user is necessary. User involvement supports customer confidence to the development team and products as well as increases trust to the management.

4.3.9.4 Team Work

Team structure and work motivate for high productivity in any sizable project. The team can be in different types: democratic, hierarchical and virtual; based on the structure, communication, roles and responsibilities, power and location [EH04]. Obtaining a common decision is difficult for a democratic team, because no one is in charge. But this type of structure fits well for the agile development process. Hierarchical team is common, however may hinder an effective communication among the team members, specifically when hierarchical layers are increased. Virtual team is suitable for distributed development environment which can be either a democratic or a hierarchical organization structure. In a virtual team, cultural gap, miscommunication and lack of coordination are common problems. Whatever the team type, there are common factors which are important for a productive team work: construction of a balanced team with sound knowledge of project domain and technical issues, talent and creative members, healthy arguments, problems solving, effective communication and coordination, ability to work with uncertain objective and top management. Practitioners who find their work interesting are highly motivated. Therefore, for a productive team work, practitioners must group and assign work that they find interesting. A software development project team, no matter what size, requires differentiating among the roles played by the team members. A small team improves the communication and supports high degree of collaboration. In a project, which runs locally, it is sufficient to have regular either informal contact or meetings among the team members or members of different teams. In a distributed project a more formal way of sharing knowledge and information is required: project members should contribute actively and update their work status via Wiki-pages, discussion groups or whatever the suitable communication means worked for the project.

4.3.9.5 Management Support

Management leadership quality and their direct or indirect involvement to the development activities are already recognized to be essential during the development. Leadership is the ability, which influence the project practitioners to act in a particular way in order to achieve specific goals. Different styles of leadership are needed in different situations [HC99]. However, factors like positive attitude, timely and reasonable decision, motivating and empowering team and removing obstacles signify a good leadership quality. The sponsor commitment needs to be lasted throughout the project. They should motivate the project practitioners for productive and creative work. Management skills (or lack of) have direct implications for project risk management and ultimately project success [Gro95]. Good project leadership establishes a clear vision for the project [McC98]. The management support can also encourage the customer/user to increase the active involvement. They should have high confidence level, motivated to review overall organizational improvement and customer focus for a successful project outcome.

4.3.10 Environment (Internal & External)

Surrounding environment of the software development projects both in-house sourcing or outsourced are the main elements and factors of this component.

4.3.10.1 Organizational Stability

It focuses on the structure of the organization that facilitates the overall project operation. Organizational structures can have an enormous impact on the overall project execution [HC99]. A formal organizational structure, in particular, hierarchical structure is articulated in the staff hierarchy chart. However, this type of structure is supported by an informal structure of contacts and communication which gradually grows up among the practitioners during the course of work. Unstructured hierarchy hinders the decision making process and incurs chaos to the development and deployment of the system. Organization structure is also departmentalized based on employees' skill, customer category, product line, services, and geographic location. Effective communication and coordination among the departments are important to keep the overall stability. In software development domain, functional/task oriented approach or based on life cycle phases are used to departmentalize the groups. Whatever is the structure, free communication and effective decision among the technical, business, marketing, customer support and human resource groups at the right time is critical for an organization. If the groups are more decentralized, then the communication will be slower. Projects with tight schedule and budget pressure need to consider centralized communication. The organizational context also includes processes to support the software development life cycle process. It includes management, infrastructure, improvement and training process by following the IEEE standard 12207 [IEEb]. These processes can be applied across multiple projects undertaken by the organization. Organizational existing policies, procedures, culture and political bias are influential factors for an effective project and risk management as well as software development practice.

4.3.10.2 Resource

We consider two types of resources: one for the development, such as availability of appropriate tools and personnel, adequate training facilities, periodic training need assessment and training evaluation; the other for infrastructure, such as adequate communication(i.e., Internet, telephone) and other(i.e., power, office space, computer) facility. We need not only to ensure availability of the essential resource in key areas when required, but also proper utilization of these resources. Therefore resources are identified and allocated according to the activities during the project plan. Stable organizational structure and adequate development facilities certainly motivate the practitioners to increase their productivity.

4.4 Model Based Development

Model based development is a cross-cutting concern within the component-element-factor hierarchy. Modern software project involves a variety of stakeholders, inherently complex and interconnect multiple components through heterogeneous environment. Model based development using base concept, specific notations, syntax, semantic and level of abstraction effectively contribute to analyze the systems [FR07]. This recent development trend is used for tool support, development process, product specification, quality factors (i.e. security, safety), testing and validation. It describes the real system or part of the system, its characteristics, properties and application view. There are many advantages when one works on the model-level, in particular, cost effectiveness and communicative capabilities providing a discussion platform in the industrial contexts. Models use graphical notation, which facilitates communication of the model across subsystems within various application domains [Her99]. Several factors are involved for a successful model based development, such as level of abstraction, modeling concept and its implementation, tool support, technique, scalability and complexity. There are also other factors, such as lack of abstraction, imprecise mapping between model and code, low scalability, unpredictable and unrealistic modeling view can increase the overall complexity and cost of the model based development. Software projects consider model based development to undertake the complex project part must address the factors which incur risks of using model based development.

4.5 Conclusion

We characterize the software development components in five different dimensions and refine them to elements and factors. Detailed understanding of software development components facilitates to identify the expectations and constraints, which a project should attain. These are the goals obstructed by the risk throughout the product life cycle. Elaboration of component-element-factors allows a comprehensive understanding of the goals and risk factors of a software project. This hierarchy also supports to focus on holistic view of the development. For instance, elements and factors under human and environment are mostly non-technical and elements and factors under product and process are mostly technical, but we analyze them combinedly for the software development risk management. Depending on the actual project context, one can emphasize more on one component and downplay the others. Some of the factors can be fitted into more than one components group. Therefore a focus on one dimension also influences other dimensions. Ideally at early stage we recommend to consider every dimension equally.

CHAPTER 5

Goal-driven Software Development Risk Management Model(GSRM)

Contents

The main contribution of this dissertation is the development of a Goal-driven Software Development Risk Management Model (GSRM) to identify, assess and manage risks in requirements engineering phase and to trace the risks throughout the development and during the product operation and maintenance. This chapter presents the main contribution and describes the goal-driven risk management modeling framework, meta-model, process and the artifacts involved within the approach. Finally, the fundamental concept regarding the integration of GSRM into requirements engineering is also presented in this chapter.

5.1 GSRM Framework

The proposed risk management modeling framework is based on goal modeling languages. Risks are always negations of the goals. Therefore, the concept of risk is directly related to the concept of goal. We believe identifying goals certainly anchors the risk management activities in particular to identify, analyze and manage the risks. As the proposed approach considers risk management activities in requirements engineering, we focus on the existing goal modeling methods of requirements engineering. Goal modeling languages have long been recognized in the requirements engineering community as useful to elicit, analyze, negotiate and document requirement specifications for the system environments. They certainly play the central role in the requirements engineering process. Several methodologies and frameworks such as KAOS [DvLF93, vL09], i* [Yu96], and Tropos [BPG$^+$04] contribute to the Goal Oriented Requirement Engineering methodology (GORE). Our framework adopts the basic goal modeling concept and focuses on the holistic view of software development risk management.

The proposed approach extends the KAOS (Keep All Objectives Satisfied) goal modeling method [vL09] to support software development risk management. This is because KAOS defines an obstacle as a construct that is a pre-condition for non-satisfaction of a goal in terms of expectation, assumption, domain property and constraint [vLL00]. The concept of an obstacle can directly be mapped to the software development risk that negatively influences specific project goals. Furthermore, KAOS also provides a detailed specification about the goals through a goal taxonomy by classifying the goals as hard, soft, functional and non-functional. This allows to categorize the goals related to software development components. GSRM adopts the goal and obstacle concept from KAOS and maps them with project success indicator and software risk. The identified risks must be addressed and GSRM does this by assigning suitable early treatment actions. The approach extends further with risk assessment and treatment for managing software development risks. This allows reasoning and tracing of treatment actions and their ability to mitigate risks, and hence, to fulfill the identified goals so that the project can successfully reach its desired destination. Note that KAOS also introduces the risk analysis based on the goal models. However, the focus is more to ensure the completeness of requirements and to analyze safety hazards and security threats for the critical systems. But our focus is on the software development risk from the holistic perspective considering all technical and non-technical development issues during the development. In our case, the requirements' completeness or security is considered as a sub-goal by GSRM that critically influences the project success.

5.1.1 Levels of Abstraction

The model supports different levels of abstraction from goal to obstacle and finally to the treatment. Figure 5.1 gives an overview on the different levels of abstraction, depicting exemplary questions that symbolize the characteristics of the proposed model. We divide the levels of abstraction into three main areas within the scope of this research. These levels build the bridge from the goals of any software project to the risks that obstruct the goals and treatments that reduce the risks in order to satisfy the goals. On the top, there are the goals, i.e., objectives, expectations and constraints of the development components. In the middle are the risk factors that directly or indirectly obstruct the goal to fulfill and incur problems to the software development. The risk factors cause undesirable events and these events further bring negative consequence to the goals. Risk events are then assessed to estimate the severity of the risk. At the bottom part, there are the control actions that obstruct the risks and their consequences and contribute to attain the goals. Risk treatment also requires to monitor the effectiveness of the control actions and to identify any new risk within the development. Risk specification is the main artifact type produced by the model as shown in the right part of the Figure. It includes several concepts such as goal, obstacle, treatment and associated visualization model.

These levels of abstraction support refinement of goals and obstacles and establishment of the obstruction and contribution link among them. The refinement is fulfilled through vertical abstraction so that goals are traced by the control action for their satisfaction [SPHP02]. The more goals and risks are refined, the better it can support the risk management at early stage. The refinement links are con-

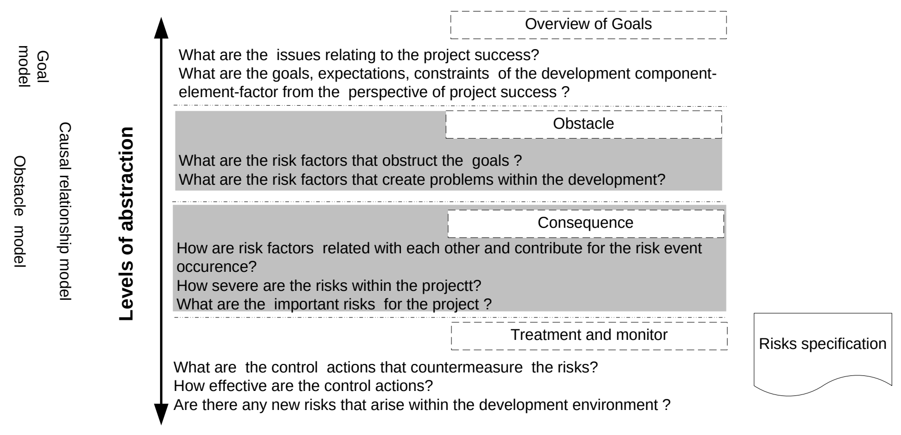

Goal model
Causal relationship model
Obstacle model
Levels of abstraction
Overview of Goals
What are the issues relating to the project success?
What are the goals, expectations, constraints of the development component-element-factor from the perspective of project success ?
Obstacle
What are the risk factors that obstruct the goals ?
What are the risk factors that create problems within the development?
Consequence
How are risk factors related with each other and contribute for the risk event occurence?
How severe are the risks within the projectt?
What are the important risks for the project ?
Treatment and monitor
What are the control actions that countermeasure the risks?
How effective are the control actions?
Are there any new risks that arise within the development environment ?
Risks specification

Figure 5.1: Overview of Risk Abstraction Levels

tributed through two different ways. One way shows the goal decomposition from the high level goals to sub-goals and obstacle decomposition from risk factor to risk event. The other way shows control actions and their contribution to the goal satisfaction. Therefore during top to down as well as during down to top refinement, we have to satisfy the relation among the activities and artifacts to serve the purpose of software development risk management.

5.1.2 Modeling Views

There are three different modeling views structured by GSRM related to goals, obstacles and causal relationship.

Goal model: Goal is the core concept of this approach. It specifies the expectations, objectives and constraints of the development components, stakeholders and business environment related to the project success. The identified goals are refined from higher to lower level so that they can be quantified to more achievable ones. Therefore the goal model shows the interaction among the higher level goals to the lower ones so that they can combinedly contribute during the development to attain the factors relating to project success. The model is represented by a diagram and consists of goals and their refinement links from higher to lower level goals.

Obstacle model: Obstacles are the negation to the goals and responsible for the occurrence of any undesirable events during the software development project and use of the product. The obstacles and their consequences are identified, analyzed and prioritized by following the identified goals. Similar to goals, obstacles are also refined from risk factor to the risk event. Thus the obstacle model elaborates the goal model (i.e. goal-risk model) by including the obstruction link from the obstacle to the goals and refinement link from risk factor to the risk event. We follow the same graphical notation to represent the goal and obstacle models as used in KAOS [vL09].

Causal relationship model: Risk factors, as causes, are refined into single or multiple risk events that negatively influence the goals. This allows us to model the causal relationship between the risk event and associated factors and their obstruction of the goals. The model describes which risk factors are related with each other and how they are accountable to the risk event occurrence. This leads to the combination of all common risks and eases the risk event estimation of single or multiple factors. We use the Bayesian Belief Network (BBN) [Jen96, BC01] to construct the causal relationship model.

5.1.3 GSRM Layers

The framework consists of four layers to support the software development risk management model. The advantage of a layer based modeling framework is that it includes suitable task, method and technique for performing specific activity under a given layer. Each layer supports iterative activities for managing software development risks and produces single or multiple artifacts. These artifacts are part of the risk specification concept that support the decision making process during the software development, operation and maintenance.

Goal Layer The goal layer focuses on the factors that contribute effectively to complete the project activities and directly link to the project success. These goals are important as they describe what needs to be done for a project to be successful and for the responsible agents to attain the goals. Goals in GSRM consider several dimensions of the software development components including project execution, process, product, human and internal and external environment (i.e. as stated in the previous Section 4) and map them to the project success indicators. These goals are project specific and focus on the economic benefit, project success criteria and boundary, knowledge gathering and reuse, user satisfaction, quality, vendor reputation, successful delivery and other critical goals of the product. Goal satisfaction requires cooperation among the system agents. For GSRM, these agents are the project stakeholders, practitioners, tools, language, hardware, development facilities and so on. The main activity of this layer is to identify and model the goals. However, before goal specification, project stakeholder should agree on a concrete risk management plan , in particular, the risk management scope, underlying process, risk threshold and resources. A detailed goal list is the main artifact produced by this layer. Goal modeling supports refinement of coarsely grained higher level goals to finely grained lower level goals through AND or OR refinement. In GSRM, the latter is referred to as sub-goals. Each sub-goal in the refinement contributes to the parent goal. The more the goals are refined the easier it is to identify and analyze the risk factors that obstruct the goals. A graphical representation of the goal refinement is the core part of the goal model. Satisfaction of these sub-goals certainly attains the main goal. Goal types, such as soft goals are suitable for the goal-driven risk management context as there are generally multiple alternatives for single goal satisfaction. The same sub-goal that relates to a specific development component also contributes to the satisfaction of other development components. These sub-goals are important for the project and require extra care for their fulfillment. Therefore, if required, the goals are prioritized according to their importance to the project success.

GSRM provides a set of guidelines for goal and sub-goal formulation, in particular, we attempt to map the existing goal types and categories from the literature to specify the expectations from the software development components and project success factors. We follow the informal temporal pattern to represent the goal as stated in KAOS [vL09]. The pattern structures an assertion into a prefix and a condition/property. Assertion is the statement of intent of some condition/property of the software development component. These goals are hard or soft by nature. There may also have behavior goals, which represent the intended behavior declaratively. Improve, reduce and minimize are the common prefixes for representing the goals. For instance, a statement could be *reduce [erroneous requirement]*, whereby the prefix *reduce* represents a goal that demands a reduction of error from the elicited requirement as undesirable property.

Obstacle Layer Obstacles are the main causes that reduce the ability to achieve a single or multiple goals. We treat risk factors as obstacles that directly or indirectly lead to a goal negation and create problems in the project. Obstacles are the opposites of the goals (i.e., undesirable ones that shadow the goals). Therefore, the obstacle categories should be aligned with and derived from the goal categories and model the situation about how several obstacles violate the identified goals. The obstacle layer enhances the goal clarity. It identifies the potential software development risk factors and formulates the obstruction to the goal dissatisfaction. Similar to the goal model, the obstacle model also refines to provide a complete overview of the risk factors exist in the project. The main focus is to identify as many risk factors as possible so that corresponding control actions can be selected in the early stage. Software development risk factors support different categories of obstacle such as dissatisfaction, lack of adequacy, misinformation, wrong assumption and inaccuracy.

The obstacle layer supports the risk identification activities and GSRM uses a comprehensive list of questions and arranges them by following component-element-factor hierarchy to identify the risk factors. This supports to categorize the risk factors and groups them under the same category. Note that, there are also risk factors that may not directly obstruct a particular goal but pose problems during development. This layer also allows such risk types to be expressed and formulated as risk factors, both within one project, but also as a reusable risk factor component. Examples of such obstacles are: trust between customer and practitioner, legal disputes from a local context of a global software development project. The risk obstacle layer establishes the obstruction link from the risk factors to the sub-goals and from event to the main goal. The same risk obstacle can be relevant to more than one goal. This is important to be expressed by the risk obstacle layer, as it is crucial information when considering effective treatment options. Risk factors that cross-cut several goals are in general more effective to counter. Risk factor identification, categorization, and modeling are the main tasks of this layer. However the risks identified by this layer are not sufficient to determine the control actions because risk factors need to be quantified to determine its severity. Therefore the identified obstacles are analyzed further in the assessment layer.

Assessment Layer The assessment layer analyzes the risk events as a consequence of single or multiple risk factors to the goal. The risk quantification is an important first step in assessing the risk [BRT93]. However, this task is non-trivial due to the inherent subjective nature of the risks in software engineering domain [Kon01]. This layer precisely annotates the individual software development risk event. Furthermore, it also establishes the causal relationship model between risk factors and risk events. The layer focuses on the severity of the risk events consequence to the goal negation. Therefore, relationship among the technical and non-technical software development risks and the corresponding quantitative evaluation to the potential project goals, is the main contribution of this layer. For high prioritized software development goals, obstacle identification and refinement should be extensive. We use the *Bayesian Belief Network* to support the causal relationship model by mapping the risk modeling elements to probability nodes.

Each risk event is characterized by two properties: *likelihood* and *impact* [BC01]. Likelihood specifies the possibility of a risk event occurrence and is modeled as a property of risk event. The impact quantifies the negative consequence by the risk event to the single or multiple goals. The same risk factor may lead to more than one risk events and the same risk event can obstruct more than one goals. On the other way around, a goal is obstructed by multiple obstacles that relate risk events and associate factors. This representation allows to model situations where an event is influenced by more than one risk factor and impacts on one or several goals. An obstruction link is established from the risk event to the specific goal that it obstructs. This supports to construct *goal-risk model* by refining risk factors to risk events and their combined obstruction of the goals. The obstacle refinement is done in reverse manner compared to the goal refinement. The benefit of obstacle refinement is that we do not need to analyze every individual risk factor separately and in real project situation this point is important, in particular when the budget and efforts are limited.

Treatment Layer The final layer focuses on the control actions to counter the risks so that goals can be properly attained. Once the goals, risk factors and risk events are identified and analyzed by the goal, obstacle and assessment layer, then the final task is to implement the suitable cost effective countermeasures. Therefore, the aim of this layer is to control the software development risks as early as possible preferably during requirements engineering phase of the development. The layer is also responsible to monitor the risk status throughout the development and if needed during the operation and maintenance of the product. However, initial considerations focus on the risk events and associated factors that negatively affect several goals, i.e., high prioritized risks. Generally, there exists an alternative countermeasure to the obstacles but should select the most cost effective one for the risk mitigation. This layer includes two different links: contribution link from the control action to the goal that it fulfills and obstruction link from the control action to the specific obstacle that it obstructs. Treatment layer allows modeling, reasoning and tracing the adopted control action for the risk mitigation and goal satisfaction. It also includes responsibility link from the control action to the agent so that specific active agent would be responsible to implement the control action to mitigate the risks.

Risk control actions should minimize, prevent or avoid software risks to attain the goals. However, the project context is important in order to identify and select the appropriate control action. If a project is highly risky from the beginning then the selected control action should initially focus to completely eliminate the risk. However, it is not always possible to eliminate the risk factors. The system agents such as human and other development components such as tools or process should be responsible to perform a specific task for the selected control action. The risk treatment considers *risk threshold*, that is the specific level up to which a project can accept the risk without implementing any control action. Once the selected risk control action is implemented then we need to monitor the risk status until it is completely mitigated. The risk status through the course of development evolves. In particular, control actions may not effectively reduce the risks or new risk may be identified. Therefore, the treatment layer continues to monitor risks throughout the development and communicates with the stakeholder about the risk status by risk status report.

Figure 5.2 shows the modeling framework of GSRM. GSRM uses the same notations for goals(parallelogram) and obstacles (reverse parallelogram) as used in the KAOS model. On top is the goal layer which refines goal through AND and OR refinement into sub-goals. The two middle layers collectively represent the software development risks as obstacles which directly obstruct the goals and incur problems to the development. And the bottom is the treatment layer which ini-

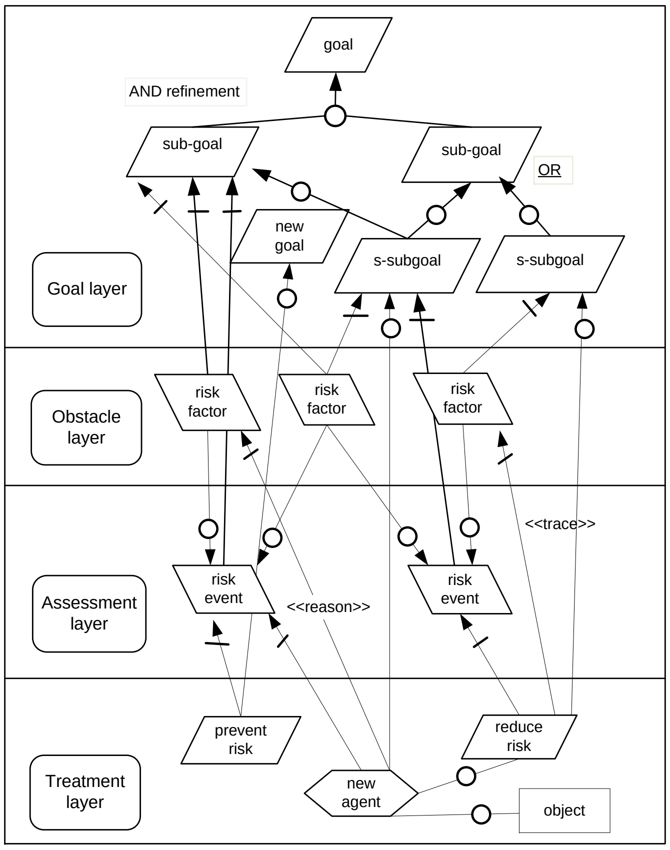


Figure 5.2: Framework of GSRM

tially contains goals in terms of prevention, reduction and avoidance of risks and assigns responsibilities to the agents who implement the selected control actions to obstruct the obstacle. The treatment layer includes contribution, obstruction and responsibility link to the top three layers. The framework initiates with the goal in terms of project success and ends with goals in terms of the risks mitigation.

5.1.4 Meta-model

We integrate all concepts that are used for the goal-driven risk management into a model called *meta-model*. The meta-model represents the underlying conceptual elements and relationships among the elements related to software development risk management. When one puts all concepts together, it is required to confirm the consistencies among the components. Thus a meta-model, as a conceptual model, defines and inter-relates conceptual abstraction, in terms of which other models are defined. The meta-model for goal-driven risk management includes several model, at different levels of abstraction. It includes meta concepts (i.e., development components, goals, obstacles, treatment and agents) and relationships among these concepts (i.e., obstruction, contribution and assignment link). It also includes meta attributes of the meta concepts such as risk level and goal description. Figure 5.3 depicts the meta-model of the proposed risk management approach by including goal, obstacle, causal relationship and treatment. The root meta-concept is the *goal* that appears as the main part. A goal may be OR-refined

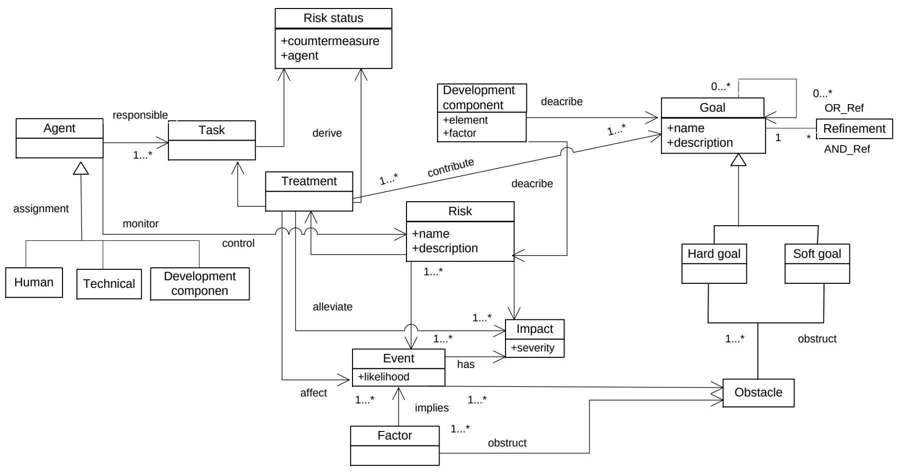


Figure 5.3: Meta-model for goal-driven risk management

or AND-refined into multiple sub-goals. The meta-concept *refinement* is included into the goal model. The goal is described by the meta-attributes such as name, description and category. A goal may be obstructed by *obstacle*. An obstacle contains meta-relationship, e.g., obstruction link from the obstacle to the goals. Both goals and obstacles are described by the development components but from two different perspectives. One is to satisfy expectation, constraint, objectives, or other desirable properties during and after the development. And the other is to incur obstacles and problems that influence any undesirable circumstances and thus hinder the goals. The goal-risk model consists of risk factors as causes and risk events as consequences that collectively contribute to the negation of a single or multiple goals. Therefore, the meta-concept *risk* specifies software development risk with meta-attributes such as name, description, category and severity. The concept *treatment* supports goal satisfaction and risk obstruction. And *agent* is responsible for controlling the risk as a part of risk treatment. Therefore, the responsibility is assigned to single or multiple agent types such as human, technical and development component to control and monitor the risk.

5.2 Generic Process Model

The generic process model describes the methods involved as well as the artifacts produced from the GSRM. As GSRM is integrated within requirements engineering, we focus on the existing activities involved in RE to support the process integration. On one hand, the process requires being systematic and simple to identify, categorize, analyze and control software development risk. On the other hand, the artifacts produced must provide accurate information about risks associated within the project so that informed decision can be made about the project and its goals from the early development stage. The generic process model defines the main activities, artifacts and information flows.

We refer, in particular, to activities, tasks and steps to specify the process model in a systematic way [Fir04, BWHW05]. The risk management process contains five activities that define the major area of concern for the goal-driven risk management. The activity describes all the tasks concerning the creation of the risk specification artifact type. Each task in turn produces an expected output based on

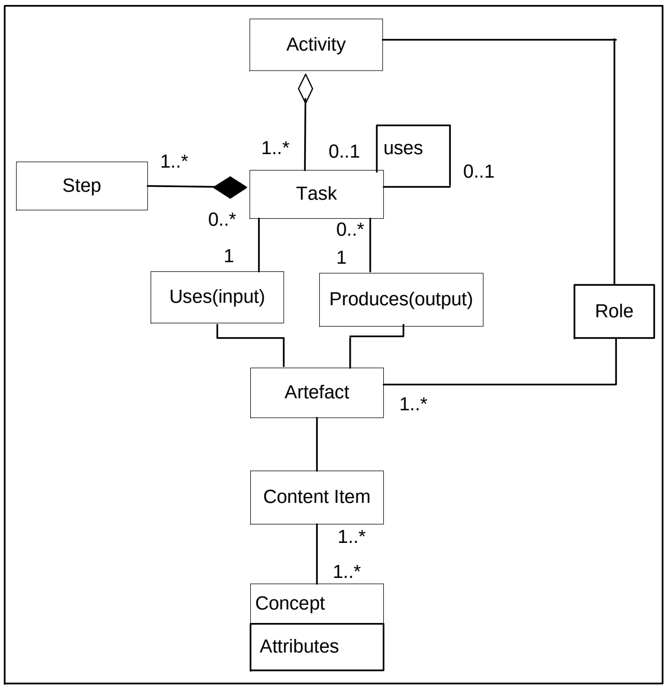


Figure 5.4: Process Model for GSRM

the input. For producing this output, each task, if required, includes steps that define a concrete method for constructing the selected output. Figure 5.4 shows the generic structure of the process model by including activity, task, step and artifact of GSRM. The concept role gives an abstract description of responsibilities that directly participate within or indirectly contribute to the risk management process. A role takes the responsibility for a specific set of artifacts and performs a set of activities within the process in order to produce or modify the artifacts. Sometimes roles indirectly participate within the development process such as management support during the software project, which may contribute to or depend on the completion of an artifact but have no direct responsibility for this artifact. Lower part of the figure shows the detailed about the artifacts, which produced in general by the activity and in particular by specific steps under a single task. The task also requires single or multiple artifacts as input elements to perform the related actions for creating the artifact. The tasks are defined under the activity which has operational characteristics representing the use of techniques. For example, techniques use to identify the risk. Therefore, individual activity is composed of a single or multiple tasks, where each task, if required, is a sequence of steps to produce the artifact. Artifacts are the deliverables that are produced, modified and used by a sequence of tasks as a part of the risk specification. It includes content that reflects according to the concept of a specific domain, precisely defining the used elements and their relation for specific description techniques [SPHP02, Sch]. The concept defines the elements as attribute of a specific artifact and their dependencies with other elements from another concept. The artifact oriented view eases to map the dependency between the requirement and risk specification artifacts. For instance, elicited system requirements artifact is reviewed to identify risks related to the requirements and once the risks are identified, it assists in completing the attributes of the concept.

5.3 GSRM activities

The activities under the generic process model describe all the tasks and steps that are required to perform goal-driven software development risk management and to create the risk specification artifact. In literature, several contributions, as well as the risk management standards agreed, that to be successful, risk management must be run as an iterative process involving repeated risk assessment and project specific risk mitigation activities throughout the system life cycle. The risk management standard in particular ISO/IEC 16085:2006 (i.e., Systems and Software Engineering Life Cycle Processes Risk Management) [IEE06] recommends to approve the risk management plan before performing any activities for risk management. We follow these guidelines to describe the activities and tasks of GSRM. The goal-driven risk management activities are performed sequentially, specifically for the initial iteration of risk management. Then continue further, if required, depending on the project context are tailored. Figure 5.5 gives an overview of the activities, tasks and steps involved within the process model. The initial activity is responsible to initialize the goal-based risk management activities during requirements engineering phase. Once the project stakeholders agree on the risk management plan then sequentially the rest of the activities are performed. A detailed overview of the individual activities is given below. The artifact constructed by the activities are commonly represented in standard tabular format.

5.3.1 Activity 1: Initialize Goal-driven Risk Management

As stated, this is the first activity of GSRM that initializes the goal-driven risk management activities during the software development. The main focus is to approve the risk management as a part of the software development project by the main project stakeholders. Therefore, this activity requires active involvement of the management and project manager to emphasize the importance of risk management at early development. It contains two tasks which are responsible to create the risk management plan concept.

5.3.1.1 Determine riskiness of the project

This task is generally performed in pre-project planning phase. However, if the task was not carried out earlier, then GSRM advocates to consider it before identifying the risk management scope. Every project has a common goal to not loose any money. There are also projects which organizations carried out to obtain experience/ knowledge or high reputation for the future business gain. However, whatever the benefit at the end, most projects focus on return of investment. Therefore, economic feasibility in terms of potential costs and benefits both in a quantitative and qualitative way and technical and operational feasibility are generally extensively analyzed during the project planning phase. The planning also needs to understand the inherent existing risks of the project. Determine project riskiness decides how risky the project is in terms of cost, schedule, development, deployment, market demand, user satisfaction, profit and other related factors. It allows to justify the rationale whether it is worthy to undertake the project. Project riskiness can be determined in three different scales: high, medium and low. It helps to understand critical input for GSRM by defining risk management scope and boundary, main sources of risks and time interval to undertake risk management activities. GSRM identifies and elaborates the main risks, understands their causal relationship and consequence of the risks for high, medium or low risk projects. There are several elements of software development projects,

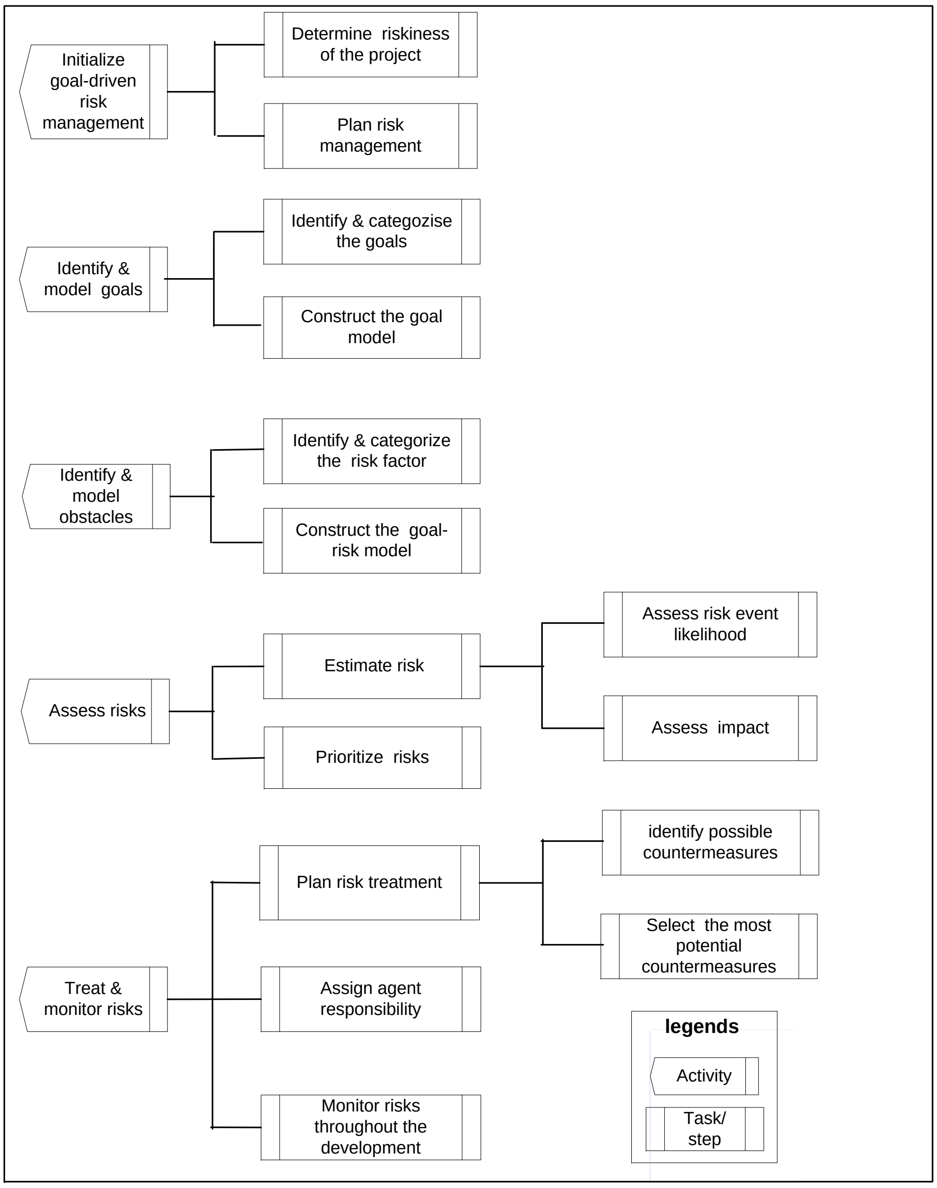

Initialize goal-driven risk management
Determine riskiness of the project
Plan risk management
Identify & model goals
Identify & categozise the goals
Construct the goal model
Identify & model obstacles
Identify & categorize the risk factor
Construct the goal-risk model
Assess risks
Estimate risk
Assess risk event likelihood
Assess impact
Prioritize risks
Treat & monitor risks
Plan risk treatment
identify possible countermeasures
Select the most potential countermeasures
Assign agent responsibility
Monitor risks throughout the development
legends
Activity
Task/ step

Figure 5.5: The Activities and Tasks for Goal-driven Risk Management Model

as shown in Figure 5.6, which need initial attention for determining the project riskiness [RL00, WK04]

Project execution factors, such as budget and schedule, scope and complexity highly affect all dimensions of project riskiness. Generally, if a software project contains less challenges in terms of complexity and innovation, realistic schedule, budget and deliverable , short duration and practitioners have adequate experience from similar projects, then the overall project risk is certainly low. On the other hand, if the project tasks are challenging and innovative, long duration, unrealistic schedule, budget and deliverable and practitioners having less domain knowledge about the project context, then there is high possibility of a high project risk. The decision whether project is developed in-house or should

be outsourced also influences the project risk. Study results showed that outsourcing is risky [RKC96]. We demonstrate a simple example to outline the riskiness of projects in terms of total cost.
Let,
TB= Total Project Budget
TE= Total Expenditure
ROI= Return of Investment
If $TB \approx TE + ROI$
Then, the project is in safe situation as there is very less possibility of budget overruns. The project can reach its economic gain. It would not be a high risk project if the focus is only concerned with economic gain.
If $TB \leq EC + TB$
Then there is a certain chance that the project would exceed its estimated budget. However in that case, stakeholders may consider what other goals can be achieved such as knowledge gain, high reputation from the users. This context eventually leads to a high risk project if the goal mainly focuses on ROI.

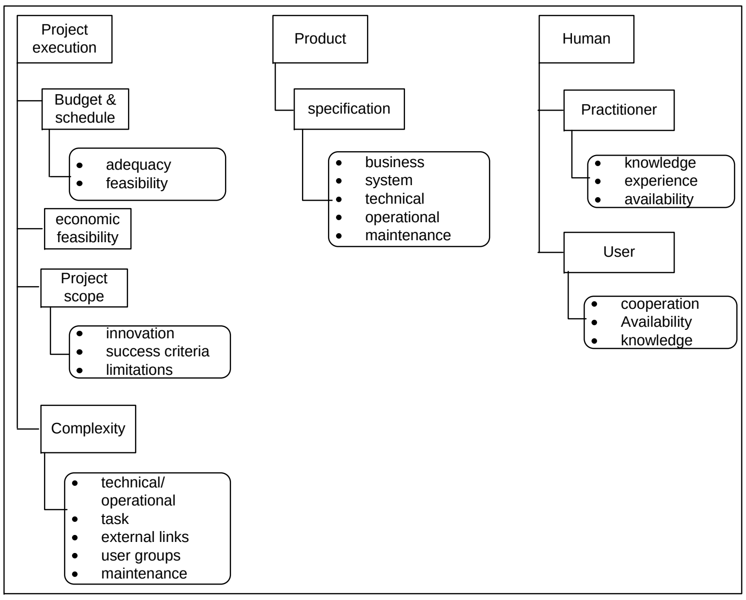


Figure 5.6: Common Project Riskiness Factors

5.3.1.2 Plan Risk Management

The task initiates the implementation of the GSRM during the early development and aligns it with the requirements engineering activities. It focuses on the several parameters that require confirming from the project manager and main project stakeholders including risk management scope, generic risk events and associated thresholds, schedule and resource, risk treatment and monitor strategy. The primary project artifacts such as business goals, project authorization documents (e.g., system vision, project plan, budget and schedule related information) and project riskiness nature are required to define the risk initialization parameters. We assume, at early stage, when the project is initiated then the business goals and project scope are already defined and the system vision is ready for the customer approval. It would be effective, at this stage, to define and agree the risk management scope and to assign responsibilities, authority and schedule for the risk management. The scope shall also define objectives along with the boundaries for risk management in particular which risks may be ruled out from the running project. Furthermore, when information needs is going to be changed such as change of business processes, goals, main project stakeholders (e.g., sponsor, management and development team members), scope, project execution constrains and the organization structure then this task can be repeated. On the other hand, if necessary, the plan also needs to be revised due to significant changes of the overall risk level in the project or wrong initial assumption about the risk scope or project goals. In particular, the risk management scope, budget, schedule, role allocation and risk threshold need revision accordingly. This activity is imperative for any project situation because depending on the project context, organization may not pay attention to consider any formal risk management practice. As some researches have already concluded risk management is not well applied in practice [Rop99, Pfl00]. Our studies results also agreed with this conclusion [IHMFJ09, IH10]. Therefore, the plan risk management task enforces the development team to consider a complete risk management process during requirements engineering so that risk assessment and management are formally integrated during the development. Once the plan has been agreed on then the remaining risk management activities continue for the project. A complete risk management plan is the main artifact concept from this activity.

Artifact: Risk Management Plan The artifact contains a set of attributes as shown in Figure 5.7 to represent risk management plan.

- **Scope:** The scope of risk management, during the early stage w.r.t. the running project in particular the main development components and boundaries which are covered by the risk management. This scope eases to determine the goals for the risk management.
- **Project riskiness:** This value is obtained from the previous tasks in range high, medium and low.
- **Source:** The organizational information, project documents, initial artifacts and the participants response are used as source reference documents for the risk management.
- **Risk Management Process:** The selected activities, tasks and techniques used for the GSRM and associate artifacts produced by the GSRM. If required, depending on the project context, certain tasks or artifacts can be tailored, to support effective risk management activities into the running project. For instance, risk monitoring can be done rarely if the project suffers from schedule problems.
- **Risk Acceptance:** The acceptance defines the threshold values of the critical software development risks, in particular how much percentage of the risk event likelihood or its consequence can be accepted for the running project. The values should also be agreed on with the project stakeholders. However at early stage sometimes it is difficult to determine the threshold values but once the project grows then it becomes easier to determine the values. If required, risk manager can assume values for the critical risk event at an early stage.
- **Risk Boundary:** Which risks should be ruled out or not considered by the project.
- **Schedule:** This attribute concerns schedule allocation for the goal-driven risk management during the running project. In particular, the time frame esti-

mates and allocates for performing the main risk management activities and for producing the risk specification artifact.

- **Responsibility and Authority:** The persons who are responsible for performing the risk management activities and managing the artifacts. The responsibility also covers the communication on the results ,in particular, the critical risk management information to the project stakeholders.
- **Communication:** This attribute concerns how risk information and risk management would communicate with the management, practitioners and customer/user.

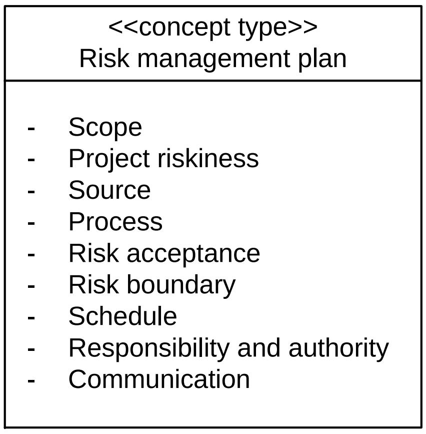


Figure 5.7: Attributes for the Risk Management Plan Concept

5.3.2 Activity 2: Identify & Model Goals

Once the risk management plan is initialized, then the next activity is to identify and to model the goals by focusing on the state of the development components and mapping them to the project success factors. This activity identifies the objectives, expectations and constraints of the component-element-factor hierarchy so that project participants contribute to fulfill these constraints on the way to a successful development path. The activity consists of two different tasks: identify and categorize the goals and model the goals. The identified goals are, if necessary, refined or revised so that they reflect the stakeholders expectation, project success factors, project scope and business goals. The more the goals are elaborated, the better it eases the obstacle identification for any context.

5.3.2.1 Identify & Categorize Goals

The goals are mainly expectations, constraints and problems from the project context. This task identifies and categorizes the goals of the development components. We give a rule set that assists this task and advocate to use it during the risk management.

- **Rule #1 Analyze the input artifacts:** There are several input artifacts of the development components, which are analyzed to identify the goals. The artifacts are: project information such as contract, project scope, development team details, management information, development process information such as existing activities, tasks and methods used for the development, deployment and maintenance, organizational information such as strategies and objectives, policies, structure, corporate environment and development facilities and information related to the product and customer/user. We focus on the details of the each development component and map them to the project success indicators, as stated by the other research [JKD02, PV09]. These success indicators are the desirable outcomes of a software project. Furthermore, customer/user, development team, and management are also interviewed for this task. The identified goals can be of several types, such as:

  - **Information goal:** This goal concerns the availability of adequate information about any element or factor of the development component. For instance, a practitioner requires adequate domain knowledge while participating in a project and the running project domain should provide detailed information about its specification. Adequate knowledge of practitioners or users and detailed information about the project domain are considered as project success indicators.
  - **Satisfaction goal:** This goal concerns the fulfillment of any objectives, constraints and expectations within the project. For instance, customers would expect that the developed software satisfies their business needs and the product meets the desirable quality. Therefore satisfy business need and desired quality are the success factors for the product quality.
  - **Maintain goal:** This goal concerns the retainment of certain desirable properties of any elements and factors throughout the development. For instance, budget as an important element of the project execution always requires maintaining the estimated ones.
  - **Improve goal:** This goal concerns the enrichment of any desirable characteristic of any development component properties. For instance, user active participation is always desirable, therefore their active participation, as success factors, must be improved during the development. Improving adequacy of tasks and methods is essential for the development.
  - **Reduce goal:** This goal concerns the reduction of undesirable characteristics of any element and factor. For instance, errors of the elicited requirements are always common in the development. Therefore goal should be to reduce errors in the requirements.
  - **Product quality factors goal:** It explicitly considers quality factors of the development, e.g., goals relating to security, safety and usability of the product.

- **Rule #2 Categorize the initial goal:** The goals are categorized by following the hierarchy of the development components. Thus at the top level, goals relate with the single development components (i.e., project execution, process, product, human and environment) and categorize to elements and factors. The component-element-factor hierarchy eases the goal categorization and allows to refine the top goals to sub-goals or sub-sub-goals. Once the goals are categorized by the components then additional classification is included based on the goal types (i.e. as stated in rule #1). Goals can be *behavioural* such as *maintain certain desirable property* of the development component which represents in clear cut sense [vL09]. These goals express the system behaviour declaratively, which should be satisfied or maintained throughout the development. On the contrary, there are also *soft goals* that propose the desirable characteristics of the development components from several alternative options, such as improve target condition, reduce target quality or increase target quality [vL09]. Therefore, soft goals can not state the behaviour in a strict sense like behavioural goals. They can state to improve the desirable property or to reduce the undesirable property. Goals can also be functional (e.g., information, satisfaction) or non-functional (e.g., product quality factor, milestone and cost) depending on their context. Goal types and category allow to properly specify the goals for the software development risk management.

- **Rule#3 Delimit the identified goals:** Once the goals are identified and categorized, then we need to delimit the number of goals that are relevant to the project and to the risk management scope. The focus is on the goals that are important for effective development and deployment route towards a successful project. Some goals may be commonly related to the project success indicators and development components, but may not be important for the running project context. In projects with tight budget and schedule pressure, it is always hard to consider all goals for the risk management. Business critical projects focus more on the customer's business specific goals. Sometimes same goals can be repeated several times and may not be well understood by different people. Therefore the goals need to be revised or excluded. However, one should take care not to overlook any important goals from the development components within the project context. This rule set, if required, also supports prioritizing the goals related to the project.

5.3.2.2 Construct the Goal Model

The identified goals are required to be linked up with each other, in order to specify their contribution to the completion of the software development activities. Occasionally some of the identified goals are too much abstract which make it hard to understand the specific meaning of the goals. Therefore, initial goals can be refined to provide more concrete meaning, in terms of their contribution to the project success. For instance, managing estimated budget thought development is a critical goal. But stakeholders may feel that the goal is rather abstract level and needs adequate details. The goal requires refinement to sub-goals such as clear milestones, accurate estimation and other related sub-goals, so that they can contribute jointly to achieve the main goal. This task focuses on the refinement of the goal and establishes refinement link from the sub-goal to the parent goal. The refinement, if required, would be represented conversely by specifying goal abstraction through *WHY questions*, so that higher level parent goals can be obtained from the contribution of the sub-goals [vL09]. On the other hand, the refinement can be formulated by asking *HOW question*. In particular, how parent goals can be achieved through the sub-goals. In goal modeling, depending on the context, *AND* or *OR* refinement can be applicable. An AND refinement means higher level goals can only be satisfied by satisfying all lower level sub-goals. An OR refinement means higher level goals can be satisfied by satisfying any of the sub-goals. Sub-goals are complementary for AND refinement and alternative for OR refinement. These two goal refinement mechanisms are different and should not be confused with each other. The finer the goals are refined, the easier it is to identify obstacles and treatments. The model supports contribution link from sub-goal that contributes to the parent goal.

Goal refinement should ideally be complete and consistent. For completeness, all identified sub-goals would be sufficient to satisfy the parent goals. Note that, sometimes a single sub-goal may be sufficient to support the completeness of the parent goal as well as a single sub-goal may contribute for more than one parent goal. This is important in the software development risk management, because

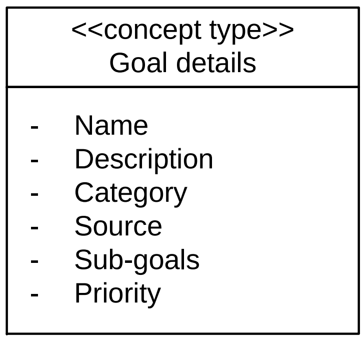


Figure 5.8: Attributes for the Goal Details Concept

such goals are more important from the perspective of project success and require proper protection from the obstacles. Goal refinement should be *bidirectional*: one way is that the higher level goal decomposes into sub-goals and in the other direction a sub-goal contributes to the parent goal. To support consistency, goals would not contradict with each other. Therefore, goal refinement requires adequate domain knowledge about the context. For GSRM, it requires knowledge about the project context, project main success criteria, complexity, inherent risk factors of the project, customer organizational environment, customer/ user detailed and other relevant issues. Sub-goals should not conflict with each other in order to support consistency about their contribution to the parent goal. We use the same notation for representing the goal as stated in the KAOS [vL09].

Finally, the activity constructs a detailed goal specification document that triggers the risk assessment and management activities. The specification provides a precise description of every individual goal that should be focused by the risk management. A goal should not mix up with the operation because goal captures the objective that should satisfy expectation or constraint. An operation is mainly concern to satisfy the goal and specifies solution oriented view to satisfy the goal. Therefore we cannot treat an operation as goal rather as a solution for a goal. The detailed structure (i.e., concept) of the goal artifact content is given below.

Artifact: Goal details This artifact characterizes the concept goal for software development risk management. The goal details precisely specifies the objectives, expectations and constraints of the development components through several attributes, as shown in Figure 5.8

- **Name:** A unique reference name given to each goal, which is used throughout the risk management.
- **Description:** A short overview of the goal in terms of how it contributes to the software development project and maps to the project success indicators.
- **Category:** This attribute classifies the goal as functional and non-functional category and based on the hierarchy of the development component. Additionally, it also includes the specific goal types (i.e., behavioural, soft) to provide more precise meaning.
- **Source:** This attribute denotes the source of the goal origination. The goals may originate from several sources including project stakeholders, initial project artifacts, preliminary company document and so on.

- **Sub-goals:** The sub-goals which are associated with the main goal and their refinement type either AND or OR.
- **Priority:** If required, goals are prioritized, as they are the most critical ones for the successful project. The priority should qualitatively rank goals as very important, important and less important.

Artifact: Goal model The goal model is also constructed by this activity. It is mainly the refinement of the higher goal to the sub-goals through the refinement link. This visual representation allows to represent and understand the goals for risk management.

5.3.3 Activity 3: Identify & Model Obstacles

This activity identifies a list of risk factors that obstruct the identified goals and models them as goal obstruction links. Similar to the goal definition, we follow the development component-element-factor hierarchy to identify and categorize the risk factors. This activity contains two different tasks and details of individual task is given below.

5.3.3.1 Identify & Categorize the Risk Factor

The main obstacles from the early development environment that obstruct the goals and incur problems during and after development are identified and categorized by this task. We need to identify as many likely obstacles as possible so that development team is aware of the problems from the early stage. There are two main desirable properties that an obstacle should comply with: completeness and consistency. *Completeness* specifies whether the identified obstacles are sufficient to obstruct a specific goal. This property allows to consider all possible ways by which the goals can be obstructed. *Consistency* specifies whether any conflict exist among the identified obstacles or not. Same obstacle should not be represented by different name. Thus, obstacle completeness and consistency generally depends upon adequate domain knowledge of the software risk and project context. To support these two properties, we need to categorize all identified obstacles by following the component-element-factor hierarchy. There are several categories of obstacles that obstruct the goals to the software development project. And obstacle category should be influenced by the goal category. E.g. if satisfaction supports a goal then dissatisfaction considers the issues that oppose the goal satisfaction. The identified risks can be large and can represent similar risk factors from the different perspectives. It is essential to categorize the risk factors. We follow our classification of the development component-element-factor that is related to the goals. There are several types of obstacles which facilitate to identify and categorize the risk factors [vL09]. They are:

- **Dissatisfaction:** The obstacle that directly obstructs a goal to satisfy.
- **Unavailable information:** When necessary information about a particular project context, i.e., business process details, interdependencies of the components, is not available or not available on time, then the obstacle is defined as unavailable information.
- **Wrong information:** The provided information is incorrect so that the corresponding assumption is wrong.
- **Incomplete information:** The provided information is incomplete so that the corresponding assumption is not completely up to date.

- **Wrong belief:** The required information of an artifact or the assumption made for project management is different from what it actually should be. Wrong belief can be caused by inability, wrong information and incomplete information.
- **Product quality obstacle:** The obstacles that directly obstruct the product quality such as security by threat, safety by hazard, privacy by harm, usability by unusable and so on.
- **Inability:** An agent such as project stakeholders (i.e.practitioner, management or user), tool and device is not able to perform certain task.

Several techniques are employed for the risk-obstacles identification. However the focus is always on what can be happened for the goal negation or what can go wrong during the life cycle of the product. Obstacle identification reviews project documents such as system vision, project plan and management, goal details, requirement specification, user and practitioners perspectives, user organization environment where the software will be deployed and other relevant documents. Therefore the input elements for goal and risk identification are almost similar but are analyzed from different perspectives. For instance, if the goal is to improve some target property such as *improve customer/user involvement* then the focus is on what might go wrong to depreciate the customer/user involvement. We follow a questionnaire consisting of a set of closed questions, shown in appendix A, to identify the initial set of obstacles. Our checklist was developed by investigating the existing literature about risk identification [CKM+93, Kar95, LWE01, SLKC01], risk factors impact on the software project [WK04, NI09] as well as from our empirical investigation of software development risk management model. The questions are comprehensive and arranged through component-element-factors hierarchy so that one can systematically extracts the status of the development components and links them to the goals. Besides the questionnaires based checklist, we have observed during the case studies that brainstorming session with the development team and project stakeholders is also effective for the obstacle identification. But it consumes more time and some times is difficult to arrange in tight budget and schedule pressure. Closed questions can introduce biasness to identify the actual risk factors. Therefore, combining both of these techniques can effectively contribute to the risk-obstacles identification , in particular, starting with the informal ones, such as brainstorming and workshop and then continuing with the more formal ones such as structure interview through a checklist. Once the risk factors are identified then we focus on modeling the obstacles as goal negation.

5.3.3.2 Construct the Goal-risk Model

Goal-risk model is an extension of the goal model by including obstruction link from obstacles to the goals. Similar to goal refinement, obstacles are also refined through AND or OR refinement. The AND refinement shows how single or multiple sub-obstacles jointly satisfy the parent obstacle and OR refinement shows the alternative ways from the sub-obstacles to satisfy the parent obstacle. Goal-risk model includes the obstacle model that refines the risk factors to risk events. The obstacle model requires identifying risk events from the risk factor so that risk factors are refined to risk events and the identified events also establish obstruction link to the specific goals. The goal-risk model includes refinement of goals to subgoals and refinement of risk factors to risk events. It comprises two different links: a contribution link from sub-goal to the goals and risk factors to risk events and a obstruction link from risk factors and risk events to the goals. This allows specifying the visual representation of the goals and risk factors. The model also shows the critical sub-goals and risk factors which need more attention compared to the others. And this information is critical at any stage of the development.

Software development risk factors are generally followed by OR refinement because it is difficult to claim or measure that all risk factors are jointly responsible for the occurrence of a risk event. However the focus should be to ensure consistency and completeness related to the refinement from the risk factor to the risk event so that one should not skip any critical risk factor. Before constructing the goal-risk model, this task identifies the risk events that are influenced by the identified risk factors from the previous activity. To make the task simple, initial consideration is the risk event which is common for every project such as budget overruns and erroneous requirements and then followed by the project specific risk events. Once all risk events are identified, then the goal-risk model can be constructed.

5.3.4 Activity 4: Assess Risk

This activity quantifies individual risk by estimating the risk level and priority. We start with the causal relationship model by following the identified goals and risk factors from the previous activities. This allows to focus on the relevant risk events for the risk level estimation, rather than considering all raw risk factors. Risk estimation prioritizes the risks so that appropriate treatment actions can be planned and implemented. This activity provides a detailed description of the software development risks w.r.t. the project context.

5.3.4.1 Estimate Risk

Each risk is estimated through two properties: likelihood and severity of impact. However risk estimation is always challenging in software development projects [Boe91]. As stated in Chapter 3, in software engineering domain historical data hardly ever exists in adequate volume for statistically reliable assumption of the risk event, we rely on the subjective probability to estimate the risk level. Probability becomes purely subjective when a true value does not exists [Hou07]. In such a case, the focus is put on the observable values based on observation, individual belief or by focusing on the states of the development components. Quantifying uncertainty in an uncertainty analysis is always conditional. Estimating unconditional uncertainty is always difficult [BC01, FN09]. However the risk factors values' in software engineering domain are fuzzy and that makes it difficult to aggregate these factors into a single instance. For instance, measuring of a risk factor related to unauthorized system access differs from the risk factors related to poor team performance. For an effective integration of risk management during the development, it is necessary to have a simple and straight estimation process.

We use the Bayesian Belief Network (BBN) [Jen96] to estimate the risk by modeling the uncertainties based on the risk factors values and causal relationship of these risk factors with risk events. BBN is formulated on the Bayes formula which eases to model the risk event uncertainty that conditionally depends upon the risk factors. This allows to visualize the software development risk from cause to control via consequence [FN08]. In software development risk assessment, uncertainties involve a wide variety of factors and are surrounded by the development component-element-factor hierarchy. We follow the goals and goal category to elicit a risk event that conditionally depends on the risk factors. A risk factor may influence to single or multiple risk events depending on the context. For instance, inadequate practitioner domain knowledge, passive customer/user involvement, ambiguity terms in requirements are the risk factors that influence for the occurrence of several related risk events like erroneous requirements, schedule-overruns

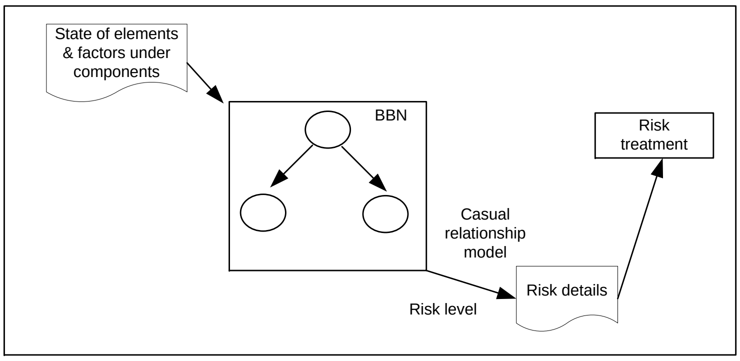

State of elements & factors under components
BBN
Risk treatment
Casual relationship model
Risk details
Risk level

Figure 5.9: Risk Estimation Using BBN

and poor product quality. Therefore causal relationship among the risk factors and events eases to specify the conditional foundation of the critical risk factors at an early development stage.

To estimate the risk we measure the likelihood of risk event occurrence and impact of individual occurrence to the goals. Therefore the result of this task prioritizes the risk so that adequate treatment actions can be identified and employed as early as possible. Figure 5.9 depicts the employment of BBNs for the risk estimation. BBNs develop a causal relationship model so that risk event likelihood is obtained that helps to determine the impact to the goals. The activity consists of two steps: assess risk event likelihood and quantify the risk event consequence.

Step 1 Assess Risk Event Likelihood We consider risk factor, risk event, consequence and risk priority as variables so that they can be mapped with the BBN nodes. The mapping is considered as follows: risk factor as a target node, risk event and consequence as an observable node and risk priority as a decision node. Risk factors are the main causes of any obstacle and controlling of these factors is the initial concern of software development risk management. Risk event and consequence consider as observable node as they are influenced by the identified factors. Finally, the risk consequences prioritize the risk so that the appropriate control actions can be undertaken to mitigate the risk. The nodes represent three possible values from the individual belief or from the response of the closed questions. In BBN, arcs represent the cause-effect relationship among the nodes and support to construct the causal relationship model from the target and intermediate node to the observable or decision node. Depending on the context, the causal relationship can be serial, diverging and converging. If a parent or an intermediate node diverges to several observable nodes then the factor which belongs to the node must get higher priority. On the other hand, when several nodes converge to a single node then the converged nodes are observable or decision node. A decision node responsible to prioritize the risk is concerned with the consequence of the risk event to the goals.

Figure 5.10 shows the mapping of risk factor, event and priority with the BBN's node. However as stated previously, obtaining risk factors and events values is challenging ,in particular, at the early development stage. Some of the variables are unconditionally independent from other nodes value while others are conditionally dependent upon the other node values. Experts' belief or practitioners' perceptions are important to express the dependencies of the variables for estimating the joint probability. Bayesian probability theory allows one to model the dependencies of the uncertainties and foresees their outcomes of interest by combining common-sense knowledge and observation evidence.

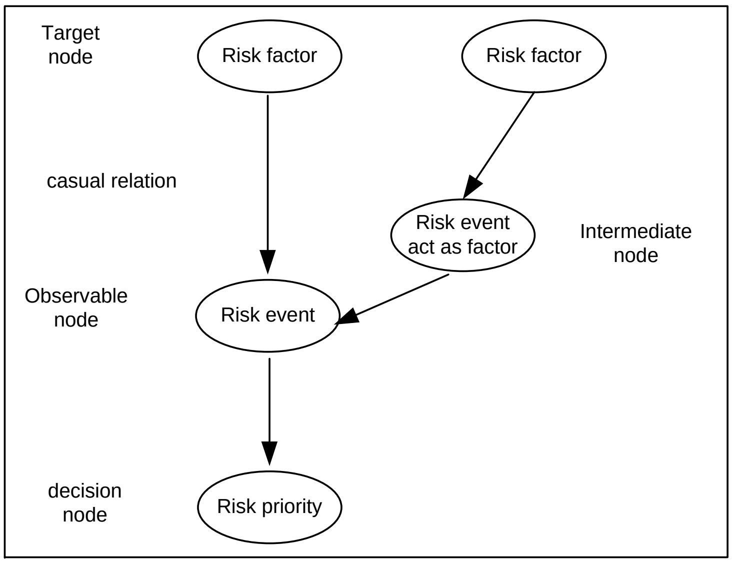
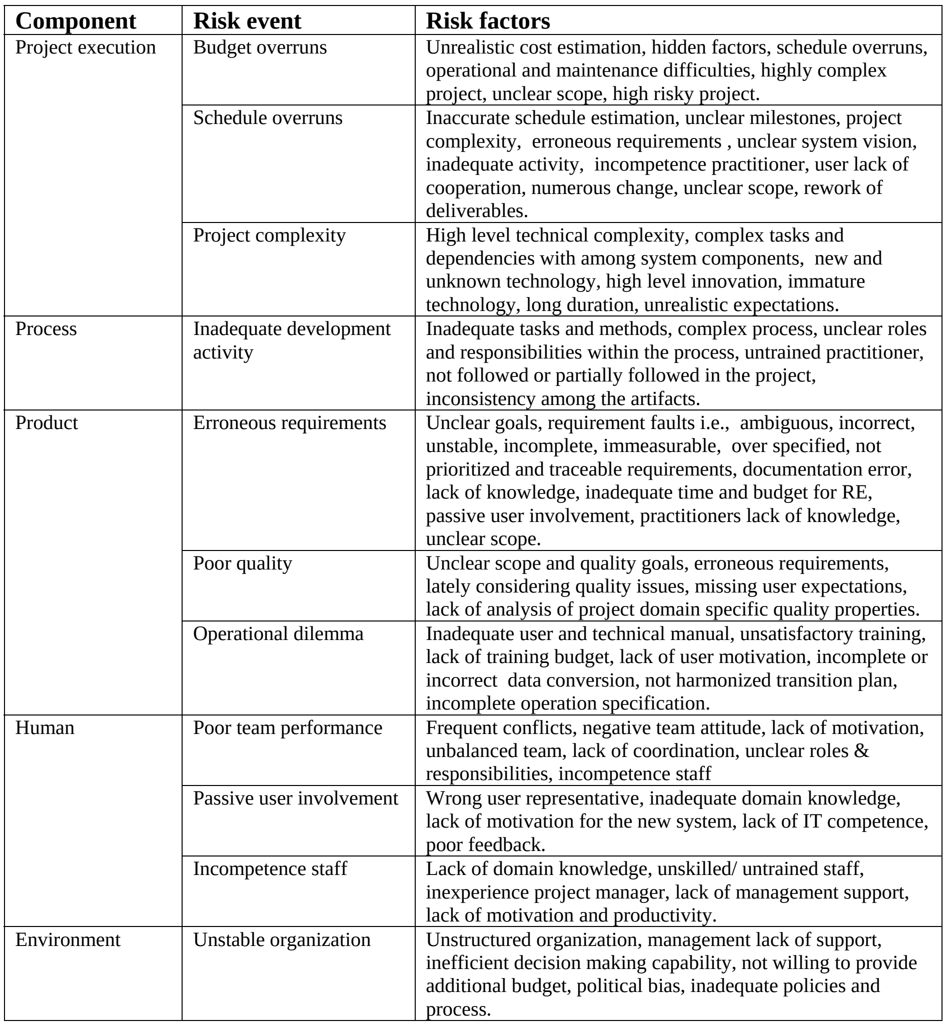


Figure 5.10: Mapping Risk Elements to BBN Nodes

Consider, risk event E occurs under the assumption that risk factor F_1 is responsible for the occurrence of the event. The probability P of the risk event E that depends on risk factor F_1, can be derived as

$P(E|F_1) = P(E, F_1) \div P(F_1)$
The true Bayesian actually considers conditional probability as more basic than joint probability. The conditional probability can be rearranged as
$P(E|F_1)P(F1) = P(E, F_1)$
Symmetrically it can be rearranged as
$P(F_1|E)P(E) = P(E, F_1)$
Therefore it follows that
$P(E|F_1) = P(F_1|E)P(E) \div P(F_1)$
This formula is widely known as Bayes rule [Jen96, Vos00]. It is also possible to derive the risk event probability from single or multiple risk factors, i.e., $F1$ or $F_1 or F_1, F_2....., F_n$ by using the Bayes rule.

We follow the Triangular distribution for assigning the node values, i.e., low, medium and high. The probability distribution of the risk event likelihood is discrete. To determine a risk factors value, we consider the closed questions responses. The questionnaire is arranged into three different scales that are generally used to obtain the project participants' perception about the risk obstacles w.r.t. project context. Once the risk factor values are identified, BBN assists to estimate the risk event likelihood by following the causal relationship from the relevant risk factors to risk event. However, if an event does not contain any causal relationship with the factors then likelihood estimation is simpler compared to an event which is causally linked with several factors. We consider HUGIN tool [Lit] to support the basic computation of BBN for n number of risk factors. The risk event likelihood is estimated within the range [0.0-1.0], with three different scales; maximum



Component	Risk event	Risk factors
Project execution	Budget overruns	Unrealistic cost estimation, hidden factors, schedule overruns, operational and maintenance difficulties, highly complex project, unclear scope, high risky project.
	Schedule overruns	Inaccurate schedule estimation, unclear milestones, project complexity, erroneous requirements , unclear system vision, inadequate activity, incompetence practitioner, user lack of cooperation, numerous change, unclear scope, rework of deliverables.
	Project complexity	High level technical complexity, complex tasks and dependencies with among system components, new and unknown technology, high level innovation, immature technology, long duration, unrealistic expectations.
Process	Inadequate development activity	Inadequate tasks and methods, complex process, unclear roles and responsibilities within the process, untrained practitioner, not followed or partially followed in the project, inconsistency among the artifacts.
Product	Erroneous requirements	Unclear goals, requirement faults i.e., ambiguous, incorrect, unstable, incomplete, immeasurable, over specified, not prioritized and traceable requirements, documentation error, lack of knowledge, inadequate time and budget for RE, passive user involvement, practitioners lack of knowledge, unclear scope.
	Poor quality	Unclear scope and quality goals, erroneous requirements, lately considering quality issues, missing user expectations, lack of analysis of project domain specific quality properties.
	Operational dilemma	Inadequate user and technical manual, unsatisfactory training, lack of training budget, lack of user motivation, incomplete or incorrect data conversion, not harmonized transition plan, incomplete operation specification.
Human	Poor team performance	Frequent conflicts, negative team attitude, lack of motivation, unbalanced team, lack of coordination, unclear roles & responsibilities, incompetence staff
	Passive user involvement	Wrong user representative, inadequate domain knowledge, lack of motivation for the new system, lack of IT competence, poor feedback.
	Incompetence staff	Lack of domain knowledge, unskilled/ untrained staff, inexperience project manager, lack of management support, lack of motivation and productivity.
Environment	Unstable organization	Unstructured organization, management lack of support, inefficient decision making capability, not willing to provide additional budget, political bias, inadequate policies and process.

Figure 5.11: List of Risk Events and Associated Factors

(≥ 0.7), most likely (≥ 0.3) , and improbable (≥ 0.0). The value 0.0 means that the risk event will never be exploited and the value 1.0 means that the event is certain to occur.

To make the whole estimation process simpler, we advocate to initially consider the generic but important risk events of any software development project. The identified risk factors are then used to establish the causal relationship with these risk events so that risk event likelihood can be calculated. An extensive literature review has been performed to gain more insight into the factors which is responsible for the risk event [Lin99, LWE01, RL00, SLKC01, PVOD02, WK04]. We summarized eleven risk events and associated factors, shown in Figure 5.11, by following our literature investigation result. However this summary only provides a guideline for this task and in particular, helps to formulate the risk factors and associated events from the the project participants' observation.

Step 2 Assess Risk Event Impact Once the risk event likelihood is obtained, then the next step is to estimate the impact of individual events to the goals. The

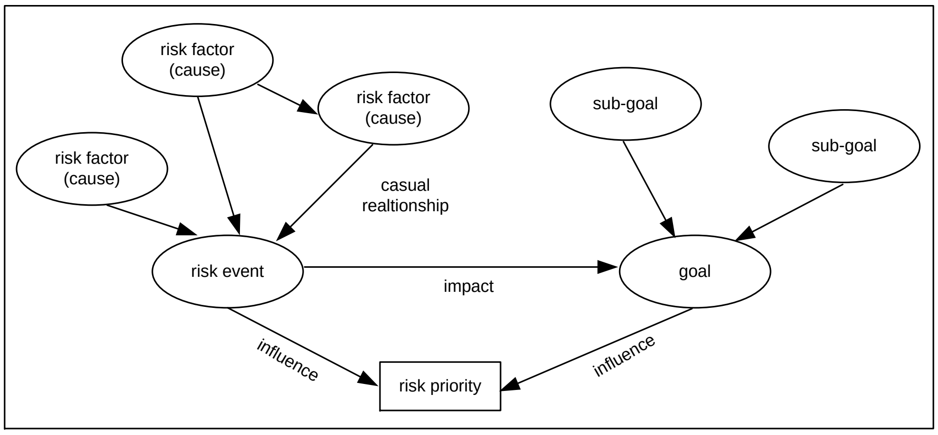


Figure 5.12: Software Development Risk Causes and Consequence Relationship

impact assessment builds the cause-consequence relationship from the risk event to the obstructed goals. We follow the same measurement scale for estimating the impact as used in likelihood estimation, i.e., high, medium and low. It simplifies the whole risk assessment task and makes it effective in software project to prioritize the risk. Figure 5.12 shows the cause-consequence relationship of risk factor and events to the goals and to determine the risk priority.

The impact assessment step, similar to risk event likelihood estimation, is non-trivial ,in particular, when it comes to quantify the risk event consequence. Project context certainly plays an important role for this step. For instance, a project emphasizing goals related to cost and schedule will have different impact compared to a project emphasizing knowledge gathering, user satisfaction, scope or quality. The prioritized goals from the previous activity also supports this task. In the simplest case, initially two outcomes are considered: true and false, by specifying whether the event has negative impact on the goals or not. If yes, then we need to calculate the impact to quantify its severity. However, GSRM considers the risks that have mainly negative consequence to the goals, i.e., goals related to the project success. Therefore, once the risk event likelihood is above its threshold level, then the step focuses on calculating the risk event impact. In the literature, several contributions outline the software risk effect on the project outcome [FHK+01, WK04, IJH09]. We analyzed the literature and came up with the common understanding about the risk event impact to several goals which are relevant for a project. We consider a rule of thumb consisting of 4 rules to support the impact estimation. They are as follows:

- **Rule 1** Initially, the analysis considers the obstruction link from the risk event to the goals. For instance, the important goals for any project context are: maintain estimated budget and schedule, reduce erroneous requirements, improve product quality, effective development activities, staff competency, active user participation and effective communication. However the focus should also be on the project specific goals and associated sub-goals that were already identified. Once the risk events are linked to the goals, we need to consider the most influential risk factors associated with more than one risk events. The goal-risk model allows understanding the mapping and associated dependencies of goals and obstacles. If a risk event obstructs a specific goal and factors associated to the event obstruct sub-goals, then the obstruc-

Event	**Goal**						
	Budget	Schedule	Requ.	Quality	Activities	StaffCom.	Eff. Comm
Budget Overruns	High	High	Medium	Medium	Medium/ Low	Medium /Low	Low
Schedule Overruns	High	High	Medium	Medium	Medium	Low	Low
Project Complexity	High	High	High	High	Medium/ Low	----	----
Inadequate Activity	Medium	Medium	Medium	Medium	High	Low	Medium
Erroneous Req.	High	High	High	High	----	----	----
Poor Quality	High	High	----	High	----	----	----
Poor Team Performance	Medium	Medium	Medium	Medium	-----	----	High
Passive Customer/User Involvement	Low	Low	High	Medium	Medium	----	High
Inappropriate And Incompetence Staff	High	High	High	High	Medium	High	Medium
Unstable Organisation	Low	Low	Medium	Medium	Low	Medium	Medium

Figure 5.13: Software Development Risk Event Impact Assumptions to Goals

tion is complete. For a complete obstruction, the risk event likelihood and threshold values (if available) are considered by this rule to estimate the impact. If the risk event likelihood goes beyond the threshold limit, then the impact is between medium and high. In many cases high risk event likelihood is a common phenomenon. For instance, schedule overruns is a common problem in software project and the stakeholders must agree on how much the overruns in terms of duration the project will sustain. If schedule overruns exceed the agreed limit then certainly the risk impact is high. As stated previously, risk factors are related with each other and the same risk event can be treated as factor for another risk event. Therefore, a risk event with high impact to a specific goal does not imply similar impact to another goal. We need to calculate separately risk event impacts on several goals. Sometimes, risk event occurrence may have positive impact on a specific goal (e.g., personnel shortfall may reduce costs while delaying schedule). However the model concerns with the risk events that have only negative consequence to the goals.

- **Rule 2** We consider a list of presumptions of events' impact on the goals. Some research results showed, that requirements errors and inappropriate and incompetent staffs are the important risk events which severely impact on the goals related to the budget, schedule and quality. Once the risk event probability is obtained, these presumptions facilitate to specify the risk event consequence to the goals negation. However there is no hard rule about this assumption set, it rather acts as a guideline for the impact estimation in particular when no other evidence exists at hand. Figure 5.13 outlines the list of the presumptions about the risk event impacts on the goals.
- **Rule 3** Several important development factors may be, at least partially, beyond the control of a project manager, such as customer/user involvement, issues on project complexity, hidden factors for cost and schedule overruns, inadequate customer/user knowledge, inherent factors for a high risk project, difficulties to deploy the system in the user's premises, operational and maintenance problems and management commitment. We believe these factors impact more on the goals compared to the factors which are under the project manager control such as practitioners skill and development process. Therefore, a risk event that is strongly influenced by these risk factors has higher negative impact on the goals.
- **Rule 4** The expert opinions are always useful in addition to the stated impact estimation rules. Nevertheless, experts are not always available during the software development project. Project manager or other development

team members with adequate experience generally contribute within this context. Individual perception always alleviates the impact estimation but the perception needs to be correctly match with the reality. For example, if risk event consequence is considered higher than the actual value, then there may be a chance to overspend the time and money on mitigating the risk and vice versa [Sch08]. Underestimation or overestimation of each aspect of the potential factors can bring on the wrong evaluation for the trade off [ID08]. People, by nature, prefer risks that they know or concern individually. Sometimes they pay too much attention to the new and unfamiliar risks and less attention to the common, anonymous and less discussed risk. The risks from some controlled and trusty external sources are generally underestimated. Therefore it should always be taken care of when only expert judgments are used for the impact estimation.

Once the risk events likelihood and impact are obtained, the next task is to prioritize the risks. Thus risk assessment layer finally prioritizes the risks based on the risk event likelihood and their impact to goal dissatisfaction.

5.3.4.2 Prioritize Risk

Risk prioritization identifies the risks that require immediate attention. Sometimes it is not feasible to look at all the identified risks due to budget, schedule or other constraints. Therefore this final task of the risk assessment activity prioritizes the most important software development risks w.r.t. project context. The risk prioritization also considers the same measurement level as used in risk likelihood and impact estimation, i.e., very important, important and less important. We use the risk prioritization matrix by considering the event likelihood and impact values. Traditionally in software development, risk prioritization dimension is based on likelihood and impact values, by having a combination of *high-highs* attract the most attention to rank the risks and immediate select the control action [Ban08]. The prioritization then focuses on the *medium-mediums* combination while *low-lows* might be ignored. However there are other two combinations: *high-lows* and *low-highs*. Risks having low likelihood but high impact are commonly known as extreme events. Extreme events are difficult to handle ,in particular, at an early stage of the development. A risk event with high likelihood and low impact also requires attention so that treatment actions contribute for the likelihood value reduction. High prioritized risks, i.e., top-ten or top-twenty risks, attract the project manager to select immediate control actions. Table 5.1 shows the risk priority matrix based on risk event likelihood and impact. It can be high/medium (H/M) in case of likelihood and impact combinations high-medium and medium-high or can be medium/low (M/L) in case of the combinations medium-low and low-medium. Otherwise, priority is high, low or extreme. Several risks can have the same priority and that makes the same rank value for more than one risk.

Table 5.1: Risk Prioritization Matrix

Event likelihood	Impact		
	high	medium	low
High	H	H/M	Extreme
Medium	H/M	M	M/L
Low	Extreme	M/L	L

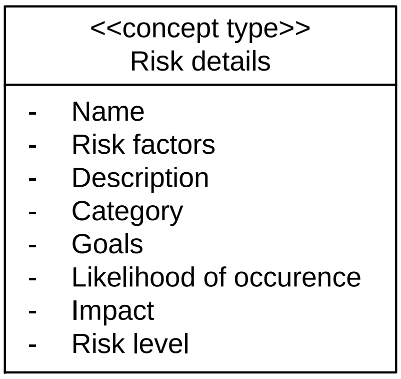


Figure 5.14: Attributes for the Risk Details Concept

Artifact: Risk details At the end of the risk assessment activity, the risk details artifact is compiled. This artifact document details about the concept of software risk. The work product is based on the two activities, i.e., identify and model obstacles and assess risks. It represents the details about the obstacle concept related to the software development project. Several attributes are used to specify the risk details, as shown in Figure 5.14. The attributes are:

- **Name:** A unique reference name of the risk event that provides precise specification of the obstacle which is used throughout the risk management.
- **Risk Factors:** Risk factors are the causes of the risk event. This attribute contains the list of risk factors that are causally linked to the risk event.
- **Description:** A brief description of the risk event,in particular, how risk factors from the development component influence the risk event occurrence and consequence to the goal negation.
- **Category:** The risk should also be categorized based on the development components and by following the obstacle's category such as dissatisfaction, unavailable information, inability, wrong belief and product quality obstacle.
- **Goals:** The single or multiple goals and associated sub-goals that are obstructed by the risk event.
- **Likelihood of occurrence:** Result of the risk event likelihood estimation is represented by one of the three estimation values: maximum, most likely and improbable.
- **Impact to the Goals:** Result of the risk impact estimation is represented by one of the three estimation values: high, medium and low.
- **Risk level:** It is the priority represented by one of the three values: very important, important and less important.

5.3.5 Activity 5: Treat & Monitor Risk

This is the final activity of GSRM. It controls the risk as early as possible and monitors the effectiveness of control actions. It also identifies any new risk at later phases of the project. This activity identifies the potential possible countermeasures and selects the most appropriate ones to mitigate the risk. It is a continuous activity during the development and consists of three tasks.

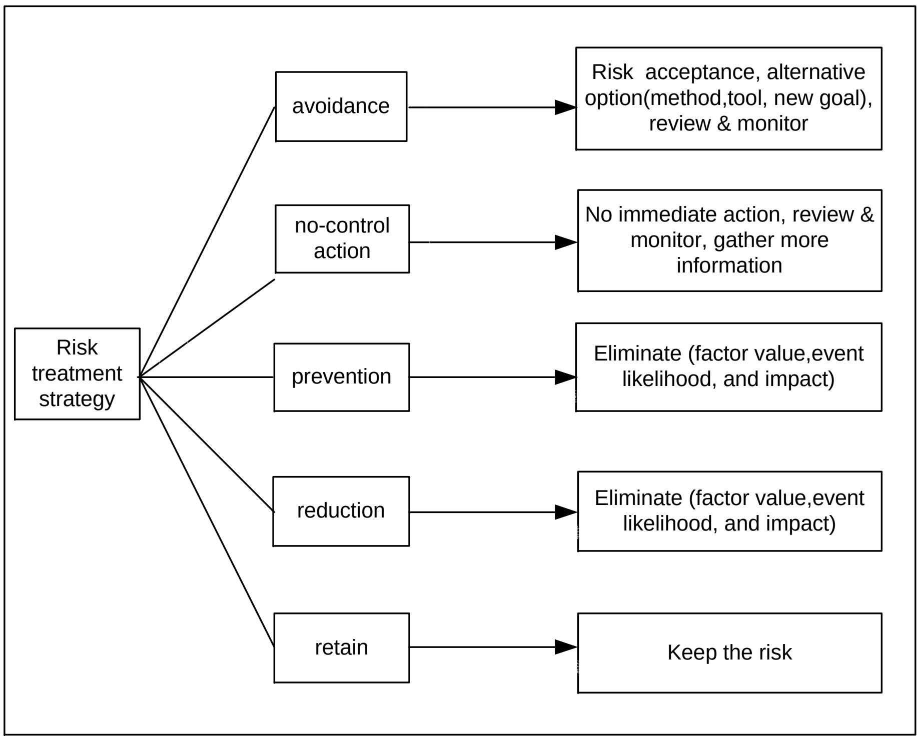

Risk treatment strategy
avoidance
Risk acceptance, alternative option(method,tool, new goal), review & monitor
no-control action
No immediate action, review & monitor, gather more information
prevention
Eliminate (factor value,event likelihood, and impact)
reduction
Eliminate (factor value,event likelihood, and impact)
retain
Keep the risk

Figure 5.15: Risk Control Strategy

5.3.5.1 Plan Risk Treatment

The task mainly determines the possible control actions which are relevant to control the risk event and selects the most appropriate ones so that the control action can be immediately implemented. It consists of two main steps: identify the possible risk controlling actions and select the most potential ones. The steps under this task are linked with each other. The selection of the suitable control actions mostly depends on the identified possible countermeasures and overall risk control strategy. Therefore, the first step should be adequate and accurate so that appropriate control action can be selected.

Step 1 Identify possible Countermeasures Once the risks are prioritized, the initial focus is on the high and medium prioritized risks, such as top-ten or top-twenty risks, for identifying possible countermeasures. We propose to synthesize this step with the high level risk control taxonomy [Boe91, Cha99]. Figure 5.15 shows the strategies of the risk control taxonomy facilitates to identify the possible control actions for the software risk. The identified countermeasures on one hand obstruct the obstacles and on the other hand contribute to fulfill goals. However the countermeasures should be consistent and should have the ability to adequately control the risk. It is a bottom up reasoning from the responsible agents to implement the control actions and to trace the obstacle and goal. We provide details on how the strategies support identification of the countermeasures:

- **Risk avoidance** Risk avoidance considers other alternative options, in order to control the risk and the associated factors, so that negative consequences of the risk event to the goals can be avoided. For instance, review a specific development task or method, substitute a practitioner responsible for a specific task, revise goals, change of tools and so on. These alternative options contain some characteristics that contribute to avoid the causes of the risk event. Risk avoidance also considers the amendment of a specific goal if the goal is expensive or unrealistic in the project context.

- **No control action** This option is preferred in several situations such as if adequate information is not available at early stage or getting information is expensive or risk priority is medium or mostly low. It implies, that there is no immediate control action available to reduce the risk. The strategy expects more information during the course of the project and explicitly monitors the risk as no control action is immediately taken.
- **Risk prevention** This strategy prevents the risk by eliminating the likelihood of risk event occurrence or by eliminating the influential consequences to the goals. Nevertheless, pragmatically there is no guarantee that specific countermeasures are capable to completely eliminate the risk and its associated consequences. But theoretically, control actions under the risk prevention focus on removing the risks entirely from the development environment. One way to achieve this is to introduce a new goal to prevent the specific obstacle. For instance, increase budget or include an external consultant to the development.
- **Risk reduction** As stated, complete obstacle elimination is not always possible therefore risk reduction is an alternative option to minimize the risk to a desirable level. Risk reduction focuses on lessening the likelihood of risk event occurrence or impact of the occurrence to the goals. It implies to focus on the relevant risk factors and to reduce those, so that the associated event's likelihood or consequence can reduce. One possible way for this strategy is to allocate extra support to the development such as include more competent staff for a specific development module, increase active customer/user involvement, emphasize on competence user training, allocate backup schedule or improve the existing development process such as improve the activities and tasks or introduce and intensify the tool support for the development and communication or improve training facilities. Alternatively, risks can be transferred to the project stakeholders. E.g., sharing the risk with the customers such as product operational dilemma can be shared with customer. It implies provisional approval of some risks by the customer or project owner. But before the approval, the details of the selected risks need to be clearly communicated with the stakeholders ,in particular, when there is no realistic solution based on the project context. However transferring risk at early stage of the development components is not generally recommended.
- **Risk retain** If any of the above strategies is not suitable for a risk then the risk can retain as the last alternative. It is used when there is no capability exists to control the risk factors. Nevertheless, this situation is rare at an early development state. Our case study results showed, that project managers are mainly concerned at this stage preventing the risk by selecting the appropriate control actions.

These strategies guide the selection of the possible countermeasures. However, these strategies are not supposed to define an optimal list of countermeasures. It always requires intensive discussion among the main project participants to select the appropriate ones.

Step 2 Select the most potential Countermeasures We need to select the most potential control action, once the suitable countermeasures are identified for the risk treatment. Every treatment action requires evaluating the schedule, cost, agent availability, project goals, riskiness nature and risk management scope for its implementation. We define those as evaluation criteria for selecting the control action. A project always has to trade-offs among the goals to select the appropriate countermeasure. However, this trade-off also depends on the goals the project emphasizes on. If the goal is only economic benefit then control actions related to budget overruns would get the highest priority or if the project concerns more on customer satisfaction for future business gain then control action would focus more on the user satisfaction. The previous step stated several strategies for identifying the possible countermeasures. This step selects the most suitable ones from the identified countermeasures which satisfy the risk management scope. It is often desirable to have an adequate possible countermeasure so that selected control action would be complete. Generally, the initial focus can be to reconsider the risks such that no unnecessary treatment action is assigned and implemented. The potential treatment strategies in terms of prevention, reduction, avoidance, transferring and accepting should come after that.

We consider *the factors review technique* to select the most potential treatment action. It is based on examining the underlying risk factors as the main causes for the occurrence of the risk event. This technique simply reviews whether the selected single or multiple countermeasures are able or not to eliminate or reduce the risk factors and are able or not to improve the situation caused by the risk factors. It also calculates the number of obstacles resolved by a specific countermeasure. As previously stated, some risk factors are more potential compared to others and have the chance to influence multiple risk events. These factors require special attention in terms of their reduction or complete elimination. This technique also reviews whether the countermeasures are adequate to satisfy the goals. In particular, the selected control actions should be realistic based on the project context in terms of cost and agent availability.

5.3.5.2 Assign Agent Responsibility

Agents are the active components within the software development that perform specific role for the goal satisfaction. Risk treatment actions generate single or multiple tasks and agents are responsible to perform the tasks so that the control actions are implemented. This task identifies the agents, allocates resource, if necessary, assigns the responsibility to implement the control actions. GSRM models the agents and associated responsibilities, capabilities and dependencies with other agents which contribute to the goal satisfaction. The main responsibilities of the agent are to control the risk as early as possible and to monitor the status of the risks throughout the development. A responsibility link is constructed from control action to the agent. In GSRM, we consider different categories of agents. They are:

- **Human agent.** Practitioners, i.e., project manager, risk manager, requirement engineering, management and customer/user, are the human agents that take part during the development. The human agents mainly play a specific role directly or indirectly throughout the life cycle of the product.
- **Technical agent.** It includes both hardware agents such as devices, communication media, network infrastructure and software agents such as tools and languages required during the development, use and maintenance of the software.

- **Development, operation and maintenance agent.** It includes activities, tasks, artifacts, policies and management related to the software development, operation and maintenance.

To support the capabilities and responsibilities, agents should perform certain operations. In particular, human agents monitor the threshold condition of the software development risks once the control actions are implemented. An agent is responsible for a goal satisfaction by preforming certain operations. In software development risk management, humans are the main agents for the implementation of the risk control actions and play central role to interact with other agents. Agents interact with each other for the successful risk management. A responsibility link exists from the control action to the agent to specify the responsibility of the agent for the risk mitigation and goal satisfaction.

5.3.5.3 Monitor Risk throughout Development

The risks in general evolve over time, during the course of development, use and maintenance of the product. Furthermore, new risks can also emerge. A continuous monitoring is essential and effective for analyzing the changes of risks in the software project. This activity is initiated, once the identified risks are analyzed and selected control actions are implemented. It is a continuous task that monitors the status of the identified risks and control actions at a regular interval. Frequent interval for the risk monitor is not always possible for the projects with tight schedule and budget pressure. Risk monitoring also depends on the level of overall project risk. For instance, a high risk project risk monitor needs to be more frequent and intensive. The interval can be agreed on during the risk management plan with the project stakeholders or based on the risk status result. This task also updates the risk status report by providing actual status of risks and control actions. The updated risk status aids in decision making about whether the implemented actions adequately mitigate the risk and how to control the newly identified risks. The task focuses on the individual risk threshold values and critical factors of high risk project and provides accurate information about the prioritized risks. It also updates any special remark about the risks to the risk status report artifact.

Artifact: Risk status report This artifact documents the risk state after implementing the risk control actions. The risk status report provides accurate information about the software development risk. It helps to communicate the risk information to the stakeholders so that in-time decisions about the risk can be taken. This artifact, as shown in Figure 5.16, allows a condense view about the risk state. The attributes are:

- **Risk name:** Risk name of the risk details artifact.
- **Risk priority:** Risk priority of the risk details artifact.
- **Potential countermeasures:** The identified possible countermeasures based on the risk control strategy.
- **Selected countermeasures:** The selected potential single or multiple countermeasures.
- **Responsible agent:** The assigned agent responsible to implement the control action.
- **Time frame:** Agreed scheduled time with the assigned agent in order to implement the selected control action.

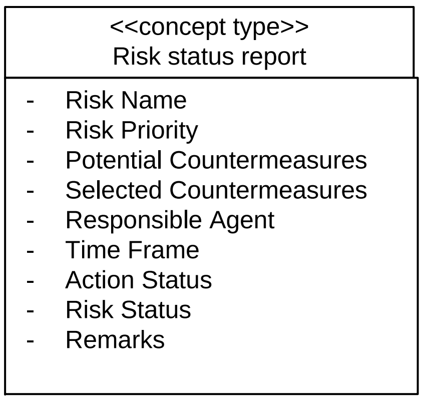


Figure 5.16: Attributes of the Risk Status Report Concept

- **Action status:** Specifies the status of the selected control actions. The attribute has two possible values: initiate and plan.
- **Risk status:** Current position of the risk, once the control action is implemented. The attribute has three possible values: control, prevent and avoid.
- **Remark:** Any special remark about the risk observed by the treat and monitor risk activity.

5.3.6 Artifact Type: Risk Specification

We introduce the artifact type *risk specification* that jointly represents the concepts related to software development risk management. The risk specification reflects the essential needs for managing risks from early stage through deployment to the maintenance of software. The activities of the GSRM in the previous section, subsequently describe the underlying concepts, dependencies among the concepts and corresponding syntax of the risk specification. The artifact type contains *content items* such as goal, obstacle and treatment and *concepts* such as goal details, risk details, risk status report, goal-risk model, causal relationship model and dependencies of the concept. Any concept under the risk specification relies on attributes or properties to define the content of the artifacts. The content and concept allows to distinguish the artifacts' structure and artifacts' underlying concepts [Sch]. Figure 5.17 depicts the content items and encompasses concepts for the risk specification. The attributes of an artifact support relationships to and dependencies on other attributes of another artifact. For instance, selected risk countermeasure of the concept risk status report certainly relies on the risk priority. The concept also requires syntax that defines a concrete language or representation of an artifact. A clear and precise representation is essential to communicate the risk information with the project stakeholders. Highly structured texts in natural language that are arranged in tabular structure, by following the attributes of the individual artifact, are mainly used to represent the concept types. However the models are mainly represented through the graphical representation, for instance GSRM uses goal, obstacle, agent, and task notation from KAOS to construct the goal-risk model. It also follows the BBN to construct the causal relationship model.

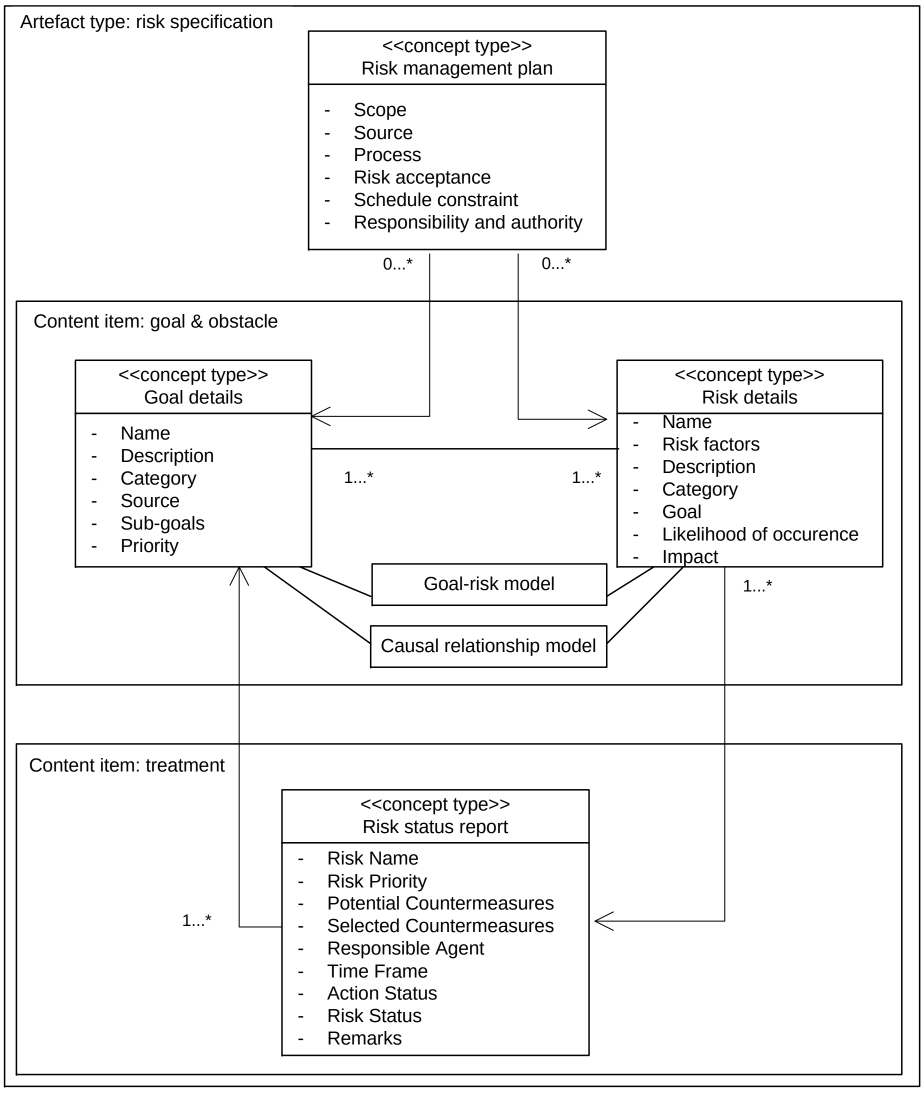


Figure 5.17: Overview of the Risk Specification

5.4 Roles & Responsibilities

GSRM introduces systematic risk management activities explicitly during the requirements engineering phase. The approach requires a clear definition of roles and responsibilities for performing the activities under the process model. Generally, a role takes the responsibility to perform a specific set of activities within the process in order to produce and modify the related artifacts [BWHW05, Fir]. However, there are also roles that indirectly participate within the GSRM process and contribute to or depend on the completion of an artifact, but they have no responsibility for this artifact such as management representative and development team members for our case. The primary role of GSRM is *Risk Manager*.

Risk Manager. The risk manager is the key role for the entire course of the goal-driven software development risk management process. In general, he has the main responsibilities on the creation and maintenance of the risk specification artifact and communicates the risk information with the management, practitioners and user. This role needs special responsibilities to generate precise and timely information about the risk during the early development. The risk manager should further communicate this risk information to the key project members. We summarize the following responsibilities for the risk manager

- Plan and implement risk management during the early stage of the project and continue risk monitor throughout the life cycle of the product.
- Communicate the risk information to the project participants and stakeholders.
- Perform risk management activities and lead collaboration with other participants.
- Manage artifacts within the risk specification.
- Ensure the effectiveness of risk management process and collect feedback to improve the overall process.
- Plan and train the development team about the software development risk management.

There exist additional roles like project manager that directly support the risk management activities. Generally, in software project, roles are differentiated and assigned to the team members, no matter what the project size is. But in small projects, several roles may be performed by a single person. To give a brief overview, further roles that directly or indirectly participate in the risk management can be:

- **Project Owner.** The project owner, also known as project sponsor, controls the resources and supervises the overall project progress and success. He acts on behalf of the management representative and requires the understanding of the importance about the software development risk management during the project. Management support is important for the effective risk management. In particular, project owner has the responsibility to take the initiative concerning the risk management activities during the development.
- **Project Manager.** The project manager is responsible for the overall project execution and makes the risk management occur in the project. Adequate schedule allocation for the risk management activities, when and by whom risk management will take place should be agreed among the management, project manager and project participants. He allocates the training support for the risk management among the project participants. In real situation for a small and medium size project, sometimes the risk manager role cannot be allocated to a dedicated person, due to resource constraints. In that case, project manager performs this role in addition to his primary role. Therefore, project manager should have adequate knowledge and experience for assessing and managing software risks during the development. Even though if the risk manager exists in the project, project manager also needs to actively participate in the risk management activities.
- **Project Participants.** The project participants such as requirements engineer, architect, designer, coder, QA/tester and release manager need to have a basic idea about the software risk management concept, so that they have common awareness about the risks and communicate the potential risk to other team members. This makes an effective risk management practice during the development. The participants also need to be familiar with the activities,

tasks and artifacts involved for the software development risk management. Since GSRM focuses on the integration during requirements engineering, *Requirements Engineers* ought to participate actively in the goal and risk modeling activities. Commonly the requirements engineer takes the responsibility on the creation and maintenance of the artifact type requirements specification. The integration of risk management into RE allows to specify the dependencies between requirement and risk artifacts. Therefore it is effective, if the requirements engineer is actively involved to the risk management activities. This allows on one hand to maintain consistency among the artifacts and on the other hand eases the integration of GSRM within requirements engineering.

- **Users.** Similar to software development, operation and maintenance, users support in risk management is also very effective. Risks in user premises or user perspectives can only be resolved by the user, such as risks related to requirements, product deployment, usage and maintenance. These risks are not under the control of the project manager. It is always important to inform the users about the updated status of risks and to involve them for risk resolution.

5.5 Integration of GSRM into Requirements Engineering

This section outlines the integration of software development risk management within early stage and in particular GSRM during requirements engineering phase. As stated previously, risk management helps to avoid problems, reworks and disasters exist in software project for stimulate successful project outcomes. It should be an inherent component of software project [Boe91] and needs to be considered as early as possible. We advocate to consider it in early requirements engineering phase. However, requirements engineering and software risk management are two different processes. The integration endeavor requires to consider the interactions among the underlying activities, tasks, methods, roles and dependencies among the artifacts that are involved between the two processes. To begin with, we examine two different perspectives, i.e., artifact and process oriented view, that allow to understand the rationale in terms of the integration. We consider several integration points from the artifact and process oriented views. By *Integration point*, we mean anything that explicitly connects the requirements engineering and risk management, i.e., the criteria to integrate between two different processes. For instance, artifact oriented view focuses on the dependencies among the requirements engineering or risk artifact types, its content items and associated concepts [SPHP02, GBB$^+$06, FK09]. The process oriented view also focuses on the interaction and dependencies among activities, tasks, along with the roles and responsibilities of requirements engineering and risk management [Fir, Fir06]. Figure 5.18 depicts the integration points between RE and risk management. This section presents the details about the individual integration points.

5.5.1 Artifact Oriented View

Artifact oriented requirements engineering is a systematic methodology that describes the problem space of the system-as-is as comprehensively as possible towards complete, consistent, correct and rigid requirements specification documents [GBB$^+$06]. It combines both structure and content of the artifacts, expressed by domain-specific concept model [FK09, BFI$^+$09]. It incorporates techniques and notions for

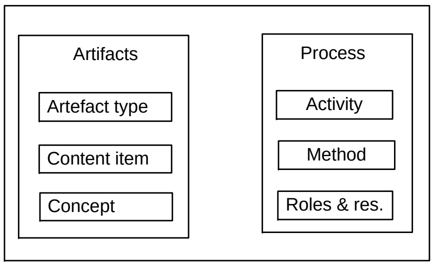


Figure 5.18: Integration Points for Requirements and Risk Management

producing the consistent and complete result of the artifact. The requirement artifact represents prescriptive statements to be enforced solely by the software-to-be. The artifact oriented requirements engineering focusing on the business information system domain mainly covers two artifacts types: business specification and requirements specification [FK09]. Within each artifact type, it contains structure, underlying concept and content dependencies. The concepts correspond to syntax regarding the possibilities for choosing the notation when the artifacts are constructed. Hence syntax specifies the concept through textual or graphical representation. The dependencies among the artifacts can be structured by decomposing a single artifact into several content items without restricting a specific types. Thus dependencies support completeness and consistency among the artifacts, which are critical for the requirement and risk specification. The initial artifacts are, if required, modified or refined, as a part of abstraction and further used by the tasks involved in the later stage of the process. Concept is decomposed into several attributes to specify the concept itself. These attributes are important to represent the requirements because requirements without an attribute are neither controllable nor traceable. We provide a brief overview of the individual requirements artifact types which is the fundamental component for artifact oriented requirements engineering.

The artifact type business specification formulates the goals, capabilities, restrictions and conditions that affect the business of a customer's organization and further information that describes the current and future state of the system. It contains several content items, such as business vision, restrictions and business capabilities [BAB09, FK09]. Every content item in turn consists of a single or multiple concepts. Every concept has its individual attributes supporting content dependencies with other concept attributes. For instance, the content item business vision and restrictions includes steering goals, which have to be achieved by the execution of the business activities and further rules and constraints that have to be preserved during that execution. A business constraint consists of attributes, such as attributes ID, reference, description and areas of validity and a business goal consists of attributes, such as attributes id and statement of intent. Artifact type requirement specification consists of several content items including system vision, information system requirements, integrational requirements and organizational requirements [Wie03, Dav93, SS97]. The system integrational and organizational requirements contain similar concept types which specify constraints of the overall organizational environment where software-to-be will integrate. The requirement concept in turn consists of attributes id, description, owner, stakeholder, source, rationale, acceptance criteria, time constraint and priority [FK09]. Requirements are the demanded properties and constraints of the information system-to-be or the system-to-be-next and its surrounding operational and maintenance environments. For instance, system requirements exclusively demand specific properties of the application, architecture and the technical environment. The content system vision defines how one particular information system will reflect the needs of the business, respectively business processes. It builds the core interface between requirements and business specification. The highly structured natural language text, in tabular or plain list format is mainly used to represent the attributes of the business and requirement artifacts. UML models are commonly used for the diagrammatic representation of use case, scenario, activity and data flow.

Similar to the requirement specification artifact, GSRM also provides, as stated previously, the artifact type risk specification. The risk specification encompasses content item such as goals, obstacles and treatment, which in turn consist of concept types such as risk management plan, goals details, risks details and risk status report through textual representation. The content also supports goal-risk and causal relationship model through graphical representation. GSRM requires visual representation to represent the goal-risk and causal relationship model. This pictorial representation is important to demonstrate the components of the software development risk management model. We consider Microsoft Visio as tool support to model the goals, risk factors and control actions and Hugen tool to model the causal relationships. Figure 5.19 depicts an overview of the artifact types for business, requirements and risk specification. We attempt to focus on the dependencies among the artifacts, through goals and certain attributes of the requirement and risk artifacts. This allows to support the integration of the risk management into requirements engineering.

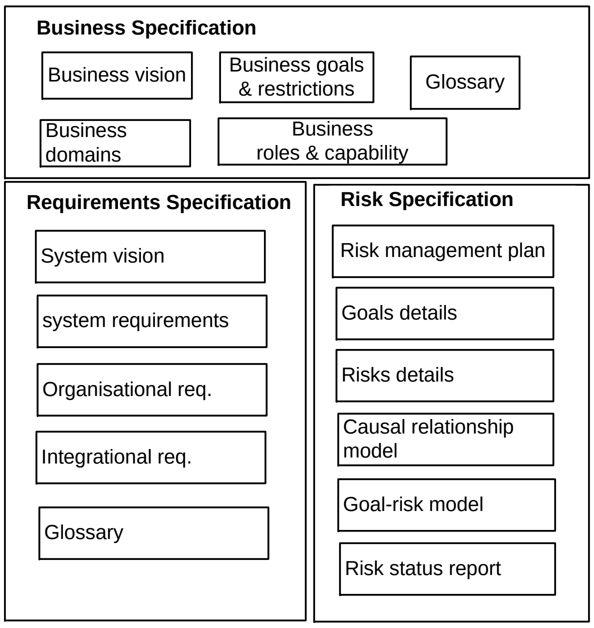


Figure 5.19: Overview on Artefact Types

5.5.1.1 Goal Dependency

Among the requirement and risk concepts, goals are one of the fundamental ones for the requirement and risk specification to support the integration. Thus, goals provide the background foundation to elicit and analyze requirements and risks. A business process or a requirement without a goal is perhaps a statement without any value for the customer. Goals motivate the business activities and finally produce an expected business value. Goals are in several dimensions including business, stakeholders' expectations, constraints, system, project and the surrounding environment of both the system-as-is and the system-to-be. In GSRM, goals are also considered based on the development components from the perspective of project success. For instance, project scope is an element of project execution and success criteria is a critical factor of project scope. GSRM considers complete success criteria as a desirable property, i.e., a goal of the project scope which is needed to be satisfied. This goal also eases to specify the risk management scope. Typical risk management goals such as maintain estimated budget throughout development, adequate quality product, reduce product complexity and perfect deployment contribute for a successful software development project. Therefore goals related to the requirement and risk management are dependent upon each other. Obstacles are the negations of the goals and goal elaboration allows to identify and analyze the obstacles associated with the software project. Hence, goal anchors the risk management. Goal refinement supports reasoning and traceability management of the elicited requirements from the initial input documents as well as from the risk control action to the goal satisfaction and obstacle obstruction. Risk treatment also introduces new goals in terms of reduction, prevention and avoidance of the software development risk. Figure 5.20 outlines the goal dependency among the requirement and risk artifacts. It states, that requirements, risk management plan and goals for risk management are derived from the business goals, business restrictions, project execution and other related artifacts. Thus, goals play a central role in the initial part of system development.

5.5.1.2 Attributes Dependency

The elicited requirement artifacts, in particular, business, user and system requirements support to identify risk factors. In fact, requirements are among one of the elementary inputs for the risk identification. Quality of the elicited requirements is highly influenced to attain the project goals related to schedule, budget, product quality and error free requirements. Reducing project risks is one of the critical requirements of a software project. On the contrary, the purpose of requirements engineering activities mainly desires a complete and robust requirement specification document, where identified requirements should comply with certain qualities. They are: correct, unambiguous, complete, consistent, verifiable, modifiable and traceable according to the IEEE recommended practice for software requirements specification, i.e., IEEE 830-1998 [IEE98]. Defects (i.e.,opposite to these quality properties) of the elicited requirements pose requirement errors, which increase the chance of any undesirable events in the project. Risk concepts contribute to reduce errors from the elicited requirement. Attributes like risk level, priority, control action and risk status help to reduce requirement errors. Commonly in requirements engineering, the elicited requirements are prioritized. Higher priority requirements get immediate attention. Attributes like requirements acceptance, time constraint and design decision are also supported by the risk status report. Requirement attributes, such as description of business service or constraints, statement of the business goal, description, owner, source and rationale of the requirements (i.e., system, integrational and organizational) are analyzed to identify the goals and risk factors for the GSRM. The dependency between requirements and risk artifacts is *bi-directional*: from requirements artifacts to risk artifacts and vice versa. Figure 5.21 shows the interaction between requirement and risk artifact

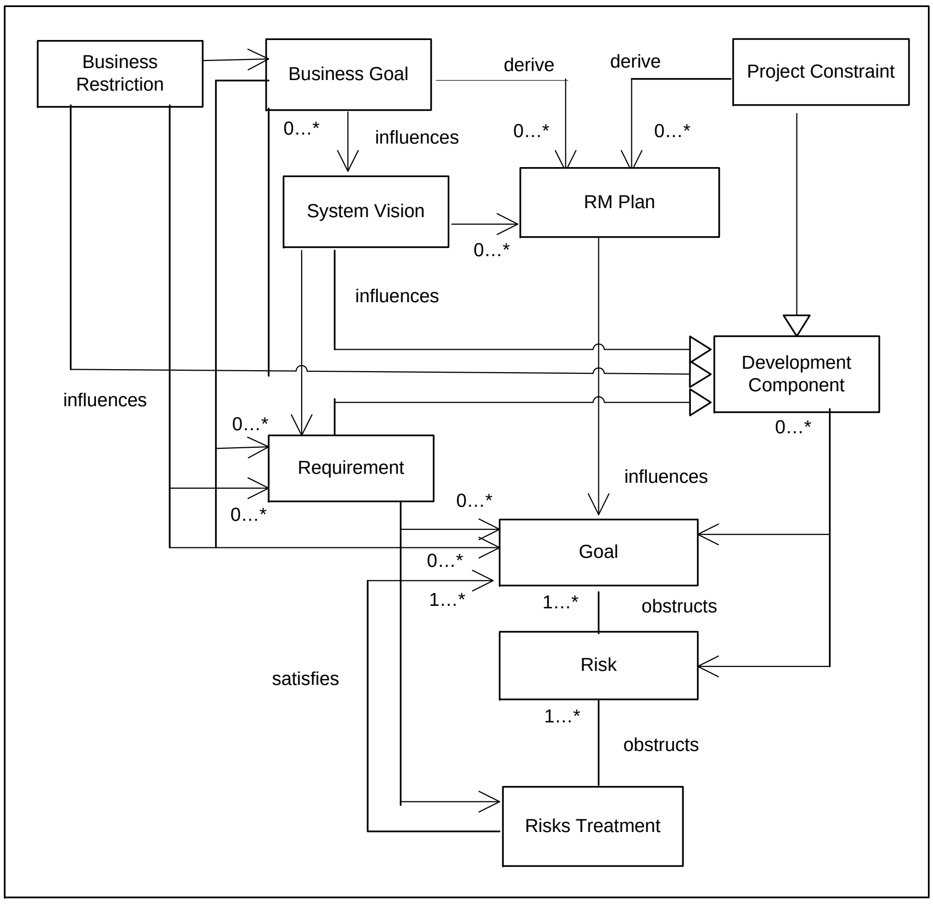


Figure 5.20: Goal Dependencies of Requirements and Risk Artefacts

concept.

5.5.2 Process Oriented View

The activities, tasks and methods of the requirements engineering and risk management process are coherent and interrelated. Both the requirements engineering and risk management processes consist of sequence of activities, tasks and steps in order to construct the artifact concepts.

Requirements engineering is mainly comprised of eliciting, analyzing, validating and managing activities, which further contain fine-grained tasks and sub-tasks within these activities. These activities begin nearly in parallel to the activities of business modeling and usually accompany the whole life cycle. Business modeling mainly focuses on the specification of the business process of a customer's organization upon which in turn the requirement specification builds on. It is concerned with the business specification artifact type that strongly supports the requirement specification. In GSRM, goal based risk management process consists of activities, tasks and steps to support the risk specification artifact type. The activities of both requirements engineering and risk management are sequential and the techniques used within the activities are partially similar. For instance,

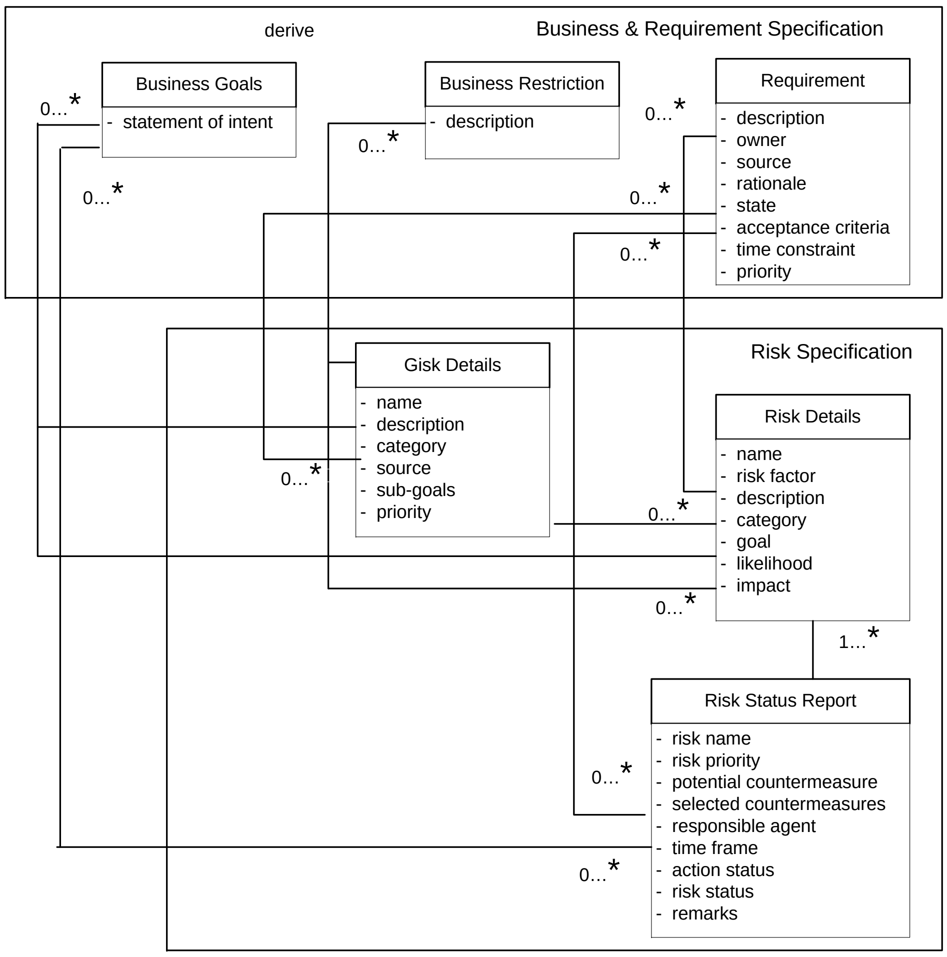


Figure 5.21: Attributes Dependency of Requirements and Risk Artifacts

requirement elicitation commonly relies on the background study of specific type of artifacts, including pre-existing documents about the system as-is, such as organizational charts, policies, work procedure, business rules, data samples and scenario analysis of the interaction among the systems [vL09]. The method also focuses on the stakeholder-driven elicitation through structured and unstructured interview and joint workshop. Goals and risks identification of the GSRM focus on the preliminary analysis of the system-as-is, such as project information, project domain analysis and requirement artifacts. The taxonomy based questionnaires and brainstorming sessions with stakeholders are very effective techniques for the risk identification [CKM+93, FHK+01]. Therefore the techniques used and the input artifacts required for the goal, requirement and risk identification are similar and very much dependent upon each other. The elicited initial requirements are interpreted and refined into the specific expected level of detail, through the requirement analysis. Finally, requirements are validated to comply with the system-to-be along with certain qualities. Risks are analyzed to estimate the risk level through likelihood of occurrence and impact so that the confidence for goal satisfaction can be obtained. Risk level prioritizes the risks that eases to plan, implement and monitor the treatment actions. Outcome of risk analysis also influences prioritizing the requirements and design decisions that are traced with the requirements.

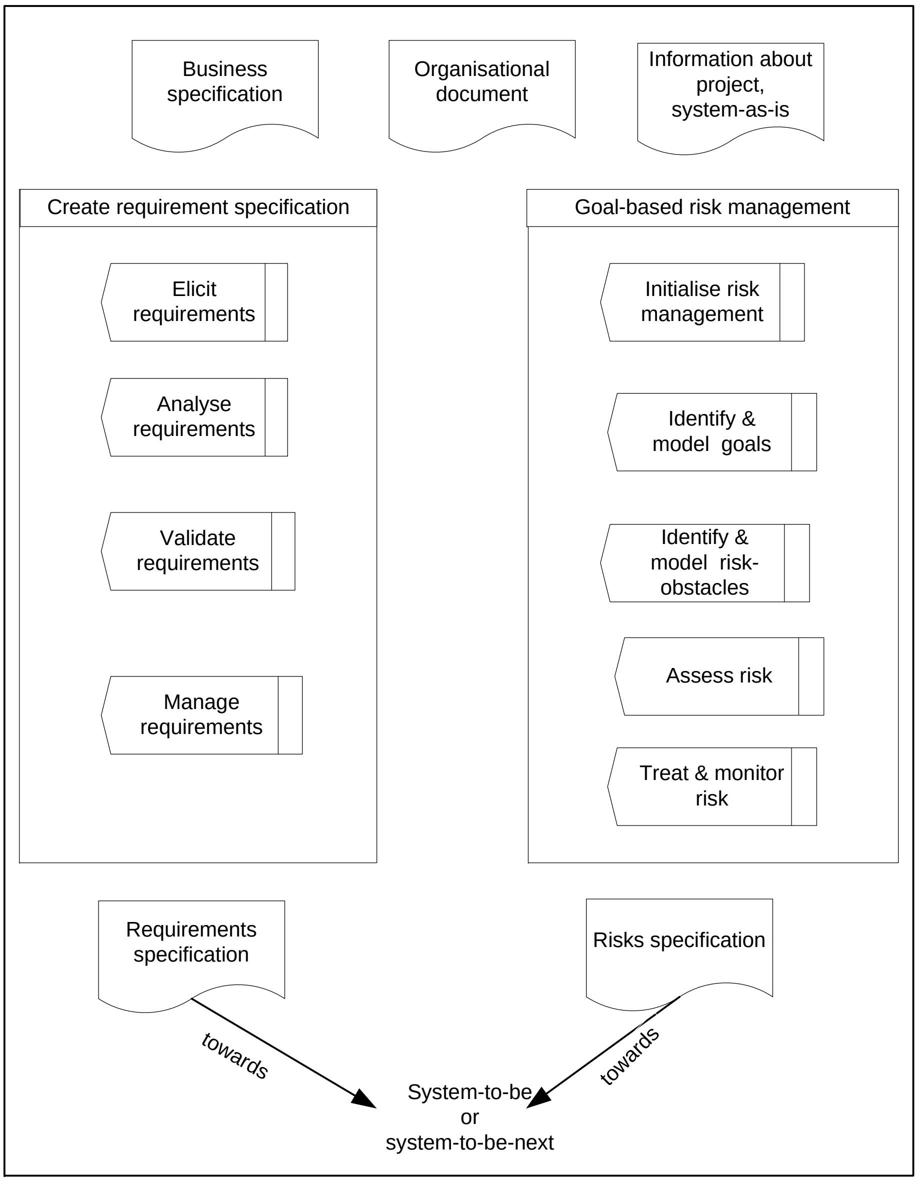

Business specification
Organisational document
Information about project, system-as-is
Create requirement specification
Elicit requirements
Analyse requirements
Validate requirements
Manage requirements
Goal-based risk management
Initialise risk management
Identify & model goals
Identify & model risk-obstacles
Assess risk
Treat & monitor risk
Requirements specification
Risks specification
towards
towards
System-to-be or system-to-be-next

Figure 5.22: Activities of Requirement Engineering and Risk Management

Risk monitoring, similar to requirement validation and management, is a continuous activity. Risk monitoring continues through out the project to identify any new risk and to control the identified risk at an acceptable level. Both requirements engineering and risk management are iterative processes. Note that GSRM focuses on risks not only from the requirements but also from the other development components, from the holistic perspective. Figure 5.22 shows the activities under requirements engineering and GSRM.

Roles Dependency The activities and tasks under the process couple to one or more roles, which take the responsibilities for producing the related artifacts. A role directly or indirectly contributes to the development process, by performing some activities and takes the responsibility for a specific set of artifacts. A clear definition of this role and a balanced assignment of responsibilities to the project participants is critical for any kind of software project. Development process would be ineffective, unless clear definitions and strong coordination among the roles exist. Typical roles for the requirements engineering are customer/user representative, business analyst and requirement engineer. Customer/user representative ,in particular, members of user groups play an important role to elicit the user requirements. Business analysts with particular domain knowledge related to certain business domains such as the financial sector or insurance are responsible for creating and maintaining the business specification. Requirement engineer is the key responsible person, which creates and manages the requirement specification by aligning the business needs to the software-to-be. He establishes the bridge among business analyst, architect, project manager and customer/user. Risk manager is mainly the responsible person for performing risk assessment and management activities. He should have adequate knowledge of the project domain and sufficient skills to handle the risks, in specific project situations. For instance, it may be totally acceptable, to accept a risk without considering any control actions due to limited budget constraint or there is no effective way to counter the risks (i.e. meaning it is not worth the money to invest in treatment). Schedule overruns is a common problem of a software project, depends on its estimated threshold value before considering it as a risk. The risk manager needs adequate experience to estimate the risk level and to select the appropriate countermeasures to control the risk. There exist additional roles like project owner, project manager and project participants (i.e., architect, quality manager). They also directly or indirectly participate in the requirements engineering and risk management process. For instance, a project owner participates as part of management, provides important decisions about allocation of the resources. The practitioner commonly participate during the requirement and risk elicitation activity. We have observed through the case studies [Isl09, IHMFJ09], that a requirement engineer is capable to contribute in risk management activities. In real project situation, for small or medium size project, there may not be any risk manager due to budget constraints [Isl09, IHMFJ09, IH10]. Project manager is concerned with the overall project execution and takes the additional role as risk manager. A successful project manager is always a good risk manager [CH93]. The requirement engineer active participation is more important within this context.

We conclude that requirements and risk artifacts depend upon each other. One of the main focuses of the integration of risk management into requirements engineering is to create and manage a error free, complete and robust requirement specification document and to control the human and organizational issues related to the project success. There are similar techniques used within requirements engineering and risk management activities. GSRM is a goal-driven approach and therefore eases the practical execution of risk assessment and treatment activities within requirements engineering. Goal-driven approach the interdependency between the artifacts and activities of both processes.

5.5.3 Initiation of GSRM into Requirements Engineering

By the term initiation of GSRM into requirements engineering, we mean the starting point of risk management. There is no strict rule, more specifically, it is hard to define any fixed point when risk management starts during development. We advocate to start the goal and risk identification activities of the GSRM nearly in parallel to the requirement elicitation activity. This is because it is beneficial to carry out these activities as part of preparing artifacts such as business vision, business goal and system vision for the customer approval. Goals and risks related to the business needs and project scope can be easily and effectively identified at this stage. Certain goals and risks related to project schedule and budget, staffing, participant competency, customer/user involvement, development facilitates, tools support and project complexity can also be analyzed before the elicitation of the business specification and system vision, i.e., prior to the requirements elicitation. Focusing on these issues at early stage allows to capture non-technical project risks up-front, factors related to the high risk projects even before any requirements have been identified. The identified risks can quickly be countered and later be refined, if it happens to be a persistent difficult problem to address. To effectively tackle risks at an early stage and to reduce errors or wrong requirements, as stated previously, it is important to align the risk management plan with the project scope, business and system vision. It is absolutely crucial to be concrete, despite it being difficult. A concrete project scope and system vision can be used to guide the communication and to consult, monitor and review the activities of the risk management. As a minimum, the risk management plan should define the scope for risk management, schedule and pre-conditions of the risk identification, analysis and evaluation activities and align these with the requirements engineering activities, such as requirements elicitation. GSRM is iterative and can be used to guide the requirements elicitation, analysis and verification activities. The framework is also flexible and can be tailored to the particular project, such that it fits with the project scope, budget and development time frame. Further iterations refine these initial goals and identify and refine associated risk artifacts by mainly focusing on the elements under the product from the elicited user and system requirements and other relevant artifacts. Risks are then analyzed and appropriate control actions are applied to mitigate the risk and thereby attain the goals. This activity circle (identify goals and risks, estimate likelihood and severity and assign treatments) is then repeated until the level of project risks is acceptable, until the project is stopped because of a too low potential success rate, or upon completion of the project.

5.6 Conclusion of GSRM

GSRM combines the goal oriented approach with the software development risk management to model, assess and manage software risks. The underlying process to perform the goal based risk management model is documented in details by this research. We use Microsoft Visio to construct the goal-risk model and Word to create fixed templates of risk specification artifacts and Hugin tool to support the causal relationship model. Our approach systematically introduces goal concepts for risk management and integrates risk management activities into requirements engineering which makes it unique compared to the other works. We start with the goal elaboration related to the project success and every identified goal is shadowed by risks. Therefore, GSRM aligns the risk management process with the project success goals based on the development components. We believe that the goal definition eases risk reorganization and makes the whole process simple. Risk perception enhances the goal clarity. The component-element-factor hierarchy allows us to identify and categorize the goals and software development risks from a holistic perspective and assists to construct the goal-risk model.

CHAPTER 6

Evaluation

Contents

This chapter focuses on the evaluation of the proposed goal-driven risk management model, in particular, assessing the strengths and weaknesses of the GSRM and its integration into requirements engineering. Data is collected and analyzed systematically through a survey and case studies. An informal reasoning is provided about the effectiveness, scalability and usability of the GSRM's practical applicability. The evaluation focuses on the fulfillment of the research questions and practical use of GSRM in the industrial context. Finally, a goal-risk taxonomy proposes based the basic concepts of GSRM and the evaluation results.

6.1 Empirical Evaluation and Data Collection

We have chosen empirical approach to evaluate the main contributions of this research. The demand of empirical studies and their contribution to increasing knowledge in software engineering domain [RH09] is continuously increasing. In software engineering domain, it is difficult to select an appropriate empirical method which is suitable for a specific research context. Because the domain is a multi dimensional discipline which consists of evolving technology, knowledge, intensive human activities, complexities, numerous uncertainties and many other social boundaries [ESSD07]. But the society critically depends on high quality software which demands a systematic investigation of the concepts, methods, technologies and tools used to develop or use the software. Empirical study has well proven to be an effective research method to collect relevant data for investigating a specific issue of the software engineering domain [SDJ07]. We combine two main classes of research methods for evaluating the proposed approach, i.e., survey and case study. Both of these techniques are widely used in software engineering domain [PK01, KPP$^+$02]. This allows to obtain data about the impact of goal-driven risk management on a running project as well as practitioners' perception on GSRM. However, the method selection depends on parameters such as availability of resources (i.e., availability of practitioner and other project stakeholder), access to subjects (i.e., project document, customer business domain, risk management information, users and their organizational information, practitioners information) and opportunity to control the variables of interest (i.e., risk control) [ESSD07]. Our studies confirm these parameters and provide a precise understanding of the evaluation part for this dissertation. We use qualitative data through questionnaires, requirements errors checklist, interviews and brainstorming sessions with the project participants and analyze it in both qualitative and quantitative ways to understand the study result.

Two different set of questionnaires, i.e., closed and open questions are used for the data collection. The closed questions are concerned with the existency and frequency of anticipation and foresight problems in a software project. These questions are arranged by following the component-element-factor hierarchy described in section 4. The closed questions answers consist of three possible values which ease to quantify the state of factor or element under a development component. On the other hand, open questions are mainly descriptive and comparative questions. These questions are mainly used to identify the participants' perception about the software risk management in particular about the usefulness and drawbacks of the goal-driven risk management and its integration into requirements engineering. In the beginning of the case studies, a kick-off workshop was carried out to provide an overview of the GSRM, issues related to the risk specification artifact and its integration into requirements engineering. Furthermore brainstorming sessions were also carried out to identify and model goals and risk factors, risk assessment and treatment and to observe the lessons learned.

The survey and first case study are mainly considered exploratory basis, as they are used for the initial investigations of some key properties of the proposed software development risk management model. A single case study would not justify any conclusion. The final case study is mainly a confirmatory one, which further evaluates the results obtained from the initial studies. Figure 6.1 shows how the results of survey and case studies are related with each other and contribute to the research conclusions. It also shows the context of the individual study. The case study also supports action research because the project data is used as input for GSRM and results from the GSRM employ to improve the overall project situation by controlling the identified risks. This method allows empirical investigation within its real life context [Yin02].

6.1.1 Difficulties for Empirical Study in Software Development Risk Management

Research results showed the barriers or challenges existing to implement the complete risk management process into a software project. For instance, recent survey study result showed that one of the main reasons why risk management is not always applied concerns costs [OG09]. Practitioners do not perceive risk management process seriously and they have commonly inadequate knowledge and experience to perform risk management activities [KS04, NKM08]. We have identified

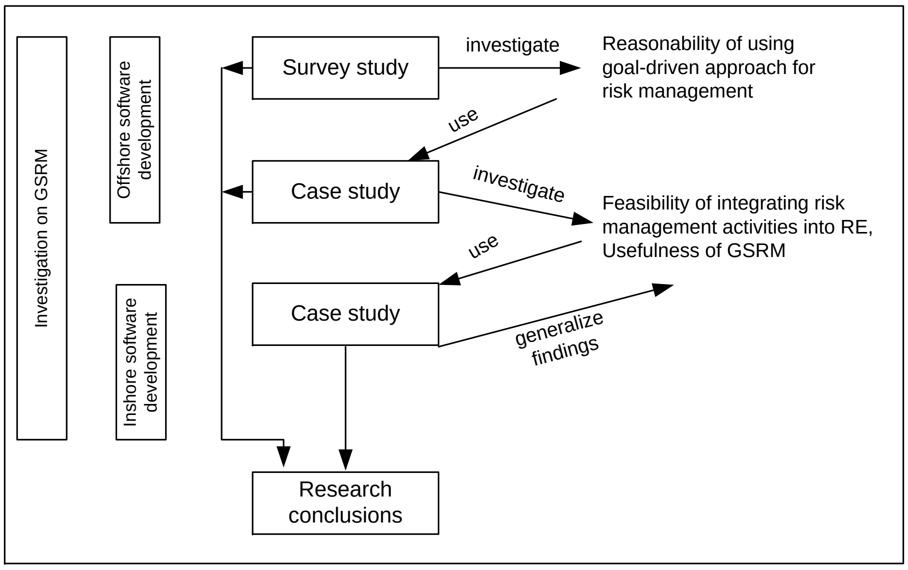

Investigation on GSRM
Offshore software development
Inshore software development
Survey study
Case study
Case study
Research conclusions
investigate
Reasonability of using goal-driven approach for risk management
use
investigate
Feasibility of integrating risk management activities into RE, Usefulness of GSRM
use
generalize findings

Figure 6.1: Empirical Study Methods and Contexts

some fundamental difficulties which pose the challenges for conducting empirical study in software risk management domain. They are:

- Software development projects generally contain a long fixed duration of development life cycle. Projects always pressure to put the focus on time, budgetary and quality control. A comprehensive risk management practice is not always possible. Activities like risk identification and monitoring can be more difficult, in particular, tracking risks across large projects throughout the development. Furthermore, it is difficult to examine software development risk management in an isolated setting within a running project. As a consequence, there exist limited data points to validate the empirical study.
- Project managers and practitioners have lack of motivation to perform risk management. They perceive risk management process and activities as extra work. Risks are the sensitive information of the software project. Lack of practitioners motivation and involvement restrict to reduce the threats related to the validity and reliability of the empirical study result.
- Software risks are subjective by nature. Depending on the individual perception, risk values can be overestimated or underestimated. Every project is unique and evolves due to the change of project scope, market demand, stakeholder expectations and technology. There can be too many risks in various dimensions which make it impossible to control all factors. Risks and its perceived values on past project may not be able to provide accurate estimation in the running project. Lack of empirical data restricts to assess the effectiveness of risk management method into the software development project.

In literature, there exist several survey study results, which identify the risk factors and their impact on project context by interviewing the experienced practitioners. Only a few research focuses on the implementation of risk management method into software development projects. Empirical study results about the effectiveness of integrating risk management activities into the early development phase are rare. Our study results can be compared with the risks or goals of other study results but in terms of evaluating effectiveness of the overall method, we rely on the case studies results of this empirical study.

6.1.2 Study Constructs

We consider three constructs that are relevant for this context and provide the scope of validity of the proposed approach. Construct definition is an important part of any empirical study research, which presents a precise direction of the study and associated measurement [SDJ07]. It provides a specific conceptual representation about a phenomenon. Absence of study construct can lead to improper conclusions about the context. Moreover, in software engineering domain, many studies try to measure phenomenon which are poorly understood. This leads to poor validity of the study result as well as overall quality of the study. However, we try to make the construct narrow and specific, so that the studies systematically analyze the construct and contribute to an unambiguous findings.

The main construct of this evaluation is to specify the impact of the goal-driven risk management model on the software development project, in particular, effectiveness of risk management to attain overall project goals. However, precise measurement of the construct is a difficult undertaking due to the barriers involved in implementing comprehensive risk management activities into the software development project and lack of empirical data in the state of the practice. We consider three main constructs which are relevant for the proposed approach. This makes the study result more precise in terms of the purpose of the empirical evaluation.

- **Goal-driven approach for risk management:** First construct evaluates the characteristics of the goal-driven approach specifically the effectiveness of using goal modeling for software development risk management. It focuses on the input used and artifacts produced for the approach. Scalability of using a goal model is analyzed. The initial survey of this empirical study identified the goals and risk factors associated with a software development project. The case studies results specify the complexity associated with using the goal model for the software development risk management.
- **Overall GSRM method:** Second construct focuses on the overall process model by analyzing the usefulness of the activities, tasks and associated methods used for the goal-driven risk management model. The main motivation of the proposed approach is to develop an easy to use and less overhead risk management practice during the early development activities. It specifies the characteristics related to used input, underlying activities and methods, resulting artifacts as well as overall advantages and limitations of the GSRM.
- **Integration of GSRM into requirements engineering** Third construct focuses on the integration of the goal-driven risk management approach into early software development ,in particular, the explicit integration of GSRM into requirements engineering phase. It focuses on several integration points such as attributes dependency of the requirements and risk artifact, roles and responsibilities, activities and techniques used with the RE and GSRM. These integration points are used and analyzed to measure the explicit integration of the goal-driven risk management model into requirements engineering.

6.1.3 Validity of Study Results

Validity is a critical issue of an empirical study. For instance, case studies are prone to bias. General threats to case studies are related to the difficulties of collecting reliable results and on generalizing the findings. Threats related to the study constructs and internal and external validity affect the output and quality of the study results. However, right quality of the study result needs availability of data, active participation of the interviewer, minimum bias, comprehensive implementation of the method and proper analysis and interpretation of the data. Unfortunately, as stated, it is difficult to obtain the desired level of these factors in software risk management domain. During the course of studies, we tried our best to overcome the threats related to the validity. We examined possible validity threats from the beginning of the study and study data was collected from multiple sources. The studies were systematically planned and implemented.

Our studies are located in a single geographical region which is considered as low cost development software environment. There is a possibility of cultural bias. But we compare the obtained result to the other published results in the literature to generalize our findings. The participants in the survey and case studies include different groups of stakeholders and questionnaires are refined and improved after every study to eliminate any wrong context related to the study objective. We believe the study results contribute to improve ,in general, the software development industry ,in particular, low cost development environment like Bangladesh.

High quality precision of the study result is difficult in software risk management. The main scope is to evaluate the proposed goal-driven risk management model, i.e., methods, inputs, artifacts and its integration in requirements engineering. We believe the scope is concrete and narrow. The survey and case studies participants were experienced practitioners who were involved in software development for at least more than 3 years in Bangladesh. The case study companies are actively involved in both local and offshore software development for at least more than 10 years. The result of the case studies summarizes the complexity associated with constructing and managing the risk specification artifacts. However, the scalability is not fully studied due to the barriers associated to assess the integration of risk management into the development. One case study result provides a credible conclusion about the integration of the proposed method into the requirements engineering. The result shows the practical use of goal-driven risk management into software development project and positive feedback of using the goal-driven approach for the software development risk management. We believe this study results provide usefulness of software development risk management into the project.

6.2 Study 1: Goal and Risk Factors in Offshore Outsourced Software Development

This section presents the survey results [IJH09] of investigating goals related to project success and threats which obstruct the goals in offshore-outsourced software development from Vendors' Viewpoint. This study was conducted using the Delphi survey process. The participants in the survey were experienced practitioners of five software development companies in Bangladesh involved in offshore-outsourced development projects. The main focus of the survey was to identify the goals and risk factors based on actual experiences in offshore projects. The survey context is from a developing country with limited IT infrastructure facilities, but where the offshore market is rapidly expanding through significant increase in investments in recent years. E.g., the European Union has ranked Bangladesh as one of the top 20 outsourcing destinations in the world [Bas].

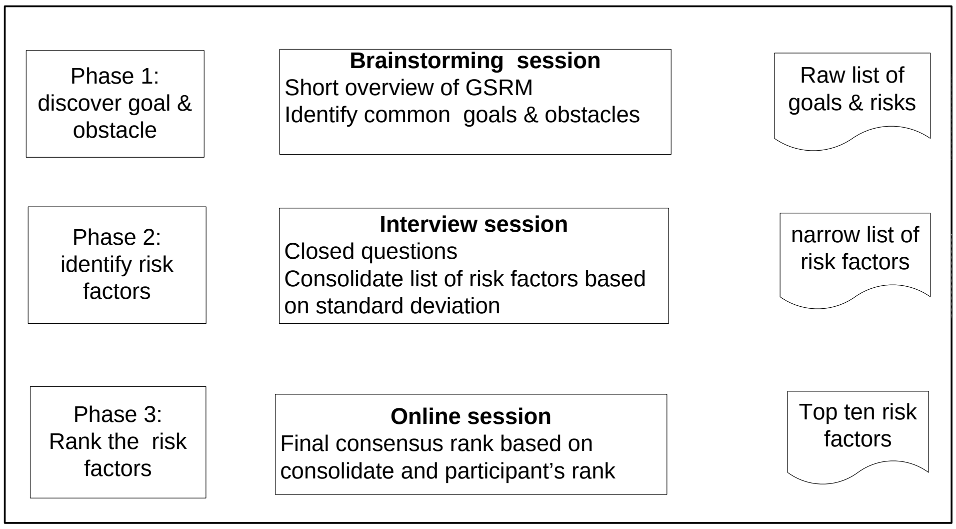

Phase 1: discover goal & obstacle
Brainstorming session
Short overview of GSRM
Identify common goals & obstacles
Raw list of goals & risks
Phase 2: identify risk factors
Interview session
Closed questions
Consolidate list of risk factors based on standard deviation
narrow list of risk factors
Phase 3: Rank the risk factors
Online session
Final consensus rank based on consolidate and participant's rank
Top ten risk factors

Figure 6.2: Steps of Delphi Survey

6.2.1 Survey Method

We provide a brief overview of the survey and associated results. The method followed was a three round Delphi process [Gei95, Sch97]. We have chosen Delphi method because it facilitates multi-phase iterative survey with controlled feedback loops. Here phase 1 involved discovering the goals involved from the perspective of project success and high-level obstacles to these goals. This phase also included collecting background information of the companies. Phase 2 focused on identifying risk factors by refining high-level obstacles identified in phase 1. These risk factors were then ranked in phase 3. Phase 3 was carried using a questionnaire that distributed electronically using e-mail. Figure 6.2 shows an overview of the tasks and goals by the three phases. A total of 15 participants were involved in phase 3. According to the figure, phases are organized through a brainstorming, an interview and an on-line session. Six Master of Information Technology (MIT) students were mainly involved in the survey. The students are properly trained in Delphi method and have adequate knowledge about GSRM. The output of initial phase is used for the next phase. The survey was carried out between November 2008 and January 2009.

6.2.1.1 Phase 1: Discover Goals and Obstacles

In this phase, we used a brainstorming session with open ended questions to elicit the main goals of offshore software development activities from the perspective of project success. Student members directly interacted with the company participants to organize the session. The session also considered any challenges involved which obstruct the identified goals. In addition, a short profile of the vendor companies was outlined. The brainstorming session started by identifying a set of general goals as success factors of the offshore software projects. We also analyzed the goals by following the existing literature [Lin99, PVOD02, vL09] from the perspective of project success. These initial goals were high level and considered as a starting point to discuss offshore outsourcing specific goal elaboration and refinement. In this phase, the participants shared their assumptions and experiences from offshore projects considering what makes the project success. Note that, the brainstorming session was carried out separately at each of the five vendor companies involved in phases 1 and 2. And two student members coordinated the session for each company. At the end of phase 1, the participants within specific vendor company agreed a common set of goals as project success indicators. Table 6.1 gives an overview of the background information gathered on the participating companies

Table 6.1: Brief Overview of Companies

Survey Companies Information	
Company Profile	More than 50 employees for 4 companies & less than 20 for the fifth. Each company has more than three years of experience with offshore software development projects.
Projects	Share of total development activities and coverage: 40% of the projects targeted complete product and full development activities, 35% of the projects involved only coding, 11 % testing and 14 % maintenance.
Survey participants	3 participants inhibiting both roles of director and system developer, 5 participants had the role of project leaders, 4 participants had the role of software engineers, 2 participants had the role of testers, and 1 participant software architect. Average job experience of the participants in offshore project is more than 3 years.

6.2.1.2 Phase 2: Identify Risk Factors

A total of 128 closed questions were prepared for the interview rounds in phase 2 with three possible answers. These questionnaires were based on the feedbacks from phase 1 and aimed at facilitating the refinement of the goals and high-level obstacles. Furthermore, there were 15 open questions aimed at capturing any missing goals, risks or information not addressed in the questionnaire. The open questions response also had the role to reduce the bias of the closed questions, as it gave the participants an opportunity to provide feedback on issues they had not been able to express thus far. The participants were also asked to rank the risk factors internally. The interviews lasted for about two hours per participant. At the end of each interview, the participant was asked to narrow down the risk factors. We produced a consolidated rank of risk factors based on mean rank of individual risk factors and associated standard deviation.

6.2.1.3 Phase 3: Rank Risk Factors

In the final phase, participants were presented by the risk factors with their own rating and the consolidated ranking. They were also asked to reconsider their rankings and to provide a final ranking of the top ten risk factors. We received five responses from five vendors because all participants of the same company jointly selected the top ten risks. This enabled them to reevaluate their previous opinions, so that the result moved toward a prefect consensus [Sch97].

6.2.2 Result of the Survey

The result of the survey showed a high consensus on five of the identified high-level goals: (1) Attain project execution factors, (2) Manage human factors, (3) Manage organizational factors, (4) Attain information security and (5) Compliance with legal issues. These high level initial goals were refined into sub-goals by the participants as part of the information collection activities in phase 1. The goals are internally ranked according to the scale (1-5), where 1 means *highest priority*, by means of their priority in contributing to a successful offshore project. In phase 2, risk factors were identified and aggregated into a consolidated set based on the participants' experience from both successful and failed projects. These risk factors were then ranked according to the quantitative scale (1-10) in phase 3, where the values from 1 to 6 refer to a various degree of *very important* and the values from 6 to 10 refer to a various degree of *important*, in terms of posing a risk to a successful project. The risk identification focused on the factors related to the offshore-outsourced software development environment. Risk factors that are influenced by external parameters, such as local infrastructure facilities and culture, interaction with the client, usually require a longer time to fix. These risks were usually ranked higher than the risk factors that can be addressed locally at the offshore location. At the end of phase 3, the risk factors were linked to the subgoals using a cause-consequence analysis. Table 6.2 shows the identified top ten risk factors and subgoals which they obstruct.

Table 6.2: The Top Ten Risk Factors

Rank	Risk Factor	Sub-goal
1	Lack of client involvement in development activities	Effective client involvement
2	Unstable requirements	Reduce errors from requirements
3	Lack of communication ability and coordination possibilities	Effective communication and coordination
4	Ambiguous requirements	Reduce errors from requirements
5	Lack of domain knowledge of practitioner	Quality and relevance of practitioner
6	Lack of commitment and capability among management	Proper management direction and support
7	Lack of ability for effective change management	Effective development activities
8	Employees not showing up for work	Quality and relevance of practitioner
9	Failure to early identify hidden costs and extra expenses	Stay in budget
10	Failure to consider factors that delay works , i.e., interrupt internet service shortage of power supply	Maintain realistic schedule

To understand Table 6.2, it is necessary to look into the identified sub-goals from the phase 1. The goal *Attain project execution factors*, assigned as the highest priority in phase 1, focusing on the budget and schedule estimation, project complexity, change management, specification, system operation and maintenance from the perspective of project success. The goal was refined into six sub-goals: (1) clear business and system vision, (2) stay in budget, (3) maintain realistic schedule, (4) attain technical feasibility, (5) effective development process and (6) attain product quality. In general, early in the requirements engineering phase, initial elicited development artifacts such as business vision (e.g., business goals, domains, business processes and rules), system vision and system specification(e.g., system, integrational and organizational requirements) require to review for analyzing the goals under this category. Furthermore, factors related to project execution, such as budget and schedule estimation, change management capability and project complexity also influence the ability to attain the sub-goals. The risk ranking and linking to sub-goals in phase 3 identified two very important risk factors: (2) unstable requirements and (4) ambiguous requirements. This makes sense, as clear understanding and freezing of requirements are difficult to achieve in general and particularly in an offshore development context. The reasons are: lack of direct face-to-face communication, lack of knowledge and understanding of the end-users expectations and client's business context and ineffective requirements elicitation with little interaction with the client. (10) Local environmental factors, such as interrupted network service, electricity problem, bank fees and strikes obstruct the staying under budget sub-goal. Although according to the participants experience this risk was not ranked as very important as compared to the other risk factors ranked within 1 to 5. In addition, some of the offshore companies have implemented specific strategies to pull the project despite of the low profit margin. This tendency, along with some unwanted costs (9) obstruct the goal *Stay in budget* as shown in Table 6.2.

Looking at the goal with the second highest priority assigned, *Manage human factors*, this was refined into four sub-goals: (1) quality and relevance of practitioners, (2) effective communication and coordination, (3) proper management direction and support and (4) effective client involvement by mainly focusing on the non-technical issues within an early development environment. As shown in Table 6.2, all these sub-goals were affected by the top ranked risk factors and were thus of great importance for succeeding with an offshore development project. Several survey participants mentioned that there were often lack of involvement and effective communication with the client. The consequence of this was noted to be reworked, conflicts between client and vendor, wrong assumption and incomplete requirements. In fact, all participants stated that they had faced this risk in most of their offshore outsourced development projects and agreed to rank *lack of client involvement* as the top risk factor (1). Furthermore, *lack of communication ability and coordination possibilities* (3) by the vendor was also considered as a risk factor that directly influenced effective communication and coordination. It was revealed that this was mainly due to language and cultural barriers as well as lack of direct (face-to-face) communication. Project members' *lack of domain knowledge* (5) was also a very important risk factor mainly due to incomplete understanding of client business needs. Almost every participant agreed that there is always adequate technical expertise within the vendor site. *Lack of management commitment and capability* (6) due to inadequate experience and professionalism often jeopardize the overall project success and hinder the goal *proper management direction and support*.

The third highest prioritized goal was *Manage organizational factors*. This goal focuses on managerial issues rather than technical issues and can be refined into four sub-goals: (1) effective risk culture, (2) stability of the organization, (3) adequacy of the development facilities and resources and (4) effective policy and procedure. Risk factors that influence meeting this goal are lack of management capability (6) and employees not showing up for work (8) (i.e., see Table 6.2).

In addition to the three above-described high-level goals, two more goals were identified: *Attain information security* and *Compliance with legal issues* in phase 1. However, no direct risk factors were identified for these goals because existence local legal infrastructure does not obstruct the offshore software development environment. Moreover time difference do not hinder the practitioners for the effective communication as they were always work in shifts. However, some participants mentioned that variation of bank fee sometimes create difficulties to transfer money from a foreign country. No participants faced problems with security issues related to exchange of project documents and communication, hence they were not discussed further in this study result.

6.2.3 Survey Study Conclusions

This survey study was performed at the initial phase of this research and focused to understand the practitioners perception about the goal orientation view for the software risk management. The complete model was not evaluated by this study but we indeed identified some important findings about the goal-driven approach for risk management. Based on the response, participants appreciated the integration of goal-driven approach into risk management. It seems reasonable to understand the risk factors which obstruct the goals. And every software project considers generic and project specific goals as the project success indicators. Goals also ease to communicate risk information with the project participants and management. The report identified cultural context as one of the important influential factors for software development risk. Client participation affects more on the offshore software development activities. Most of the identified factors are from the non-technical perspectives, i.e., human and environmental factors. Some risk factors identified by the other studies are not applicable to the local context i.e, time zone difference, legal disputes, loss of control, technical expertise difference and Information security problems [RPJLNA06, AMV06, TSY$^+$07]. As many projects run over budget, reducing production cost is essential. This is particularly true in software development, in which there has been a move from in-house development to global and now also to offshore-outsourced software development. We believe the result aids to better support the emerging offshore outsourced development environment in low cost development environment.

6.3 Study 2: Integration of Risk Management Activities into Requirements Engineering

This section presents the case study result [IH10] of integrating GSRM into the requirements engineering phase of an active on-going software development project. The survey study result from the previous section gives us positive feedback from the experienced software practitioners about using goal-driven approach for software development risk management model. This study focuses on the issues related to the integration of the GSRM into requirements engineering phase. We consider two different perspectives, i.e., artifact and process oriented view, which support the integration points between requirements engineering and GSRM.

6.3.1 Demonstration Integration of GSRM into Requirements Engineering

This section presents an overview of the case study and its results.

6.3.1.1 Study Context

The company was a software development house in Bangladesh established in 1998. From 2003, the company expanded its business strategy to include offshore customers and it has completed a total of 15 offshore projects focusing mainly on the coding (implementation), testing and maintenance phases. At the beginning of 2009, the company started an offshore development project covering all development life cycle phases. Fortunately, as a former part-time employee of the company, I was able to obtain consent from the managing director to perform the risk management activities into the project. Four Master in Information Technology students of University of Dhaka, Bangladesh mainly took part in the case study. They were my former project students and have obtained adequate background knowledge about software risk management and goal modeling language through lectures and tutorials. Moreover, two of them have gained experience by working in three different software projects and the others have sound knowledge of software risk management and requirements engineering.

The development team was on a tight schedule and was not interested in following a detailed tutorial on software development risk management and GSRM. Our team decided to give a high-level overview of GSRM and to take an active part in the risk management activities themselves. The situation is similar to action research but required an even tighter communication with the developers for a successful study. The project concerned the development of a business information system to support the clients core sales business processes. The project focused on the two modules: account and reporting, both were comprised of a number of features, such as bar code readable sales system, inventory and purchase. The existing software on the client side that the project aimed at extending had modules supporting item management and sales. The tasks for the development team were to extend this existing software with accounting and reporting features. The challenging part of the project was transformation of old data and data format into the new modules, which was built on a new platform and to integrate this new platform with the existing hardware, i.e. bar code reader. The project size was estimated to be approximately nine man months with a total duration of eleven months. The development team consisted of a project manager, a requirements engineer, an architect, developers and testers with an estimated duration of ten months. In early development projects at the company, risk management had been performed in an informal way focusing on generic risks without any formal process for risk identification, analysis, treatment and monitoring.

6.3.1.2 Study Objective

The main objective from our side was to analyze the effectiveness of software development risk management during requirements engineering and particularly for GSRM. Note that by the term effectiveness, we refer to the advantages and disadvantages of performing risk management activities during requirements engineering phase. For evaluation purposes, we identified a set of hypothesis to evaluate the observed results. These are:

- **H1** Software development risk management activities can be well integrated with requirements engineering.
- **H2** Goal-driven risk management contributes to manage software development risks by considering a holistic view of both technical and non-technical development components.
- **H3** GSRM effectively reduces the requirements errors.

6.3.1.3 Study Instrument

The project manager initially decided not to follow any formal detailed risk management practice in the project and preferred as many others rather an informal and more ad-hoc approach. However as the management already provided their consent to implement GSRM and project contained several challenging issues, project manager and our team ended up for an action research strategy for the risk management. It combined the planned case study approach with action research to ensure the quality of the risk management part of the project. Project information and artifacts, such as project context, project execution factors, business context, business process, development team, user groups, environment and existing application were analyzed. Our team then obtained feedback from the project participants in general about the integration of risk management activities into the requirements engineering phase and in particular the usefulness of GSRM. The evaluation was performed using a mix of structured interviews, brainstorming sessions and an off-line analysis of the initial project artifacts. The data collection was done in different steps. First step consisted of two different parts: (i) interview with the project team members using our interview template of 200 closed questions and (ii) brainstorming sessions were conducted with the risk management team and other project participants. The interview results were used as input to the brainstorming sessions with the purpose to identify project goals and risk factors. The brainstorming sessions were also used to plan for the risk control actions and their implementation. The final step of the evaluation consisted of 30 open questions given to the interview participants. The goal of this step was to obtain feedback on the integration of risk management in requirements engineering and on GSRM in particular.

6.3.1.4 GSRM Activities and Tasks in the Running Project

Initialize goal-driven risk management The project manager emphasized on controlling risks related to the customer end , i.e., details of customer existing application context, sales process, relevant legislation, customer organizational environment as the main scope of GSRM. Risk management team was comprised of the project manager, student members and two development team members. The project manager agreed to include GSRM into the late phase of requirements engineering. Initially there were two brainstorming sessions scheduled and interviews were planned for the development team members and three members of the offshore user end. However, later on more brainstorming sessions will be added but not planned. The student members were mainly assigned the task to perform the interviews and to analyze the project documents. The project manager took the role as the risk manager and a feedback session was at this point already scheduled to review GSRM.

Identify and Model Goals The project participants identified an initial set of goals related to project success by linking to particular business process, system vision, project execution, interaction with existing software and the user expectations as part of the requirements engineering activities. The risk management team executed an off-line review of the initial project documents to elaborate the goals based on project constraints, process, product, human and the environment. The team completed the goal identification and modeling together with the requirements engineers and one customer representative via a series of conference calls.

Identify and Model Obstacles An interview template with 200 closed questions were used to identify the initial raw risk-obstacles from the project that obstruct the goals. A brainstorming session was also conducted by the risk management team to review the raw risk factors and clustered them into groups according to the hierarchy. At this stage goals and risk factors were modeled and goal-risk models were constructed. Furthermore, goals and risk details were documented to construct the risk specification artifact. The interview template focused on the issues that obstruct the project goals in particular related to budget, schedule, requirements, human factor, project complexity and business specification. Details on questions are presented in appendix A.

Risk Assessment and Treatment Risk levels were estimated by identifying the likelihood of risk event occurrence and impact of the occurrence. Causal relationship model was considered for linking the related risk factors to a single or multiple risk events. I.e., risk factors as target nodes and risk events and consequences as observable nodes. The risk event consequence focused on its negation to the project specific goals. The rule set provided by GSRM was considered within this context. The risks were prioritized and the project manager was initially interested in the risks having risk level between high and medium. Finally, countermeasures were identified and planned to control these risks. Note that, as GSRM focuses on effective use of time and resources, the project manager was more concerned to prevent or reduce risks. Our team focused more on the control actions that can prevent or reduce risks. The selection of appropriate control actions for the prioritized risks also depends upon the availability of specific relevant agent who is responsible for implementing the action. For instance, the agent can be a practitioner who is responsible to implement the risk control action. This means that the project manager role is also important when selecting the suitable risk control action. He knows the status of practitioner and customer/user representatives who can effectively implement the control action. At this stage, our team documented details on the risks and the state of the risk status report. The project manager was assigned the responsibility to monitor the risks throughout the development. However no scheduled plan was undertaken for the risk monitor.

6.3.1.5 Feedback about the effectiveness of GSRM

Regarding the effectiveness of GSRM in requirements engineering, our team used 30 open-ended questions to structurally collect comments from the project practitioners. Apart from the risk management team, requirements engineer and one developer participated in this last feedback-loop. The questions also helped them to form their opinion about GSRM as goal-driven risk management approach in general and its contribution to requirements engineering in particular. The open questions details are presented in Appendix B.

6.3.2 Study Result

There are several findings with respect to the GSRM and its integration in requirements engineering that should be noted:

- **Time and effort of GSRM activities** The activities of GSRM were regarded as systematic and did not incur any extra burden to requirements engineering activities. Around 15% (i.e. 4 person days for 45 days) of the overall project effort is allocated for producing a complete requirement specification. GSRM only consumed 14 % of the estimated requirements engineering effort.

Table 6.3: Agreed Project Goals

Goals
Project Constraints
Improve[RealisticBudgetEstimation]
Maintain[EstimatedBudgetThroughoutDevelopment]
Improve[RealisticScheduleEstimation]
Maintain[EstimateScheduleThroughoutDevelopment]
ClearRolesAndResponsibilitiesAssignment
ClearBusinessGoals
ClearProjectScope
Minimize[TechnicalComplexity]
Process
Improve[AdequacyofTasksAndMethods]
Improve[FormalRiskManagementPractice]
Product
CompleteSalesProcessInformation
Reduce[ErrorFromRequirements]
PreciseIntegrationofExistingData
CompleteSystemSpecification
Human
Improve[CompetencyofTeamMembers]
Improve[Customer/UserParticipation]
Reduce[Customer/UserDissatisfaction]
Improve[OverAllTeamPerformance]
Improve[EffectiveCommunicationAndCoordination]
Improve[ManagementCommitment]
Environment
Improve[StabilityofTheOrganization]
AdequateDevelopmentFacilities

- **Identification of Goals and Risks** There were several goals identified and agreed with the project manager and other practitioners of risk management team. Some of the goals are outlined in Table 6.5. These goals are important and desirable for any software development project. We follow KAOS temporal notation to textually represent the goals.

  Risk factors identified from the project context that directly obstruct the goals are also outlined in Table 6.4. Our team observed that some factors influenced more than one risk event as well as obstructed more than one goal compared to the other risk factors. These factors are important and require extra attention to control as early as possible. Table 6.4 shows the high prioritized risk factors and associated events identified from the project.

  The elicited requirements are one of the main sources for these risk factors. A total of 165 system requirements were identified while performing the risk management activities. Our approach facilitated to identify the errors from the elicited requirements. Our team found that 12 of the requirements were under-specified or ambiguous, 12 were unstable, 8 were incorrect and 5 were technically infeasible. 35 out of the 165, i.e., approximately 22 % of the system requirement were erroneous. There are several causes for these requirement errors, for example the project was inherently complex due to the large number of links among system components and with external customer application, lack of knowledge about the customer business environment and

Table 6.4: Identified Risk Factors and Events

Risk factor	Event
Under-specified,unstable,incorrect and infeasible requirements,	Erroneous requirements,
Incomplete specification,	Operational infeasibility,
Software demands several external links with other part,	Project complexity,
Unclear business process,	Incompetence practitioner,
Practitioner inadequate domain knowledge,	Unclear system vision ,
Local environmental problems,	Ineffective communication,
Missing legislation information,	Passive user involvement,
New development platform.	Customer/User dissatisfaction,
	Budget overruns,
	Schedule overruns.

difficulties to convert existing data to the software-to-be use. There are also other causes, for instance customer/user representatives were not actively involved during the requirements elicitation, information regarding regulatory compliance was partially missing and new development platform was required to support the specific device for the project. Moreover, local environmental factors, such as power shortage and interrupted Internet bandwidth were also creating problems. Risk factors were raised from all development components and consisted of both technical and non-technical issues.

- **Assessment and treatment** The control actions were considered by conducting a brainstorming session by the risk management team. The project manager mainly focused on the issues related to the user representative for resolving these risks. Inadequate knowledge and lack of user participation and cooperation was the most influential risk factors of the project context. These factors obstructed the goals like clear project scope and business process and active user involvement. Initially the focus was to completely prevent the risk, otherwise reduce it as much as possible to satisfy the goals. Unfortunately, due to the inherent nature (i.e. factors beyond the project manager control), not all risks were resolved. Figure 6.3 shows the goal-risk model for the *reduce error from requirements*. It shows lack of participation, inadequate knowledge and missing information are the most influential risk factors which obstruct sub-goals related to the reduce requirements error. Some of the requirements were also unclear for both customer and developer sites. The project manager considered it as being a common situation in offshore projects. However, due to the schedule pressure, these requirements can pose severe problems later on. But no immediate actions were taken in respect to these requirements. It was rather decided to gather more information in particular about the existing applications, component dependencies, business process and legislation context to make the requirements clear and complete. The identified under-specified, unstable, incorrect and infeasible requirements are reviewed further as a part of reducing requirements errors. Goals, system vision and end user expectation were analyzed further. Two requirements were removed due to their technical infeasibility after approval from the user. Out of the 35 requirements, 15 requirement errors were completely solved. The remaining requirements errors were not resolved at that time and it was decided to analysis then further. In addition to the requirements errors, some other risks, such as inadequate knowledge about programming platform were also resolved. E.g., the project manager recommended to assign additionally one or two new members with expertise on

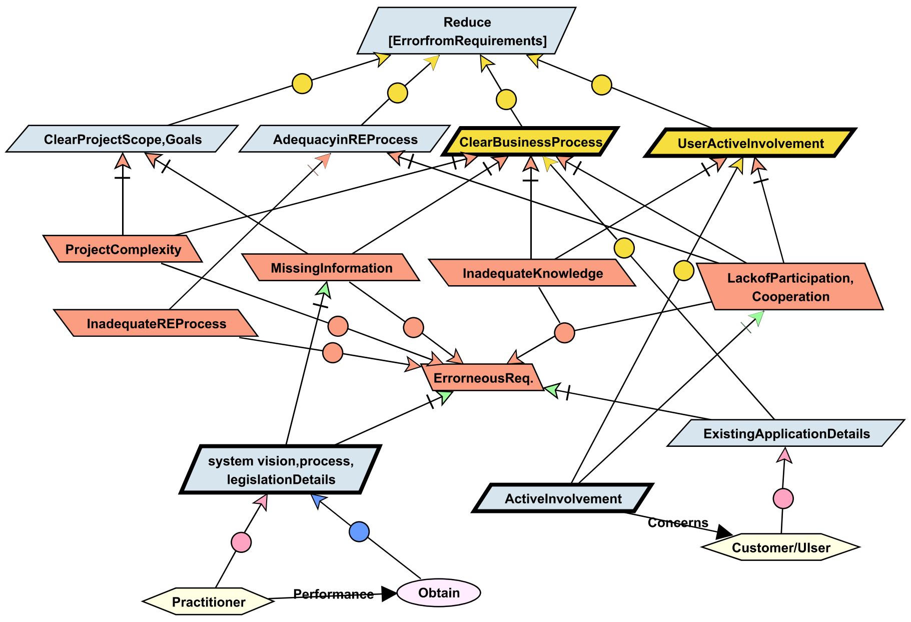


Figure 6.3: Goal-risk-treatment model to Reduce Requirements Errors

the system-to-be required for the project. The interaction links between the existing application and the software-to-be developed needed more investigation so that external link can be properly understood. Active involvement of the key members of user groups and detailed information of the existing applications were required.

6.3.3 Discussion

We made several observations about GSRM from the case study context and these are discussed in the following.

6.3.3.1 Integration of Risk Management into Requirements Engineering

There are indeed strong dependencies among requirements and risk artifacts. In particular, business specification, project goals and system specification(i.e., system, user, organizational and integrational requirements) closely support goal and risk identification activities. Risk management as a part of requirements engineering contributed to produce a complete requirement specification document. It also supported to control the risks related to human and environment, such as practitioner domain knowledge, customer/user participation and adequate development. Activities of GSRM did not introduce any conflicts or significant unnecessary burden with the requirements engineering activities. A project manager with background knowledge and experience in risk management would able to perform the sufficient level of risk management activities. Current project have not any risk manager and project manager successfully performed the risk management activities. Furthermore, the requirements engineers also contributed to the goal and risk identification and later on also to risk control and monitor activities, in particular fixing requirements errors. This is because GSRM is a goal-driven approach which greatly eases the risk management activities and systematically integrates such into requirements engineering. Risk control actions showed that requirements errors can be reduced (i.e. 42 % of the errors were solved) with the support of GSRM, practitioner and user. We observed that risk assessment results help to prioritize requirements so that high important requirements get early attention to the later development phase. We can conclude that the observed results corroborate hypotheses *H1* and *H3*. Moreover, in the evaluation process, we considered risks from both the technical and non-technical perspectives. Thus the result of the case study also confirms hypothesis *H2*.

6.3.3.2 Overall Observation of GSRM from the Case Study

GSRM is a goal-driven approach and eases the practical execution of risk assessment and risk treatment activities. Goals are identified from the project success criteria and by following the component- element-factors hierarchy. On one hand several risk factors may influence multiple risk events. On the other hand the same risk event may have different impacts on different goals. For instance, erroneous requirements obstruct two goals, i.e., reduce errors from requirements and maintain estimate budget throughout development. But risk event impact on the goals always varies based on the nature of the goal. These goals and risk factors were ranked highest priority w.r.t. current project context. Our team also observed that the same risk event can be a risk factor in another context. For instance erroneous requirements as a consequence of requirements faults, such as under-specified, unstable, incorrect and infeasible requirements and further being a risk factor for the schedule or budget overruns. The consequences and causes of a risk event may vary depending on the context.

Further on, there were several points that the participants ,in particular, the project manager and requirements engineer remarked, in addition to those mentioned above:

- The closed questions and the brainstorming sessions are effective techniques for the risk identification.
- Development component-element-factor hierarchy eases to identify and categorize the goals and risk factors.
- Goal refinement is difficult as there may be several sub-goals under one parent goal. Large number of sub-goals can increase the complexity for handling the goals through assessment and treatment. Precise goal quantification is also a difficult undertaking. Goals related to budget,schedule, requirements and human perspectives obtain the highest priority.
- The effort involved in developing risk artifacts, e.g. risk status report and goal-risk model, are in general reasonable. However, if the number of sub-goals increases substantially it will incur extra burden on managing the artifacts in particular for projects with tight schedule and budget pressure.

We treat the last two remarks as limitations of GSRM. The limitations can increase the overall risk management effort in requirements engineering. During the requirements engineering activities, it is also not possible to plan and control all identified risks due to inadequate knowledge of the system-to-be's problem space and uncertainty about the future project execution activities. Additionally, if a project contains many risk factors, then modeling the goal-risk and causal relationship and maintaining the risk status report would consume more time. As we have only considered a single software development project, the data is limited and the validity of the experiences made, as well as its generalization, cannot be concluded upon. This restricts the choice of data points to analyze the results. What we did was to document all information collected from the interviews of both closed and open questions and the brainstorming sessions. The result of the identified risks was compared with published risk factors [SNKT08, IN08, NI09] from similar development environment to augment our limited experienced data. The risk factors and the consequences from the case project coincide with the published survey result of the offshore projects' risk factors . E.g., requirement errors, in particular unstable and incorrect requirements, inadequate project domain knowledge, are also highly ranked by other research results. The local environmental context highly influences the risk factors; we do realize that project risks are culture-dependent [SNKT08], which is also observed in the related research.

6.3.4 Case Study Conclusions

This study implements the proposed approach in a running offshore software project to analyze the effectiveness of GSRM. The results showed that GSRM can be well integrated into requirements engineering phase and effectively contributes to reduce the requirements errors. GSRM is particularly beneficial at the early phase of development because at this stage the project generally focuses on formulating and understanding the core goals and specifications of the system-to-be. GSRM provides warning of the issues which would negatively impact on the project later on. The model also supports in identifying potential risks from both the technical and non-technical development components. The case study context was a developing country with limited IT infrastructure facility. We believe that this type of research contributes positively to the offshore market in the local context which is continuously growing. Further work includes more case studies as well as work towards improving our understanding of integrating risk management into software projects in particular at the early stage. We would also like to review GSRM for further improvement by following the stated observations of the participants within the case study.

6.4 Study 3: Implementation and evaluation of a goal-driven risk management model

The previous survey and case study context is mainly offshore outsourced software development project. This study focuses on a software development project where the customer is a local government ministry. Customer context of this study is different compared to the previous studies. This study shows some unique results however the goal is not to differentiate between local and offshore projects. Here a replicated evaluation of the same risk management method is considered, i.e., GSRM , to generalize our findings in varying context. This also allows to identify any contextual factor important for the software development risk management. This result is the final part of our continuous investigation on the GSRM and its integration into the software projects. The objective of this empirical study is to evaluate the usefulness of goal-driven risk management model from early requirements engineering phase of software project.

6.4.1 Evaluation approach

We have chosen empirical method to evaluate the main contribution of this research. Similar to our previous investigations, here case study method is used to evaluate the GSRM. We are concerned about the practical use of the goal-based risk management model in the industrial context. Software development project is a multidimensional undertaking, validating a single method from a complex set of activities is difficult. Hence, there exist a number of challenges to perform an empirical study in software risk management domain. Firstly, software development projects generally contain a long fixed duration and always pressure to put the focus on time, budgetary and quality control. A comprehensive risk management practice is not always possible. Secondly project manager and practitioners have lack of motivation to perform a comprehensive risk management activities into software development projects. They always perceive risk management as an extra work. Lack of practitioners' motivation and involvement restrict to reduce the validity threats and reliability of the overall study. Finally, every software development project is unique and evolves due to the change of project scope, market demand and technology. The project generally contains various numbers of uncertainties. Project can have too many risks in various dimensions and it is not possible to control all these risks. Risks are subjective by nature and past project risk values may not provide accurate estimation in the running project context. These challenges make it difficult to assess the precise effectiveness of comprehensive risk management method in a software development project.

Nevertheless, despite of these challenges, there exist a limited number of study results such as [JKL98] about the impact of risk management on the overall software project. We previously conducted a case study for evaluating GSRM and its integration into requirements engineering [IH10]. The results provide important conclusions about the impact of GSRM into software development project. Current study focuses to further investigate GSRM by implementing it into an active on-going software development project. It mainly repeats the similar investigation but in a different context. This study treats as exploratory bias because the results from the current study generalize the existing findings within different context. In this study, we combined case study method with action research. Generally an action research makes an effort to provide practical value to the study subject while simultaneously contributing to the acquisition of new theoretical knowledge [ESSD07]. This allows us on the one hand to guide the development team for managing risks and to attain goals during the development and on the other hand to identify ways to improve the GSRM process. For our case, GSRM follows the project documents and artifacts to identify the goals and risks. The risk management results effectively use to control the identified risks of the on-going software project and contribute to meet the project goals in order to improve the overall project situation. Our study combines theory, practice, case study and action research so that in-depth understanding of the GSRM impact into the software projects can be demonstrated.

6.4.2 Evaluation of GSRM

This section outlines the details of our empirical investigation of the proposed goal-driven risk management model.

6.4.2.1 Study Context

Company profile We present the findings of the empirical study result from a software development project at Domain Software Technologies Ltd (Domain Tech), a subsidiary company of Domain Technologies Ltd. (London, UK parent company) and Domain Consulting (M) Sdn. Bhd. (Kuala Lumpur, Malaysia - Sister Company) [Dom]. Domain Tech started its operation at 2002 in Bangladesh with multi dimension lines of business, such as software development, global IT support and consultancy. Since then, several projects were successfully completed for clients located in UK, USA and Malaysia. The company obtained adequate experience in offshore as well as inshore software projects. Currently the company has a total of 72 employees at the Bangladesh site.

Project context The project considered in the case study aimed to automate Planning Commission Campus of ministry of planning, as a part of the e-Governance project of the government of People's Republic of Bangladesh. The project context is mainly development of an application software which contains ten separate modules to automate day to day ministry activities. The modules are: Project Planning, Personnel Management, Payroll Management, Budgeting, Auditing and Accounting, File Tracking, Letter Dispatching, Meeting minutes, Library Management, Inventory and Vehicle Management. Every module has own features, constrains and requirements and relates with other module. For instance, personnel management system mainly manages profiles of all planning commission employees including individual job record, leave management which will help to generate employees transfer order. The features of personal management system are: configuration, staff profile, leave management and reports. Similarly other modules contain its own features, such as payroll management system features are: configuration, salary structure, festival bill, loan/advance, increment and reports and budget system features are: configuration, budget, bill, audit and reports. Two features are common to every module, i.e., configuration and reports. Configuration allows users to customize individual module for its effective use of the software with minimum change. The purpose of the reports is to produce documented output like office order and notice. The main project goals are:

- Automate the whole planning commission campus
- Digitize old and new data and dynamic report generation
- Interconnect with existing software in the Commission campus
- Train employees for the new system

Project riskiness The project was challenging from the Domain Tech point of view due to several reasons. This was the first government project of Domain tech and main users are the government employees. Initially there existed a high level technical specification about the different project modules. The specification was developed by a consultant company on behalf of the ministry. Based on the specification, Domain Tech similar to other vendors submitted an expression of interest and successfully obtained the work order. Apart from the office automation application software-to-be developed, project context also included a comprehensive user training. The development team consisted of 15 members with duration of 13 months. 90% of the users will belong to campus users. A total of 500 users are located in the main campus. The Domain Tech management and main project team members considered the project as high risk even though practitioners have experience to work in similar projects. The reasons are:

- Lack of experience to handle government employees
- high level initial specification
- Large numbers of users training who are government officials
- Effective usage of the product and user satisfaction
- High reputation of the project

Despite the project is considered as high risk project, the management decided to undertake it to accomplish the vision. The management vision was to successfully develop the office application and customize it for the other government ministries and agencies. High reputation and user satisfaction may give Domain Tech a competitive advantage on the market as recently government decided to digitialize its ministry offices under the *Digital Bangladesh Vision 2021* project [Dgb]. The project is important for the Domain tech to capture the market for the future business gains.

6.4.2.2 Study Construct

The main focus of this study is to specify the impact of goal-driven risk management model on the software development project. Study goals are

1. Evaluate the advantages and limitations of goal-driven risk management in software development projects
2. Improve our understanding of the issues involved in integration of risk management activities into requirements engineering

We plan to implement a complete GSRM activities for the evaluation. Project participants' positive and negative observations, risk management results and process integration are mainly used to evaluate the benefits and weaknesses of GSRM. Furthermore, quantitative metrics, such as effort spent on GSRM in requirements engineering, number of risks and treatment actions over time are also considered to support the study goals.

6.4.2.3 Study Plan and Data Collection

I initially requested Mr. Zaheed who is the senior member of Domain Tech development team to evaluate the risk management activities in a software development project. Later on Mr. Zaheed was selected by the management as project manager of this ministry project. He actually helped to obtain consent from the management regarding the integration of GSRM into the project. The management was convinced to integrate a comprehensive risk management practice in the project due to the project inherent risky nature. Figure 6.4 depicts the initial plan for the study. The study will use project specific documents as input and carry out sessions like kick-off and brainstorming as well as interviews to execute the risk management activities. Risk management artifacts are produced as output which will be used to control the identified risk and to attain the project specific goals. Finally feedback about GSRM will obtain to evaluate the method in a subjective way.

We collect data mainly through interview and brainstorming sessions with the project participants, users and sponsor representatives and by analyzing the project documents and development artifacts. Both qualitative and quantitative data are collected and analyzed to meet the study goals. Two different sets of questionnaires, i.e., closed and open questions are used for data collection. Closed questions focus on the existency and frequency of problems and their causal dependencies to the risk events and consequences. The questions are arranged by following the component-element-factor hierarchy and question answers contain three possible values. The open questions are mainly descriptive and comparative questions used to identify the participants' perception about the goal-driven risk management approach and its integration into the requirements engineering. Open questions answer specifies practitioners' beliefs about risk management and its importance to the development. In the beginning, a kick-off workshop was carried out among the development team to provide an overview of GSRM, risk artifacts and its integration into requirements engineering. The open and closed question response and project artifacts were input for the brainstorming session. Two Master of Information Technology (MIT) students of Institute of Information Technology

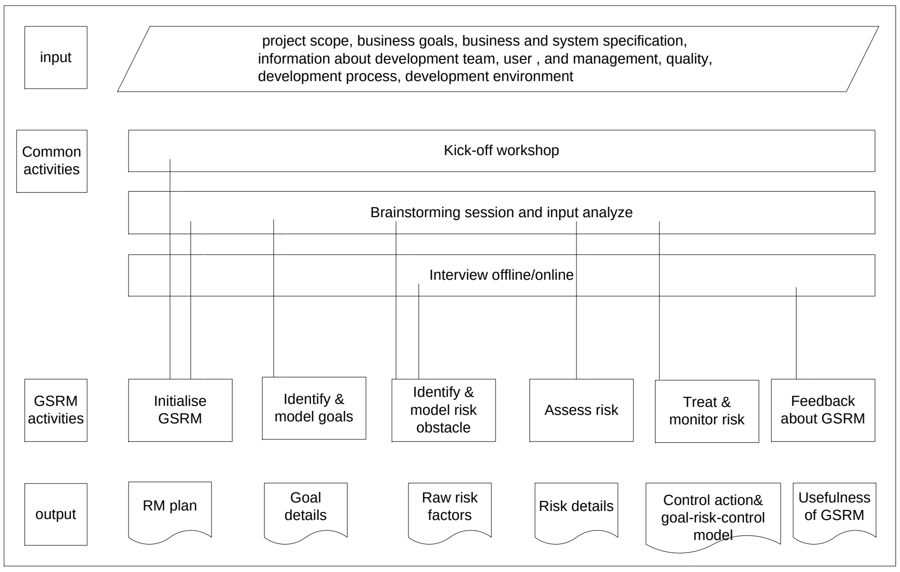

input
project scope, business goals, business and system specification, information about development team, user , and management, quality, development process, development environment
Common activities
Kick-off workshop
Brainstorming session and input analyze
Interview offline/online
GSRM activities
Initialise GSRM
Identify & model goals
Identify & model risk obstacle
Assess risk
Treat & monitor risk
Feedback about GSRM
output
RM plan
Goal details
Raw risk factors
Risk details
Control action& goal-risk-control model
Usefulness of GSRM

Figure 6.4: Case Study Details

(IIT), University of Dhaka, Bangladesh were employed for the implementation of GSRM.

6.4.3 Introduction of GSRM Process

6.4.3.1 Activity:1 Initialize Goal-driven Risk Management

A kick-off workshop by the student members was initially carried out to provide an overview of GSRM. And the final part of the workshop was used for the GSRM initialization activity. The Project Manager (PM) elaborated the factors which brought the project as a risky one. The first task under this activity (i.e., determine the riskiness nature of the project) was already done and main factors for the high risky project were noted. Project scope, success criteria, business goals and initial high level system specification were used to define the risk management context. The risk management scope was to control risks related to the application from a holistic perspective specifically project execution, user, product specification, quality and user training. The risk management scope was biased by the project inherent challenges and scope. However, risks related to project sponsor, fund support and user internal organizational problems were not considered. The Risk Management(RM) team was also approved by considering PM as the main authority, student members and 2 development team members. The PM role was the main authority responsible for the successful implementation of risk management activities and to communicate the results with management and user representatives. The risk management was scheduled at the first deliverable phase, i.e., requirements specification and design phase. The interview participants from both users and sponsor representatives were identified and schedules were also planned. However no schedule was considered for risk monitor and PM decided it as a demand-basis activity.

Table 6.5: Identified Goals and Sub-goals

Goals	Sub-goals
Complete project in estimated budget and schedule	maintain estimated budget in development maintain estimated schedule in development maintain realistic estimation clear milestones competence practitioner reduce errors from requirements user active participation user positive motivation
Complete user's training	adequate training budget professional and competence training complete training and user manual user active participation user positive motivation improve effective communication & coordination improve practitioner motivation& productivity reduce user and practitioner conflict
Obtain positive reputation, Generalize the application	user satisfaction complete project in estimated budget and schedule quality product successful training details understanding of business process successful system usage complete elimination of existing manual system complete specification

6.4.3.2 Activity 2: Identify and Model Goals

This activity was carried out by a brainstorming session among the members of RM team. RM plan is considered as input for this activity. Furthermore, project artifacts, such as project scope and execution, business scope, system specification, human and overall development environment and Domain Tech business vision were also considered to identify and model goals. Four high prioritized high level goals were agreed for the project, they are: *complete project within estimated budget and schedule*, *complete user's training*, *obtain positive reputation*, and *the application for other government ministries*. First two goals were considered by focusing on the project contract and needed to be achieved before the system goes in operation. The remaining goals were critical for the future business vision of Domain Tech. However, it needs user positive feedback, product overall quality and product successful operation and maintenance. High vendor reputation can give a competitive advantage to obtain a work order of a similar government project. Generalizing application by gathering knowledge from this project can significantly reduce the development cost. These goals were refined during the session. Table 6.5 outlines the goals and sub-goals which are agreed by the RM team. The goals are related to each other and same sub-goal links to more than one parent goal.

6.4.3.3 Identify and Model Obstacles

Initially the interviews were carried out by the student members with the selected practitioners of the development team. The interviews continued further with 12 users and 1 sponsor member of UNDP. It took more time than expected with the user members as they did follow the agreed schedule and lack of their project domain and IT competence knowledge. RM team also needed to provide an overview of goals, risks and main purpose of the interview before actual interviews have taken place. 200 closed questions were used for the interview and it took approximately 2 hours for an individual practitioner to answer the question and more than 3 hours and 30 minutes for an individual user. Interview responses were complied to generate a raw list of risk factors. The raw list was very large and was refined by following the identified goals.

Most of the identified risks were related to the project execution, product and human, in particular, from the schedule problems, product specification and user perspectives. The PM said, *"we didn't get the full requirements and missed a lot of things in the initial specification. Our management agreed to accept any user change/update at any stage in the contract and believed it will increase the company reputation, but it was indeed a great mistake. Users cannot provide detailed information and later on we have identified numerous gaps and needed to incorporate all of them"*. Users added several new things compared to the initial specification based on which schedule and effort estimation was calculated. Numerous changes were requested by the users, such as change of staff profile and lay out or addition of more functionalities under a specific module related to personal and payroll management, budgeting, auditing and accounting. The collected information is sometimes ambiguous in a sense that different interpretation of the same context. It was difficult to obtain appointment from the high officials and needed several meetings to collect the relevant information. But their remarks were important for the project acceptance. Some of the users have not adequate knowledge about the software application but played main roles to support the key business process. There were ambiguities in describing the process and roles involved within the process. Once an interview was completed no user agreed to sign the interview documents. At the end of the requirements analysis, it consumed much more time than estimated.

Another problem arose about the number of users who will participate in the training. Initially Domain Tech agreed to train 500 users. However during the requirements identification and on site observation, there were about 800 users identified who needed training for the new system environment. But the planning ministry and UNDP refused to increase the training budget as total project budget was already approved and no room for revision because several donor agencies supported the project fund. Domain Tech also had inadequate experience to handle the government officials. Existing data were hard copy documented and had very dissimilar structure, which made it difficult to convert the old data into new format. Risk factors were summarized from the raw list. These factors were mainly due to the unavailable, wrong and incomplete information, poor participation, errors from the existing system which brought incomplete specification and inaccurate estimation. Table 6.6 shows the risk factors.

6.4.3.4 Assess Risk

RM team considered interview response of the selected risk factors as initial input for the risk assessment. Moreover, PM's and other members' experience and the pre-assumptions provided by GSRM about the causal link of the risk factors to the risk events and consequence to the goals were also considered to estimate

Table 6.6: Identified Risk Factors and Events

Risk factor	**Event**
Numerous change request	Budget overruns
Users passive involvement	Erroneous requirements
Users lack of project domain knowledge	Schedule overruns
User lack of IT competence	Ineffective training
Inadequate training budget	Unclear system vision
Large number of users requiring training	Ineffective communication
High training cost	User dissatisfaction
Incomplete & incorrect initial specification	Poor system use
Over-promise & Inaccurate estimation	Poor reputation
Error in project contract	Incomplete information
Lack of experience of handle government officials	Unable to generalize product
Bureaucracy nature of organization	Poor feedback
User lack of motivation	Deployment problems
Data conversion difficulties	High variation
Unwilling to sign interview document	
Complex interactions among the modules	
Political biasness	

the risk event likelihood and risk impact. Figure 6.6 includes risk events caused by identified risk factors. These events directly obstruct the goals, for instance budget overruns, erroneous requirements, poor training and system use obstruct the goals like complete project in estimated budget and schedule or vendor high reputation. To construct causal relationship model, risk factor, event and risk priority were considered as target, observable and decision nodes of BBN. RM team agreed that budget overrun is the high prioritized risk factor. Risk factors like inaccurate estimation, high training cost, erroneous requirements and schedule over runs casually linked to the budget overruns as shown in Figure 6.5. These factors are the most influential for the project context and some of them were beyond the project manager and project environment control. The identified factors brought a complete obstruction to the goal maintain estimated budget in the development. Erroneous requirements and schedule overruns considered as risk factors but these factors treated as risk event in different context. Finally risk events were prioritized into three scales, i.e., very important, important, and less important, so that severe ones get immediate attention.

6.4.3.5 Treat and Monitor Risk

RM team initially planned to control the high and medium prioritized risks by completing eliminating or reducing likelihood of risk event and associated impacts. The session started to identify the possible countermeasures and selected the potential ones so that the goals could be attained.

Erroneous requirements are one of the main events which negatively consequence to several goals like complete project in estimated budget and schedule, reduce erroneous requirements, improve completeness in requirement specification, quality product and generalize application. Users' factors were the main reasons for the risk events and large number of users increase the total training cost and overall project budget. But these factors were beyond the PM controls and difficult to

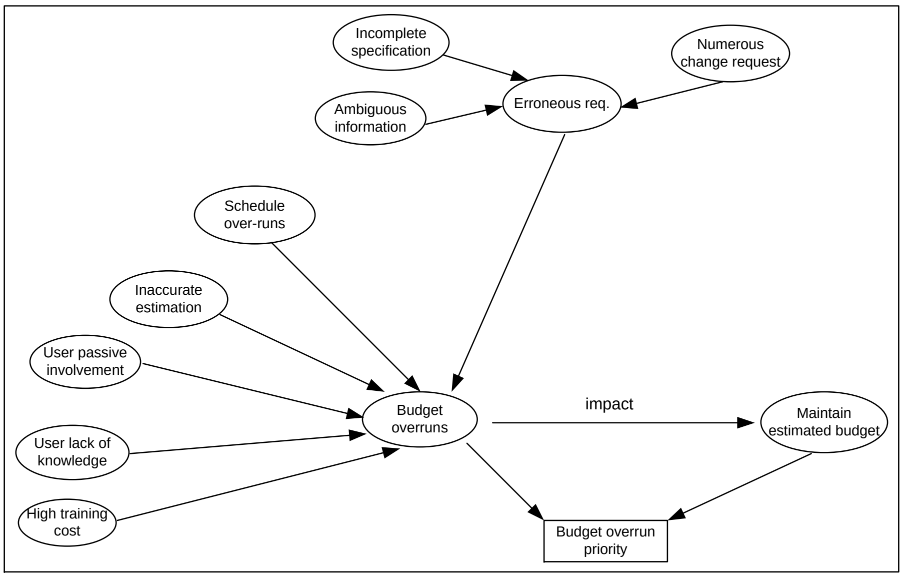


Figure 6.5: BBN Network for Budget Overruns

eliminate completely. RM team identified possible countermeasures to control the very important and important risk events, i.e., *train selected representatives from user groups*, *extensive user involvement*, *signed frozen requirements*, *complete understanding of users business processes and dependencies among the modules*, *obtain and incorporate key users feedback* and *include a J2EE expert into the project*. The identified countermeasures were potential and would not incur additional cost except integration of a J2EE expert. However, it was difficult to involve the key users even though they were available in the commission campus. And user lack of project domain knowledge and lack of IT competency could not be addresses by any means. However training selected users would reduce the overall training cost significantly. RM team emphasizes on professional and competence training and preparation of a comprehensive training manual before physically deploy the system. Still there was no effective way to address the numerous change requests but PM decided that once the feedback would be integrated then frozen requirements of a specific module will be signed from the key user. But at that stage, the project experienced a huge number of user change requests and several new things were added compared to the initial specification. PM agreed that it would not be possible to maintain the estimated budget and schedule and management decided to accept 15% budget overrun. The management also decided to keep their reputation high by any means and a complete understanding of the business processes is essential for the project as well as for the future customization. The management also decided to arrange *workshop sessions* with the high government officials by informing the status of the project and necessary action lists. These sessions would be effective to develop and implement the application into the planning commission campus.

6.4.3.6 Goal-risk model

Figure 6.6 depicts the goal-risk model for complete project in estimated budget. The goal refines into several sub-goals, such as maintain estimated budget and schedule, realistic estimation, clear milestones, user active participation and mo-

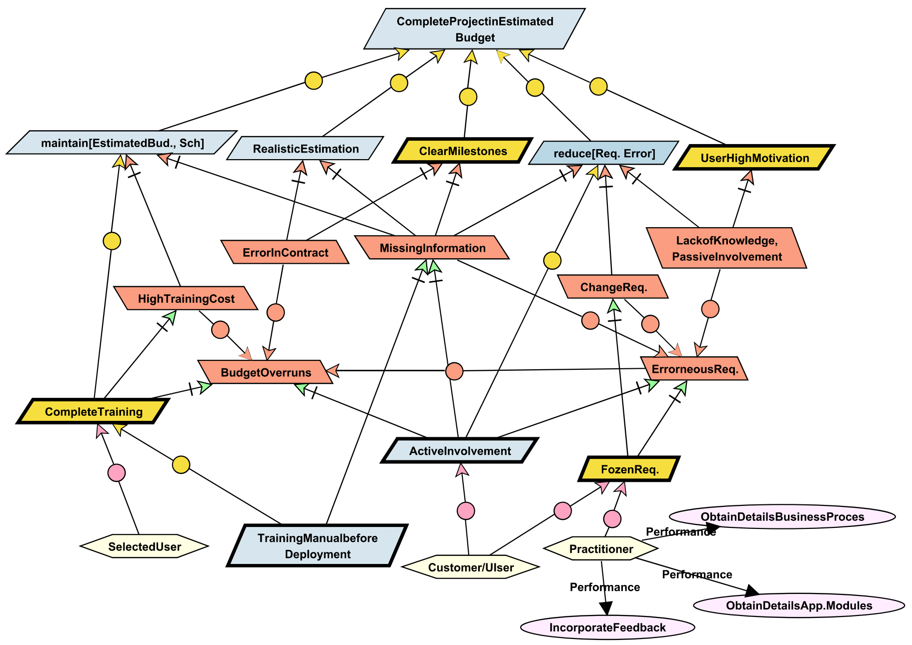


Figure 6.6: Goal-risk model for project completion

tivation, reduce errors from requirements. Risk factors, such as numerous change requests, user passive involvement and lack of knowledge, high training costs, inaccurate estimation, errors in contract and incomplete specification obstructed the sub-goals and lead to the risk events like erroneous requirements, schedule overruns and budget overruns. The bottom part of Figure 6.6 shows treatment actions and associated agents responsible to implement the actions. E.g. comprehensive and competence training.

Figure 6.7 shows the goal-risk model for obtaining positive reputation. The goal was rather subjective and refined into user satisfaction, quality product, successful training and system usage. Risk factors like poor training, ineffective communication, incomplete and complex product, retain existing system obstructed the goal and lead to user dissatisfaction and low reputation as the main risk events. User motivation for the new system, effective cooperation between users and practitioners, complete and competence training are important goals which can mitigate the risk factors. Eliminate old system was considered as a requirement within the control action. Project contract did not cover maintenance part which seemed to be important for this project context. PM decided to convince the project sponsor to allocate maintenance budget for the project. Domain Tech management also decided to arrange 2 or 3 common workshop sessions apart from training to motivate the users for the system-to-be deployed.

The difficult part regarding the risk management was to convince the key government officials about the control action implementation. The PM communicated the agreed control actions to the management and management planned to set up a meeting with the planning commission secretary. In the meeting, the secretary of planning ministry agreed on the recommended actions and assigned 20 key users to be extensively involved in development if necessary. The secretary also circulated an office order at joint, deputy and assistant secretary level to collaborate the project work by giving necessary information. Furthermore, only 58 users were

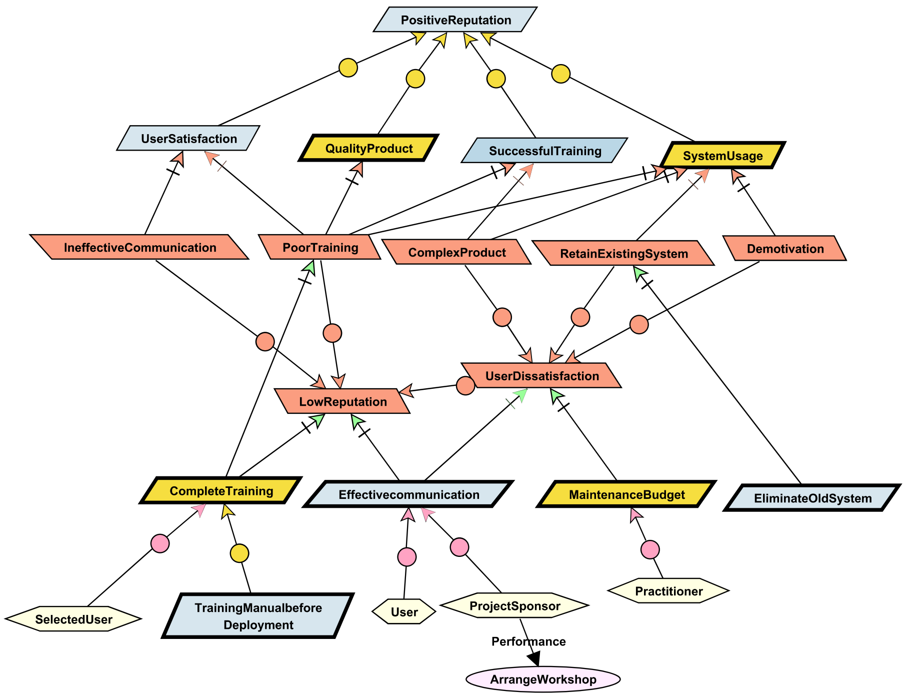


Figure 6.7: Goal-risk model for positive reputation

selected for the training session conducted by Domain Tech who will train the remaining users. PM decided to arrange a risk monitor meeting initially twice in a month and less frequently at a later stages of the development. At the end student members carried out the interview session with the open questions to obtain feedback about GSRM. The interview participants were PM and other members of the RM team, three practitioners of the development team, one management representative of Domain Tech and two users.

6.4.4 Comparison of the Study Result with Previous Research

In our previous studies, only offshore software development projects were investigated where customers/ users are located in different locations [IJH09, IHMFJ09, IH10]. But the present study context is a software project for a local customer. We compare study result with the previous research in the literature specifically those concerned software development risk factors and their influence to the project outcomes. There is a substantial commonality fully or partially among the present study result with the existence published risk factors. For instance, Schmidt et al. [SLKC01] identified a comprehensive list of risk factors and top factors like lack of adequate user involvement and cooperation, lack of frozen requirements, change scope and unclear/misunderstood scope/objective is similar to our findings.

The most noticeable areas which include distinctive risk factors by our study are user, user's organizational and requirements context which match with important risk components like schedule and budget, requirements management and personnel management demonstrated by Ropponen et al. [RL00]. The result also showed similarities with the Wallance et al. [WK04, WKR04] findings on requirements, user and complexity risk dimensions. Procaccino et al. [PVOD02] study result emphasize the factors related to the customer/user and requirements, such as user involvement, realistic expectation, complete and accurate requirements, and well defined project scope which highly influence the project success. Linberg [Lin99] also summarized factors like effective leaders, technologically realistic requirements and realistic schedule and effort estimation relevant for the project success. Our study result partially matches with these findings.

In the case study, PM realized the need of risk management and himself is assigned as the main authority for the task. However practitioner's are lack of motivated to implement the complete risk management process. The interview responses summarize a large number of risk factors from the project. A recent survey study result showed , that intangible benefit, lack of resource, too many risks to control are the main perceived barriers by the experienced project manager to a successful implementation of software risk management [OG09]. Nyfjord et al. [NKM08] and Kwak et al. [KS04] emphasize the organizational problems, such as variation of risk perception by different roles, lack of competence and process problems, such as lack of plan and coordination are additional barriers besides resource problems to integrate risk management into development. We observed that formal risk management practice was missing in Domain Tech. However, implementation of GSRM and its result certainly advocate for a detailed risk management practice in the upcoming projects. Some of the important factors from the other studies are not mentioned by the present study, such as no planning or inadequate planning, lack of management support, problems related to the development process and development team. Several risk factors which seem to have no or partial match compared to the other studies, such as user unwillingness to provide information and sign the interview document, bureaucracy nature of the organization, political biasness, numerous change/update requests, errors in contracts, training for large number of users and over promise. These factors dominate in our case and mostly originate from the user context.

6.4.5 Lessons Learned

There are several lessons that we have learned from this empirical study by employing GSRM into the software development project. This section provides a discussion of the lessons learned from this case study context.

6.4.5.1 Usefulness of the GSRM

The goal driven approach for risk management is beneficial during the early development. The goals make it easy to communicate with the users, project sponsors and development team members. The main users of the running project were government officials who do not have adequate knowledge about the purpose of automation due to lack of IT competence. Goals ease to understand the stakeholders' expectation and elaborate the risks which obstruct the goals. The integration of risk management into requirements engineering from the running project context facilitates to consider obstacles related to the budget, schedule,requirements , user training, user involvement, vendor reputation and user satisfaction even before the final acceptance of the requirements specification.

Based on the open questions responses, PM and practitioners appreciated the integration of risk management into requirements engineering. It provides them with early warnings about the problems existing in the project. The process and techniques of GSRM perceived as understandable and reasonably applicable. The explicit integration of risk management activities enforces project manager to allocate schedule for risk management and incorporate development team members to take an active part in the risk management activities. The project members gained knowledge about the possible risks and expressed their views to control the risks. The component-element-factor hierarchy allows to systematically identify and categorize goals and risk factors. Techniques, such as closed question and brainstorming session for risk identification, goal-risk model, causal relationship model, guidelines to assess and treat risks are treated as comprehensive. The identification and modeling of goals and risk factors and their visual representation facilitates to increase the understanding and communication about the risk. However this part consumed the highest effort, hence large number of goals makes GSRM more complex.

Two kick-off workshops and five brainstorming sessions each approximately 4 and 6 hours and 14 interviews (2 hours for the closed questions by the practitioner and more than 2 hours and 30 minutes by the users, 1 hour and 30 minutes for open questions) were used to implement GSRM. Furthermore, student members analyzed the project documents, in particular, artifacts like initial specification, nature of change requests, detailed requirements specification, project scope and success criteria as input data for risk management. They also compiled the interview responses to identify risks and to obtain feedback about GSRM. It took approximately four complete working days for the analysis. The brainstorming session also considered requirements engineering activities, such as analysis of the requirements documents and user change requests. It is feasible to identify the goals and risk factors apart from the requirements elicitation and analysis activities within the same session. It allows GSRM to be well integrated into the early software development. Goals and risks identification and modeling consumes the highest effort and project complexity and large number of goals increases the overall risk management effort. A total of approximately 16% effort of initial deliverable phase including the student memebers effort used for risk management. But the effort estimation excludes the risk monitor activity.

There exist several overheads by GSRM related to the scalability issues of the overall approach. First, when the project focuses on large number of goals, then GSRM needs more effort. Second, constructing a casual relationship model through BBN is complex undertaking specifically when several risk factors are considered as intermediate nodes, which increase the size of probability density table. Third, risk monitoring activity was not properly performed at later stages due to the pressure increased to handle the user changes. Fourth, it is difficult to answer some of the closed questions in particular quantify the answers into three different scales at an early development stage. Among the 7 risk events and 15 associated risk factors only 3 risk events and 11 risk factors are fully or partially controlled through 8 potential countermeasures. However, GSRM fails to control budget and schedule overruns which obstruct one of the main project goals. This is because most of the risk factors originated from the user perspectives which are beyond the project manager control. It is important to have the risk factors under the control of project manager and project environment. Some risk factors cannot be controlled, for instance requirements changes, user motivation for the system-to-be and inadequate user knowledge. But some are fully or partially controlled, such as high training cost, overall budget overruns, erroneous requirements, users participation and detailed system specification.

6.4.5.2 Overall Observation

GSRM is suitable for any complex project. Our previous case study focused on a medium size project. But this case study project is complex, where GSRM is well-integrated to identify and control the software risks. The replication of the case study about the same method in a varying context allows generalizing the main findings of GSRM. In both cases, project manger agreed that by GSRM it is possible to identify and tackle the problems in a structured way from the beginning of the development.

In the previous case study [IH10], we considered a set of hypothesis, i.e., i) risk management activities can be well integrated in requirements engineering, ii) GSRM contributes to manages risk by considering a holistic perspective and iii) GSRM effectively reduces errors from the elicited requirements. The goals were attained and risk factors were mostly mitigated and hypothesis were confirmed in the previous case study. The current study project is complex and highly risky. The user factors are the central sources of risks. One of the main goals, i.e., complete project in estimated budget and schedule was not achieved. In the present study, risk management sessions include RE activities and contribute to control requirements errors and other risk factors from both technical and non-technical issues. But not all requirements problems are solved as well as the project suffered budget and schedule overruns.

By analyzing the observations from the running study, risk management can be well integrated into requirements engineering and GSRM focuses on the holistic view of the risk management. The process and artifact oriented view allow to integrate risk management into requirements engineering. Software projects generally do not have a risk manager except large or complex projects. We have seen that a project manager or requirements engineer having basic knowledge of risk management are able to perform the GSRM activities. Our observation is, that when factors are beyond the control of the project manager and development environment then it is difficult to control the risk. The studied project failed to complete within the estimated budget and schedule. The PM later on by a conference call stated, *"we suffered approximately 25% budget overruns even though we mitigated some of the risk factors and considered 15% acceptable limits for the budget overruns. This amount is huge for us compared to our other completed projects. As the users were the government officials and project funded by the UNDP, there was no room to obtain extra budget. Our management considered it as a loss project. But we have obtained the next work order from the planning ministry which implies that the government officials are happy with our performance "*. PM believed that without the integration of risk management budget overruns could be even more, government officials would not be consider to significantly participate in the project and requirement errors could not be controlled from the early stage. The project is planned to be deployed in ministry campus and selected users are currently under training. As Domain Tech already obtained the phase 2 of the project which includes three more modules in the existing application software . Therefore one goal is attained i.e., obtain good reputation. The upcoming project concerns annual development program, project and foreign aid monitor system of the planning ministry and maintenance. The management hope that phase 2 would compensate the loss suffered by the current development.

Our further observation is , that it is not necessary to always consider budget and schedule factors at the highest priority. Software development project is a complex undertaking. There exist other issues, such as requirements, users, change management, user satisfaction and system usage which directly or indirectly influence budget and schedule constraints. These factors need early attention in the development. Early determination of the nature of project riskiness is very effective to plan the risk management activities. Current study concluded that project scope affects all dimension of risks but for high risk project risks associates with requirements specification, users, change management, project execution are more obvious.

6.4.6 Study Validity

Case studies are generally criticized for being biased, less valuable or unable to generalize the findings [RH09]. Threats to case studies are related to the difficulties of collecting reliable results and on generalizing the findings. We systematically planed the study, considered study goals and addressed the issues related to the threat from the beginning of the study. Data is collected and analyzed in a consistent way from multiple sources to overcome the study validity threats .

6.4.6.1 Internal Validity

We try to reduce the expectation bias on the case study result. The interview responses are commonly analyzed in the brainstorming session by the risk management team. None of the principle investigators of GSRM and related research was directly involved in the case study which aims to reduce the bias of the findings. Data was collected not only from the development team but also from the customer and project sponsor representatives. Furthermore, project documents are also analyzed to understand the goals and risk factors. Therefore, several sources were used to collect data and this limits the effects of one interpretations of one single data source. Interview response and result of the project document inspection was further analyzed and discussed in the brainstorming sessions. The PM and other team members were not fully motivated for a formal risk management practice in the project. This attitude was changed later on by observing the outcome of GSRM activities. This means that the GSRM approach demonstrated the importance of risk management and quickly produced visual and critical insight. Maturation effect also intimidates the internal validity in particular when the participants react differently about the perception on risk during the course of development. Activities of GSRM took place at the requirement engineering phase, the threat can not significantly affect our study.

6.4.6.2 External Validity

The case study context is located in a single geographical region, there is a possibility of a cultural bias. Our findings are from multiple data sources which support for the stronger conclusions. The study result is compared with other results from the literature to generalize our findings. The comparisons confirmed several commonalities in terms of goals and risk factors, however there are also some unique factors from the local context. Therefore, factors related to goals and risks are influenced by the local context. We provide additional insight about the impact of risk management in software development project.

6.4.6.3 Construct Validity

We considered benefits and limitations of the GSRM and correctly measured quantitative metrics like risk management effort to support the case study objective.

The quality of the case study questionnaires was improved by following the feedback from our previous study. The participants answered most of the questions apart from some which are observed as difficult at an early stage. The threat of the questions being the wrong ones to ask was mitigated. The project managers and development team members have adequate experience on several software development projects. On the other hand, user representatives have also long experience in the government service. The student members conducted a kick-off workshop to the practitioners and explained details about the interview purpose to the users. We believe that the participants understood the terms being used.

6.4.7 Case Study Conclusion

GSRM is a practical approach in an industrial context to control software risks with a reasonable effort. It provides a rational systematic framework by considering both subjective and objective analysis and involves all concerned stakeholders to assess and manage software development risks. The goal-driven risk management approach is well integrated into requirements engineering so that factors related to software risks can be handled event before analyzing the requirements. The study results confirm our findings from the previous study. We believe this study is useful to improve GSRM and to inform and motivate practitioners about the effectiveness of integrating goal-driven risk management activities from the early development phase.

6.5 Goal-Risk Taxonomy

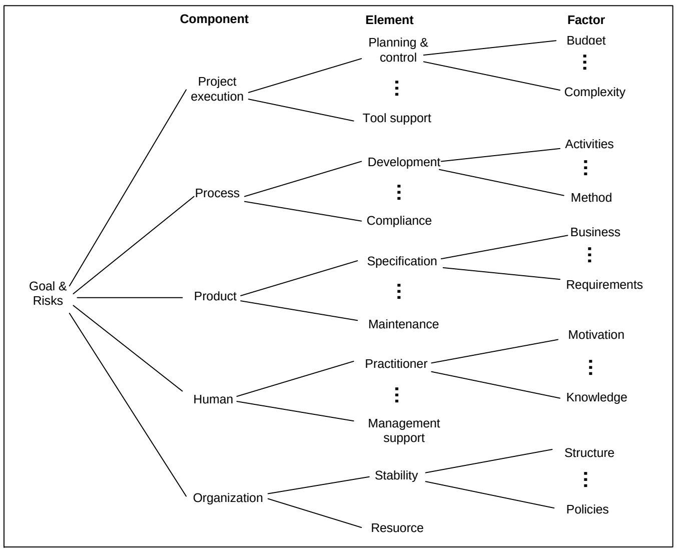


Figure 6.8: Schematic View of Goal-risk Taxonomy

This section provides an overview of the goal-risk taxonomy for a software project. We mainly follow fundamental concept of GSRM, empirical investigation results of this research and published literature on software success and risk factors to derive this taxonomy. The taxonomy will be used in GSRM activities to systematically identify, categorize and model goals and obstacles. The component-element-factor hierarchy (i.e. stated in chapter 4) used as main foundation concept to organize the taxonomy. The closed questions (i.e., included in appendix 1) and requirements errors checklist (i.e., included in appendix 3) are considered as a part of this taxonomy to identify the risk factors. The hierarchy contains five components. They are:

- **Project execution:** This component refers to the issues associated with the project planning and control, budget, schedule, project scope, complexity, tool support and overall inherent riskiness nature of the project.
- **Process:** This component refers to the issues associated with the activities and methods related to the software development and risk management.
- **Product:** This component refers to the issues associated with the product artifacts such as requirements specification, time-to-market, requirements faults, quality and product operation and maintenance.
- **Human:** This component refers to the issues associated with the project stakeholders, i.e., practitioner, customer/user, management and overall team work in the project.
- **Environment:** This component refers to the issues associated with the overall internal and external organizational environment in which a software project takes place.

Individual component refines into element and factors to provide lowest level abstraction of the component. Figure 6.8 shows a schematic view of the goal-risk taxonomy.

Figure 6.9 elaborates Figure 6.8 by showing which characteristics of the development components and its surrounding environment do the project managers should consider when assessing and managing goals and risks. These characteristics are influential issues of the project whose satisfaction increases the likelihood of project success and vice versa. We identify the goals and risk factors of these components to construct the goal-risk taxonomy. The individual goal and risk taxonomy are shown in Figure 6.10 and Figure 6.11.

Goal-risk Taxonomy				
Project execution	**Process**	**Product**	**Human**	**Environment**
Planning & control	**Development**	**Specification**	**Practitioner**	**Organization**
Estimation	Process	Business specification	Skill	**stability**
Budget	Methods	System vision	Knowledge	Structure
Deliverables	Models	System req.	Motivation	Reputation
Monitor	Level of abstraction	Integrational req.	Commitment	Process
Review	Mapping	Organizational req.	Assumption	Change
Variation	Roles	**Time-to-market**	Availability	Control
Roles & responsibilities	Inputs	Customer market	Training	Policies
Users groups	Artifacts	Competitor analysis	User expectations	Project support
Technical complexity	**Risk management**	Consumer analysis	**Customer/user**	Training
Complex task	Budget	**Quality**	Involvement	Culture
External links	Resource	QA plan	Collaboration	Political influence
Project size	Motivation	Performance	User groups	**Resource**
Volume of code	Integration	Usability	Knowledge	Development
PM skill	**Consistency**	Security properties	IT competence	Infrastructure
Change management	Concurrency	Security threats	Business env.	
Project scope	Synchronization	Privacy properties	Confidence	
Goals	Traceability	Privacy harms	Expectations	
Success criteria	**Compliance**	Safety properties	Feedback	
Boundary	Performance	Critical hazards	**Team work**	
Business value	Improvement	System failure	Team structure	
Reusability	Audit	User manual	Knowledge	
Innovation		**Testing**	Performance	
ROI		Test specification	Coordination	
Inherent risks		Test cases	Conflicts	
Tool support		**Requirements faults**	Users' feedback	
Development		Ambiguity, incorrect	**Management**	
Maintenance		untraceable, infeasible	Leadership	
Project execution		unstable & incomplete	Support	
		Operation	Commitment	
		Plan	Business strategy	
		Budget	Conflict	
		Acceptance criteria	Confidence level	
		Training	Review	
		Technical manual	Customer focus	
		Operational manual	Improvement	
		Synchronization		
		User's motivation		
		Installation		
		Migration		
		Maintenance		
		Budget		
		Scope		
		Acceptance criteria		
		Deliverables		
		Versioning		
		System complexity		
		Consistency		
		Performance		
		Testing		
		Documentation		

Figure 6.9: Detailed Properties of Goal-risk Taxonomy

Goals				
Project execution	**Process**	**Product**	**Human**	**Environment**
Planning & control	**Development**	**Specification**	**Practitioner**	**Organization**
Well planned	Well documented	Clear business-	Skill & experienced	Stable structure
Realistic estimation	Controlled process	specification	Adequate project-	High reputation
Realistic deliverables	Effective methods	Complete system vision	domain knowledge	Control processes
Clear milestones	Correct methods	Realistic requirements	High motivation	Effective policies
Maintain estimated	Model abstraction	**Time-to-market**	Correct assumption	Appropriate change
budget	Precise mapping	Competitor analysis	Staff availability	Subcontractor control
Maintain estimated	Realistic modeling	Consumer analysis	Proper trained	Adequate training
schedule	Clear roles	Unique features	Familiar with	Less political biased
Periodic monitor	Consistent tasks	Consumer satisfaction	development env.	**Resource**
Minimum variation	Complete artifacts	**Quality**	Meeting user needs	Adequate
Clear roles	**Risk management**	Complete QA plan	**Customer/user**	development &
All user groups detailed	Strong RM	Quality product	Active involvement	communication
Control complexity	Motivation for RM	Usable product	Strong collaboration	facilities
Mature technology	Resource allocation	Security threat analysis	Key users	
Technically feasible	Clear roles	Accurate security level	Adequate project	
Effective PM skill	Consistent tasks	Privacy harm analysis	knowledge	
Effective change	Complete artifacts	Accurate privacy level	High IT competence	
management	Process integration	Safety hazard analysis	Stable business env.	
Project scope	**Compliance**	Accurate safety level	User satisfaction	
Clearly defined goals,	Productive process	**Testing**	Motivation	
success criteria &	Periodic audit	Complete test plan	Realistic expectation	
boundary	Review process	Possible test cases	Frequent feedback	
Clear business value	Continual-	Addressing test results	**Team work**	
Minimum scope creep	improvement	**Requirements faults**	Diverse team	
Reusability		Reduce req. errors	Adequate knowledge	
Innovation		Unambiguous , correct	Productive team	
Economically feasible		traceable, feasible,	Strong coordination	
Control inherent risks		frozen, & complete req.	Minimum conflicts	
Tool support		**Operation**	Effective	
Correct tools		Complete plan	communication	
		Satisfy acceptance	Addressing feedback	
		criteria	User confidence on	
		Proper documentation	project team	
		Competence training	**Management**	
		Training budget	Effective leaders	
		Meeting acceptance	Adequate support	
		User's high capability	High commitment	
		Proper installation	Strong business	
		Complete migration	strategy	
		Successful use	Minimum conflict	
		Maintenance	Customer confidence	
		Adequate budget	on management	
		Goals	Periodic review	
		Clear acceptance	Customer focus	
		criteria	Continual	
		Deliverables	improvement	
		Minimize volatility		
		Minimize complexity		
		Version control		
		Retested components		
		Updated documentation		

Figure 6.10: Goal Taxonomy

Risks				
Project execution	**Process**	**Product**	**Human**	**Environment**
Planning & control	**Development/ RM**	**Specification**	**Practitioner**	**Organization**
Inadequate plan	Poor documented	Unclear business-	Lack of skill	Unstable structure
Poor control	Uncontrolled process	specification	lack of domain	Low reputation
Unrealistic estimation	Ineffective methods	Incomplete vision	knowledge	Lack of process-
Unrealistic deliverables	Wrong methods	Unrealistic req.	Wrong assumption	control
Over/under estimation	Unrealistic modeling	**Time-to-market**	lack of motivation	Frequent change
Hidden factors	Lack of abstraction	Incomplete competitor	Unavailable staff	Ineffective policies
Budget overruns	Imprecise mapping	& consumer analysis	Inadequately trained	Inadequate training
Schedule overruns	Inconsistent activities	Consumer-	Unfamiliar	Lack of contractors'
Irregular monitor	Poor RM	dissatisfaction	development env.	control
No tracking	Lack of motivation for	**Quality**	Fail to add user needs	Lack of distributed
High variation	RM	Incomplete QA plan	**Customer/user**	sites' control
Unclear roles	Lack of resource	Poor quality product	Passive involvement	Bureaucracy nature
Large user groups	Unclear artifacts	Poor performance	Weak collaboration	Politically biased
High complex task	Unclear roles	Poor usability	Missing key users	**Resource**
Technical complexity	Poor integration	Inadequate analysis of	Lack of knowledge	Inadequate
Immature technology	**Compliance**	security, privacy &	Low IT competence	development facilities
Large no of external links	Unproductive process	safety	Unrealistic expectation	Environment problems
Complex interactions	Infrequent audit	**Testing**	User dissatisfaction	
Lack of PM skill	Inadequate review	Incomplete test plan	Unstable business env.	
Ineffective change	Infrequent-	Inadequate test cases	Lack of motivation	
management	improvement	Lack of focus on result	Numerous change	
Project scope		**Requirements faults**	request	
Unclear goals, success		Erroneous req.	Inability to provide	
criteria & boundary		Incorrect, infeasible,	project information	
Unclear business value		inconsistent &	Poor feedback	
Errors in contract		ambiguous req.	**Team work**	
Continuous scope creep		Numerous change	Imbalanced team	
Significant increase of		Incomplete system req.	Inadequate knowledge	
scope		**Operation**	Less productive team	
Lack of reusability		Incomplete plan	Weak coordination	
Economically infeasible		Poor documentation	Frequent conflicts	
High risk project		Ineffective training	Low productivity	
Tool support		Lack of budget	Miscommunication	
Incorrect tools		User's incapability	Lack of coordination	
		Lack of user's support	Lack of skill	
		Deployment problems	Infrequent feedback	
		Incomplete migration	**Management**	
		Poor system use	Ineffective leaders	
		Maintenance	Lack of commitment	
		Lack of budget	Poor support	
		Unclear goals	Immature business	
		Unclear acceptance	strategy	
		criteria	Over promise	
		Missing deliverables	High rate of conflict	
		High volatility	Lack of user confidence	
		High complexity	Infrequent review	
		Lack of versioning	Lack of customer focus	
		Lack of testing	Irregular improvement	
		Poor documentation		

Figure 6.11: Risk Taxonomy

6.6 Empirical Studies Conclusions

A total of 6 companies were involved for the survey and case studies. We would like to thank Domain Tech [Dom], Spectrum Engineering and Consortium Limited [Spe] for supporting the case studies and Southtech Limited [Sou], ibacsbd Limited [Iba], Pyxisnet Limited [Pyx] and Rasis Limited [Ras] for supporting the survey. Our findings are that:

- It seems reasonable to start with goals for risk assessment and management. Goals tightly link with the obstacle, allow to understand and model risk management and ease to communicate with management about the risks existing in the project.
- The software development risk management needs to be initiated as early as possible during the development. Risk management can be well integrated into requirements engineering and facilitates to identify, assess, manage and trace risks of the early development.
- GSRM is a practical approach for software development risk management. It equally considers both technical and non-technical risks from a holistic perspective. Based on the project context, the GSRM process can be customized for an effective risk management and well integrated into software development. The component-element-factor hierarchy is systematically used to identify and categorize goals and risk factors for any software development project.

The empirical study result positively contributes about the impact of risk management into software development projects. We hope this would motivate the project stakeholders to include a comprehensive risk management practice in the software projects.

CHAPTER 7

Conclusion

Contents

Every software project is unique and undertakes multidimensional tasks during the course of development, deployment and maintenance. Early risk management is always effective to meet the project goals and to produce a more reliable and dependable product. A project combines knowledge and technology and contains both generic and project specific risks at every phase of the development, which have an extremely high influence on the success. Several researches concluded that a complete risk management process is rarely followed. A system under development is merely analyzed from the technical perspective and often non-technical issues are overlooked. For instance, issues related to humans who develop and use the product and organizational settings in which the system is developed and deployed. Therefore, a systematic and easy to use risk management process from a holistic perspective and its explicit integration at an early development stage is required to ensure an effective risk management practice. We believe that this research work contributes in this direction. The motivation for this work is derived from literature review and our own experience from software development projects.

7.1 Outcome of the Research

By extensive investigation of the state of the art, we summarized that both industry and academic emphasize to initiate the risk management as early as possible. But a detailed guideline is still missing to integrate risk management activities into the early development phase by considering a holistic view. A number of survey studies focus to identify risk factors of software projects. However, relatively a little effort has gone into assessing the overall impact of the risk management. This research contributes to explicitly integrate risk management activities into requirements engineering phase, links project specific goals and risk factors and empirically evaluates the impact of risk management method into a software project. It provides a structured way to look at the project and problems associated with the project. We consider a goal-driven approach for the software development risk management. It helps to rationalize and justify the risks and their treatment in a software development project. Natural language is used to represent the goals and risk factors by following the KAOS template. This allows to easily understand the goals and risk factors. This goal-driven approach, in particular, goal modeling for risk management is considered to be beneficial by the participants involved in the *empirical study*.

The development components and their refinement into elements and factors provide a comprehensive view of goals and risk factors from a holistic perspective. GSRM allows risk management in a multi-dimensional setting, where goals from the project stakeholders, development components and project success indicators are simultaneously considered. This also eases to identify the risk factors and to estimate their impacts for early mitigation. In a large project, it is sensible to completely implement the model for software development risk management. We precisely elaborate the underlying activities, methods and roles of the goal-driven risk management process model. The process explicitly integrates GSRM into requirements engineering. Artifact oriented view is considered to construct the risk specification and its associated concepts such as goals, risk status, goal-risk model and causal relationship model. Artifact view supports to precisely understand the work products from the risk management domain and links them with the requirements engineering artifacts. All artifact concepts and attributes of the risk specification are illustrated by the GSRM. This research develops a goal-risk taxonomy and constructs a questionnaire consisting of 200 closed questions which are arranged by following the *component-element-factor* hierarchy and a requirements errors checklist. The purpose of the closed questions and of the checklist is to identify the risk factors and requirements errors. Our empirical investigation result shows positive impact of risk management on the software project's successful outcome. This conclusion identified some unique finding from the local context. Thus novelty of the dissertation consider:

- Introduction of a goal-driven approach for software risk management;
- Applicability of risk management practice into requirements engineering;
- Impact of risk management on a software project;
- Identification of goals, risk factors and events and construction of a goal-risk taxonomy for software projects.

We believe the study results bias the project practitioners to integrate an effective risk management practice within the early development stage. GSRM is an excellent project management tool that makes aware of the pitfalls exist in a software project. Our study focused on completely different cultural and environmental context compared to other study results in the literature. We have chosen Bangladesh as a representative of a developing countries with limited IT infrastructure facilities. But in Bangladesh offshore and local software development and use market is rapidly expanding through significant increase in investments in recent years. E.g., the European Union has ranked Bangladesh as one of the top 20 outsourcing destinations in the world. This type of study is the first of its kind in Bangladesh and will effectively support to enhance the software industry.

7.2 Conclusions about Research Questions

We considered four research questions at the beginning of the research. Here we provide a summary response of the research questions after the course of research work and empirical investigation.

7.2.1 Research Questions 1 and 2

- **Question 1:** *Can risk management be integrated in the early phase of software development projects?*
- **Question 2:** *What is the effect of a goal-driven approach on the software risk management?*

These two questions mainly consider the goal-driven approach for risk management and its integration into early development phase. The results and discussion of the questions reveal that risk management is effectively integrated into early development phase specifically within requirements engineering phase. And the goal-driven approach is well fitted for software development risk management. We consider integration points of requirements engineering and risk management. The integration points focus on activities, methods, roles and responsibilities, resulting artifacts and their dependencies. For instance, similar techniques can be used to elicit the goals, requirements and risks. When project considers workshops or brainstorming sessions with the key users or stakeholders to identify their expectations and system specification, the same session can be used to identify goals and risk factors. Project managers and requirement engineers are able to perform risk management activities with having some basic knowledge of risk management. Requirements artifact concepts such as business goals and specification, system vision, system, integrational and organizational requirements are considered as input for the risk identification. Project specific goals, risk priority and risk status support to refine the requirements effectively. The results of the case studies showed, that GSRM can be well integrated into requirements engineering and the goal-driven approach is effective to realize this integration. There are indeed strong dependencies among requirements and risk artifacts. GSRM contributes to reduce the errors existing in the requirements specification. The explicit integration of risk management at early development facilitates to identify and control the critical development issues such as practitioner domain knowledge, extensive user involvement, team work, resource availability, realistic estimation, time-to-market and product deployment and training plan even before elicitating the system specification.

The goal-driven risk management method is feasible in practice even when the project is in tight schedule and budget pressure, because a project in general contains goals and expectation from users, sponsors, practitioners and development component. GSRM maps these goals with project success indicators. Project managers and practitioners easily learn and use the GSRM, the only prerequisite is the basic knowledge of goal modeling and risk management. A short training is sufficient to provide a basic understanding of GSRM and project managers and requirement engineers generally have the basic knowledge. We obtained similar results from both of the case studies of different project contexts w.r.t. research questions 1 and 2 . The risk specification artifacts were well documented. The project contexts were not similar but the main focus was to develop information systems, one for an offshore customer and the other for a local government ministry customer. The project practitioners who were involved into implementing GSRM were also satisfied with the resulting set of artifacts. The questionnaires by following component-element-factor hierarchy to identify the risk factors and a brainstorming session to identify the goals and risk factors were appreciated by the practitioners. The process involved and the artifacts produced by GSRM are consistent. The perception of risks always varies from individual to individual, as risk estimation is subjective and requires expert judgment. This inherent nature of risk makes it difficult to obtain precise risk estimation results and complicates the risk management method.

7.2.2 Research Questions 3 and 4

- **Question 3:** *What are the main goals of early development contributing to a successful project outcome?*
- **Question 4:** *How can the software development risks be assessed and managed from a holistic perspective to satisfy the goals?*

The analysis of research question 3 and 4 focuses on identifying goals and risk factors in software development project as well as assess and manage the identified risks to attain the goals. We follow the performed survey and case studies results (i.e. details in the next section) to address these two research questions. In general several goals were identified from the studies. Among them, some were generic and the others were project specific. Project goals consider various dimensions depending on the context from concrete economic benefit like return of investment to subjective value like user satisfaction, system use, reputation and knowledge management. We follow project documents, user operating environment, closed questions along with brainstorming sessions to identify and assess goals and risk factors. Initially raw risk factors are identified which obstruct the goals by following the closed questions. These factors are refined to risk events by following the causal relationship model. GSRM provides a general guideline apart from practitioners expertise to estimate risks. The guideline states which factors are in general causally linked to risk events and impact of an individual risk event on the goals depending on the influential risk factors and project context. Project riskiness nature is also important in this context so that inherent factors responsible to high a risk project get more attention. For instance in a high risk project, factors related to requirements, user and project execution are more noticeable. The project scope affects both the high and low risk projects. We advocate to determine the riskiness nature of the project during the pre-project planning phase so that risk management can effectively contribute for a successful project outcome.

7.3 Conclusions about Empirical Study Results

Survey Study A survey was performed to identify the possible goals and risk factors of offshore software development projects. The survey participants were experienced practitioners. The average experience of the participants was more than 2.5 years and they had multiple roles like director, developer, project leader, software engineer, architect and tester. The result showed, that the top 3 prioritized goals were: 1) attain project execution factors (sub-goals: stay in budget, attain product quality), 2) manage human factors (sub-goals: quality and relevance of staff, effective communication and coordination, proper management direction and support, effective client involvement) and 3) manage organizational factors (sub-goals: stability of the organization, adequacy of the development facilities and resources, effective risk culture). The result outlined, that passive user involvement, unstable and ambiguous requirements, lack of communication and coordination and lack of commitment and capability among management were the very important risk factors. These risk factors were also emphasized by other published risk factors in the literature. The survey concluded that security issues, legal disputes and time difference were not creating any problems even though customers were located in different countries. Participants observed that they have similar technical expertise as the user. However, obtaining the project work order despite marginal profits is an endemic problem by all vendors and this some times affects the employees' in particular related to their job satisfaction, motivation and productivity. As a consequence, employees' absence is a common problem in the local context.

Case Study The two case studies were from two different contexts, where we attempted to implement GSRM into ongoing software development projects. The main purpose of the case studies is to evaluate the impact of risk management in particular GSRM into software development projects. Kick-off workshops, interviews and brainstorming sessions were set up to perform the case study. Our study also supports action research as the risk management results effectively use to mitigate the identified risks in the project.

The first study was an offshore project where GSRM was not fully implemented due to tight schedule pressure. Several goals were identified such as improve realistic budget and schedule estimation, clear roles and responsibilities, reduce errors from requirements, improve team members competency and reduce customer/user dissatisfaction. The case study participants identified and analyzed several risk factors w.r.t. project context. The project was inherently a low risk project because project practitioners have done similar projects before which allowed to reuse their knowledge. But still there are some challenges involved in the projects. The prioritized risk factors were: under-specified, unstable, incomplete requirements, technical complexity, unclear business process, inadequate knowledge of the practitioner and customer/user passive involvement. These risk factors influenced risk events such as erroneous requirements, overall project complexity, customer/user dissatisfaction and budget and schedule overruns. Risk factors were mainly taken from requirements and user perspectives. The project manager agreed that integrating GSRM into the early development certainly helped them to identify and control these issues. The case study concluded, that there are indeed strong dependencies among requirements engineering and risk management process and artifacts.

In the second case study, GSRM was evaluated once again but in a different project context to generalize our findings. Here the project manager felt the importance of GSRM as the project was challenging and the development team treated the project as a high risk project. The customer was the planning ministry, the project sponsor was the aid agency UNDP and users were the government employees. GSRM was fully implemented in this study. The top four goals of the project are: complete project in estimated budget and schedule, complete user's training, obtain positive reputation and generalize the application. The vendor management's future business vision was extremely important in this case to obtain work order of similar application context, because recently local government decided to digitalize all its ministries operational activities. The vendor's management over-promised and agreed to accept any change during the course of development. This contractual error provoked numerous change requests from the users. More over users were very passive and unable to provide complete information about the system. The team also had lack of experience to handle the government officials. As a consequence, the estimated schedule exceeded and the project suffered from 30% budget overruns. Although GSRM was implemented, some goals were not attained and not all risk factors were controlled. But the vendor succeeded to keep high reputation which rewards them by obtaining the next work order of the planning ministry.

We have several observations and lessons learned from the case studies. Determining the riskiness nature of the project before initiating the risk management activities is very useful for an effective risk management practice. It helps us to be aware of the problematic factors from the very beginning of the project. Riskiness nature of the project supports to define the risk management scope and to plan and tailor activities for the GSRM. The goal-risk and causal relationship model ease to understand and communicate risk information with the key project stakeholders. More goals provide more interactions among the development components. A software development project can not focus on all the identified goals, hence goal prioritization is essentially based on the project context, complexity, riskiness nature, customer expectation and practitioners expertise. Goal and risk identification and analysis is the most critical part of the software risk management. Risks which oppose the prioritized goals need immediate attention and GSRM allows to handle these risks from the early development. In the first case study, the project manager was not really motivated about a formal risk management practice. At the end in both cases, project managers observed that GSRM was able to identify problems and tackle the problems from the beginning of the project. We have also noticed that no additional activities or artifacts were required for the risk management. The proposed goal-driven risk management model is complete in a sense that it adequately handles all goals and risks associated in the project. Requirement problems were the top prioritized software risk and mainly due to the factors related to human perspective such as passive involvement, lack of knowledge and numerous change requests. Some factors are unique and dominating in the local context such as contractual problems, bureaucracy nature of a government organization, marginal profit and over-promising tendency to win the work order and to keep high reputation.

We conclude that it is not always necessary to rank budget and schedule goals and risk factors at the highest priority for the risk management. Project context specific factors must need adequate attention and every factor directly or indirectly relates to the budget and schedule. E.g., requirements problems in the first study and numerous change requests, inadequate knowledge, contractual error in the second study. Controlling these factors can have a significant impact on the ability to meet the budget and schedule targets. Some projects are carried out to gather knowledge, experience or to obtain high reputation for the future business benefit. A project commonly contains a large number of risk factors and goals and project managers are not always interested to consider all of them. Only the top ten or twenty risk factors and high prioritized goals need initial attention. Risk factors are causally linked with each other and responsible for the risk event. Hence controlling the top risk factors certainly contributes to minimize the impact of the other causally linked risk factors. Early consideration of risk management activities is very effective for projects of any size. The goal-driven approach for risk management and its explicit integration into requirements engineering is well fit in software projects.

7.4 Limitations of the GSRM

Several limitations were observed in the proposed framework. The observed limitations are given below:

When it comes to consider scalability issues, then GSRM suffers in particular within a large project context. A large scale software development project contains a huge number of goals, sub-goals and risk factors, which incur to increase the complexity of the goal-driven risk management model. Goal refinement does not provide any specific boundary for the level of abstraction. The more goals are refined the more effective the risk assessment and treatment could be. It also brings overhead for risk management. Projects may have thousand of requirements and it is not possible to identify errors by reviewing individual requirements. A large number of goals and risk factors carry considerable overhead in constructing, maintaining and documenting the risk specification artifact. Hence it is not always feasible to construct the model for every risk event, especially for projects with continuous time or budget pressure. Also creating and managing detailed textual representation about the risk artifacts requires extra effort. Therefore, the effort of using GSRM seems to be higher for a large project context. Even if the project size is medium or small but schedule pressure is the highest priority, then project team members might not have the opportunity to detect and resolve every risk.

GSRM also needs to be tailored based on the project specific context. We advocate to involve key project stakeholders and development team members to tailor the risk management process. In particular, when risk initialization activities are carried out to determine the risk management scope and to allocate schedules and resources. Our observation through the empirical investigation was that early determination of the project riskiness guides to define the risk management scope and plan for risk management activities. More study is required to establish a link between project riskiness with GSRM. Individual perception about the risk estimation also varies a lot. One can overestimate a risk while underestimating another. This affects the actual result of the risk management , in particular to select the treatment action for the risk mitigation. Furthermore, there exist many risks that can not be avoided or confined. These risks should be mitigated or monitored despite of budget and schedule constraints.

During the development phase, uncertainties critical to success unfold as the software project proceeds. GSRM systematically models these uncertainties through the Bayesian Belief Network (BBN). But risk estimation is not trivial and BBN looks at the relationship of these components. BBN constructs a causal relationship model among risk factors and events and their impact on the goals. Computation in BBN is complex if risk factors or events play the role of intermediate nodes. It is difficult to manage the probability density table of BBN for multiple intermediate levels. This makes the whole estimation complex which is highly undesirable from the industry point of view for an effective and comprehensive risk management practice. One may also argue that risk estimation, specifically risk impact to the goal, is not precise due to common guidelines and practitioners' subjective judgment. However, as stated highly precision is not always possible but complex estimation can make the practitioners reluctant for a comprehensive risk management practice. At early development stage, sometimes adequate information is not available to judge some risk factors or to determine the risk threshold value. But we need proper implementation and monitoring of the selected control actions. Otherwise, risk management would not be effective for the software development project.

In terms of adaptability, the method is successfully employed in different software development projects. We were only able to completely implement the GSRM into one software development project, i.e., the project for the local government ministry. Both case study projects were the development of an information system. Therefore, adaptability was not fully proved. The studies were from the same geographical territory and cultural bias may exist. Internal validity for the first case study was not fully proved. We tried to compare our findings with the existing study results in the literature to generalize our findings. The comparison results give us positive similarities and some unique findings.

7.5 Future Research

GSRM allows the early identification of project specific goals, the break-down of the identified goals into sub-goals and the identification of project risks and strategies to deal with these risks. The biggest advantage of GSRM is that it is easy to use, can capture domain specific knowledge and experience, provide early feedback so that the model can easily be evolved. The layer based abstraction for goal-driven modeling approach is suitable for other domains such as safety critical systems, security or privacy risk management, or applicable to any other goal-oriented undertaking or business domain. However, several limitations have been observed for the proposed approach. A specific guideline is required to tailor the model by following the project specific context.

Another avenue of future research would be to use the past goals and risks data to support the current risk analysis. In particular, how a risk resource repository can be constructed, which can be used for risk analysis in the upcoming projects. This allows reducing the overhead of implementing GSRM when the project is under continuous schedule and budget pressure. We assume complete construction of BBN and goal-risk model can effectively be reused by major or minor modification to a similar project context. Reusability of the goal-driven risk management artifacts certainly increases the applicability of the overall approach. Two case studies were done to empirically investigate GSRM. And both case study projects considered the development of an information system. Replicate case study is necessary to another direction of future work where the validity of the result can be more generalized and revised. A case study in a different domain such as safety or security would perfectly enhance the validity purpose.

During the course of software development goals and risks have evolved. We need to carefully monitor the evolution so that prioritized goals and risks get immediate and adequate attention. Our study results already mentioned that goal details and goal-risk models facilitate to communicate with the management about the uncertainties and risks that exist in the project. But a guideline is necessary for the management to use the goal models. It also supports to decide which part of the model is necessary for the management. GSRM links the goal model with BBN, a detailed examination of the transformation of the goal model to BBN is necessary. A comprehensive investigation of the goals certainly amplifies the contribution of goal-driven approach for the software risk management.

In software projects, there exist always gaps between the project contract and technical specification, software development and the implementation, product's outcomes and the customers' expectations and between product operation and maintenance. We need to systematically look at these gaps and pay attention to minimize them. Some factors within the course of the software product life cycle such as deployment, operation and maintenance issues do not get attention at an early stage. But later these can cause real danger for the project. We define these as unforeseen risks of the project. These unforeseen risks and gaps need further investigation. Future research can also consider a detailed study on continuous risk monitor which address effectiveness of control action and evolution of goals and risks. It supports to discard the control action for the future projects which may not be effective in a running project. Risk management practice sometimes appears to have lower priority. But risk management is an essential and effective practice for a successful software project of any size. Credibility of risk management method and its integration in the decision making process need proper investigation. This allows to create an awareness about risk management practice and its benefits in the software projects.

7.6 General Conclusions

The significance of software is growing everyday as it becomes a major part of enterprise business. At the same time, complexity of the software systems is also increased tremendously. Therefore, building of high quality product in the shortest possible time to satisfy the global market demand gives an enterprise a competitive advantage. However, there exist uncertainties and risks at every stage of the software product. These risks can have an extremely high influence on the success of software projects. Making timely and well informed decision is important for controlling the risks. We also need to convince the project stakeholders for a formal risk management practice which can be tailored based on the project context. This doctoral thesis contributes for a systematic and easy to use risk management practice and enables risk management activities to be explicitly integrated into requirements engineering phase. We believe that goal-driven software development risk management model and its implementation have a strong impact on an effective risk management practice in the software engineering domain.

APPENDIX A

Appendix A: Closed Questions

The purpose of this interview is to collect the states of the development components w.r.t the running project where you are involved.

The collected information only used for the research entitled Software Development Risk management Model-a goal-driven approach and to improve the running project context by effective implementation of risk management result. In case if you are not the best person to answer, please feel free to pass on this questionnaire to other people in your organization. You are answering as an individual and we will only use the information in aggregated form. Your name or other details will not be mentioned if you want. Thanks a lot for your participation.

Project Execution

Project Planning and Control

(Q1) Are all project deliverables and related schedule realistically estimated & agreed with the customer?
☐ No ☐ Not fully realistic ☐ Estimated & agreed

(Q2) Are the estimation factors like product size, code volume, practitioners' capability, project constraints, complexity and reusability considered for the estimation?
☐ Partially ☐ Yes but not all ☐ Yes

(Q3) Is the schedule pressure tight for the project?
☐ Yes ☐ Yes but not for all deliverables ☐ Backup up to certain threshold

(Q4) How do you estimate the development cost?
☐ Neither cost model nor experience ☐ Only experience ☐ Using suitable cost model & experience

(Q5) Are the cost, schedule and effort estimation based on the previous project?
☐ No ☐ Partially ☐ Based on similar past project

(Q6) How frequent are both estimated schedule and cost planned to be revised?
☐ Never ☐ End of every deliverable ☐ Monthly or more frequent basis

(Q7) Is there any monitoring facility that tracks estimated schedule, cost and effort?
☐ No ☐ Yes but not all factors ☐ Yes

(Q8) Up to now how much is the variation of the estimated schedule and cost compared to the actual one (i.e. Over-estimate/under-estimate)?
☐ High ☐ Minor ☐ No

(Q9) Does the project contain a detailed change management plan during development and after delivery?
☐ Partially during development ☐ Partially for both development and delivery ☐ Yes

(Q10) Does the change analysis focus on major project goals and risks?
☐ No ☐ Partially ☐ Yes

(Q11) Does the project consider adequate project management method?
☐ No ☐ Partially ☐ Yes

(Q12) Are the essential technical issues (e.g. tools, language & hardware) identified, available and familiar for the project?
☐ Some ☐ More than some ☐ Yes all of them

(Q13) Does the project need any new technology/platform which is unfamiliar to the development team?
☐ Yes ☐ Not fully unfamiliar ☐ No

(Q14) Does the project contain complex business processes / tasks which are unfamiliar to the development team?
☐ Yes ☐ Partially ☐ No

(Q15) Are the software under development expected to communicate with other existing applications of the customer end?
☐ Yes ☐ Partially ☐ No

(Q16) Does the software under development need specific hardware?
☐ Yes ☐ Partially ☐ No

(Q17) What is the hardware dependency for the project?
☐ Highly platform dependent ☐ Run on some platform ☐ Independent

(Q18) What is the complexity of IO devices?
☐ High ☐ Medium ☐ Low

(Q19) What type of user interface management operation is used in the program?
☐ Complex (multimedia and virtual reality) ☐ Several sets of different types
☐ Simple I/O forms

(Q20) What is the size of the project?
☐ Large ☐ Medium ☐ Small

(Q21) Are there large numbers of user groups involved during the development?
☐ Yes ☐ Not very large group ☐ No

(Q22) Will large numbers of user groups use the product?
☐ Yes ☐ Medium size user groups ☐ No

Project Scope

(Q23) Is the project context similar to some previous projects?
☐ No ☐ To some extent ☐ Yes

(Q24) Are the project goals identified and agreed with customer and practitioners?
☐ Some ☐ More than some ☐ Completely

(Q25) Is the project success criteria realistic and achievable?
☐ Partially ☐ Yes but not all ☐ Yes

(Q26) Are the project boundary conditions clearly identified?
☐ No ☐ Yes but Partially ☐ Yes

(Q27) Is the project contract including terms and conditions approved and signed by customer?
☐ No ☐ Yes but Partially ☐ Yes

(Q28) How risky is the project by considering scope, users, complexity, constraints, reusability, specification, resource, expertise and knowledge?

☐ High ☐ Medium ☐ Low

Tools Support

(Q29) Are the project execution tools for schedule, cost and effort estimation, release management and monitor identified and available for the project?
☐ Partially ☐ More than Partially ☐ Completely

(Q30) Are the project communication and collaboration tools identified and available for the project?
☐ Partially ☐ More than Partially ☐ Completely

(Q31) Are the project development and maintenance tools (i.e., modeling, requirements management, code generation, testing and documentation) identified and available for the project?
☐ Partially ☐ More than Partially ☐ Completely

Process

Development Process

(Q32) Are there model based development methods planned to be used for the project?
☐ No ☐ Yes but partially ☐ Yes

(Q33) Does documented development process exist for the project?
☐ Partially but not documented ☐ Partially documented ☐ Yes

(Q34) Does the development process support adequate activities, steps and methods?
☐ Partially ☐ More than partially ☐ Completely

(Q35) Do the activities and tasks clearly specify input and output of the individual phase?
☐ No ☐ Partially ☐ Yes

(Q36) Is the goal modeling method planned to be used for the project?
☐ Not sure ☐ Partially ☐ Yes

(Q37) Are the current development processes suitable and adequate for the project?
☐ Partially ☐ More than partially ☐ Completely

(Q38) Are roles clearly defined for the individual task under an activity ?
☐ Partially with confusion ☐ Partially ☐ Completely without confusion

(Q39) Does requirements engineering process explicitly consider customer/user involvement?
☐ No ☐ Partially ☐ Yes

(Q40) Does requirements engineering process construct a complete specification document for the project?
☐ No ☐ Partially ☐ Yes

(Q41) Does the architectural design include every subsystem and their interactions?
☐ No ☐ Partially ☐ Yes

(Q42) Does the design construct a complete architectural and detailed design document?
☐ No ☐ Partially ☐ Yes

(Q43) Is there any coding standard followed for the project?
☐ No ☐ Partially ☐ Yes

(Q44) Does the test plan consider project goals and requirements?
☐ No ☐ Partially ☐ Yes

Risk Management

(Q45) Does the project consider a formal risk management practice?
☐ No ☐ Yes but partially ☐ Yes

(Q46) Does a comprehensive risk management process exist for the project?
☐ No ☐ Yes but very informal ☐ Yes

(Q47) Are the risk management roles clearly identified and assigned?
☐ Partially ☐ More than partially ☐ Completely

(Q48) Is the risk management process integrated into the development process?
☐ No ☐ Yes but unclearly ☐ Yes

Consistency and Dependency

(Q49) Are the development activities consistent?
☐ Not sure ☐ Partially ☐ Yes

(Q50) Do there supporting processes(i.e., documentation, configuration, reviews and audits) exist to support the primary development and risk management process?

☐ No ☐ Partially ☐ Yes

(Q51) Are the artifacts from different activities synchronized and dependent upon each other?
☐ No ☐ Partially ☐ Yes

(Q52) Does the project need a high degree of concurrency among the activities?

☐ Yes ☐ Partially ☐ No

(Q53) Do the development activities consider any traceability link from requirements to test through design and code?
☐ Partially ☐ More than partially ☐ Completely

(Q54) Are the requirements contain sufficient details for design and code?
☐ No ☐ Yes but unclearly ☐ Yes

Process Compliance

(Q55) How closely the existing processes are planned to be followed for the project?
☐ Not closely ☐ Partially followed ☐ Completely

(Q56) Are the development and risk management processes effective in achieving the required result?
☐ No ☐ Partially ☐ Completely

(Q57) Do the development and risk management processes periodically assess to measure the productivity and overall purpose w.r.t. the organization context?
☐ Partially ☐ More than partially ☐ Completely

(Q58) Do the organization follow any procedure to internally audit the existing processes?

☐ No ☐ Yes but very infrequently ☐ Yes but infrequently

(Q59) Do the audit results used to adjust the existing processes?

☐ No ☐ Occasionally ☐ Yes

(Q60) Do the practitioners reluctant to follow the process to complete their roles and responsibilities?
☐ Yes ☐ Partially ☐ No

Product

Specification

(Q61) Are the business goals and objectives identified for the project?
☐ Partially ☐ More than partially ☐ Completely

(Q62) Are the business processes, domains, uses cases and future user groups identified for the project?
☐ Partially ☐ More than partially ☐ Completely

(Q63) Are the mandatory business rules (e.g. laws, industry standards and related policies) identified & structured for the project?
☐ No ☐ Partially ☐ Yes

(Q64) Does the project identify system vision and agreed with customer?
☐ Partially identified but not agreed ☐ Partially identified & agreed
☐ Identified & agreed

(Q65) Does the project scope clearly support the business needs and add business value to the overall customer's business environment?
☐ No ☐ Partially but not sure ☐ Yes

Requirements Faults

Please if necessary follow the given requirements error checklist.

(Q66) Do the requirements provide different ambiguous interpretations or lack of support for rationale?
☐ Highly ☐ Partially ☐ Rarely

(Q67) How many ambiguous terms within the specification document?
☐ Many ☐ Some ☐ Less than some

(Q68) Are requirements within the project scope?
☐ No ☐ Partially ☐ Not fully

(Q69) Do the requirements provide incorrect meaning and contain conflicting terms with other artifacts?
☐ More than partially ☐ Partially ☐ No

(Q70) Until now, how much percent of the requirements changed over a given period of the time?
☐ High ,i.e., $\geq 70\%$ ☐ Medium,i.e., $\geq 30\%$ ☐ Low ,i.e., $\leq 30\%$

(Q71) Are the early stage requirements frozen before the subsequent development phases?
☐ Many expected to be changed ☐ Some ☐ Mostly

(Q72) Is there any mechanism used to perform impact analysis due to the change of specific single or multiple requirements?
☐ Rarely ☐ Sometimes ☐ Yes

(Q73) Are the system requirements (e.g. functional, quality, architectural and operational) completely identified?
☐ Partially ☐ Yes but not all ☐ Completely

(Q74) Are the user requirements (i.e. use cases, scenarios & features) properly specified?
☐ No ☐ Partially ☐ Yes

(Q75) Are informal and unwritten user expectations considered in the final specification?
☐ No ☐ Partially ☐ Yes

(Q76) Are the requirements traceable to downstream phases, i.e., design, code, test or deployment?
☐ No ☐ Some of them ☐ All

(Q77) Are requirements traceable to its source such as business specification, project scope and user expectation?
☐ No ☐ Some of them ☐ All

(Q78) Is there any requirement which is not possible to implement within existing technology?
☐ Yes many ☐ Some ☐ None

(Q79) Are the factors i.e., goal, scale, minimum and maximum acceptable value difficult to measure for the quality requirements?
☐ Yes some of them ☐ Very less ☐ None

(Q80) Is there any requirement which incurs excessive operational needs or causing additional system cost?
☐ Yes some ☐ Very less ☐ None

(Q81) Are all requirements categorized and prioritized?
☐ Yes some ☐ Most of them ☐ All

(Q82) Are the right customer or user representatives involved in the RE process?
☐ No ☐ Some ☐ Yes

(Q83) Do you follow any standard (template, notations and checklist) for the specification?
☐ No ☐ Partially ☐ Yes

(Q84) Are the requirements completely documented and agreed with the user?
☐ Some of them ☐ Most of them ☐ All

Time-to-market

(Q85) Is time-to-market the one of the primary product goals?
☐ Yes ☐ Partially ☐ No

(Q86) Does the project consider to analyze the relevant customer market (market size, trends, segmentation) at project planning phase?
☐ No ☐ Partially ☐ Yes

(Q87) Does the project contain features which make the product different from similar products in the market?
☐ No ☐ Partially ☐ Yes

(Q88) Does the project consider competitor analysis (i.e., transition cost, product differentiation and total purchase cost) of the similar product?
☐ No ☐ Partially ☐ Yes

(Q89) Does the project consider consumer analysis (i.e., groups, user behavior, demand and link product design with user needs) for the product?
☐ No ☐ Partially ☐ Yes

(Q90) Does the project consider customer satisfaction factors (usability, performance, reliability, maintainability, performance) for the product?
☐ No ☐ Partially ☐ Yes

(Q91) Are the product requirements stable to support time-to-market of the product?
☐ No ☐ Partially ☐ Yes

Quality and Testing

(Q92) Is quality assurance and management planned and implemented for the project?
☐ No ☐ Partially ☐ Completely

(Q93) Is the overall product performance (i.e., throughput, real time response, recovery time and access) identified, quantified and agreed with the user?
☐ Partially ☐ More than partially ☐ Yes

(Q94) Will the software-to-be developed be integrated into the existing system of the customer premises?
☐ Yes ☐ Partially ☐ No

(Q95) Does the project include a comprehensive test specification?
☐ No ☐ Partially ☐ Yes

(Q96) Have possible unit and integration test cases specified?
☐ No ☐ Partially ☐ Yes

(Q97) Have possible system test cases specified?
☐ No ☐ Partially ☐ Yes

(Q98) Does the project consider adequate acceptance testing before delivery the product?
☐ No ☐ Partially ☐ Yes

(Q99) Are the test results planned to be reviewed during the development?
☐ No ☐ Partially ☐ Yes

(Q100) Does the project consider easy-to-use and user friendly final product?
☐ No ☐ Partially ☐ Yes

(Q101) Does the project consider certain usability features such as window, icon and menu?
☐ No ☐ Partially ☐ Yes

(Q102) Does the project consider security properties (i.e., confidentiality, integrity, availability, non-repudiation and authentication) from the early development?
☐ Partially ☐ More than partially ☐ Completely

(Q103) Does the project consider security threats and risks of the critical assets?
☐ Partially ☐ More than partially ☐ Completely

(Q104) Are the security requirements identified and analyzed for the product?
☐ No ☐ Partially ☐ Yes

(Q105) Does the project consider privacy properties (i.e., consent, notice, purpose, participation, anonymity and undetectability) to process and manage sensitive information by the software-to-be developed?
☐ Partially ☐ More than partially ☐ Completely

(Q106) Does the project consider privacy harms (i.e., disclose, unawareness, storage and transfer) and risks which violate privacy of the sensitive information?
☐ Partially ☐ More than partially ☐ Completely

(Q107) Are the privacy requirements identified and analyzed for the product?
☐ No ☐ Partially ☐ Yes

(Q108) Does the project consider all critical hazards and associated severity levels relevant for the system-to-be?
☐ Partially ☐ More than partially ☐ Yes

(Q109) Does the project consider the causes of all critical failure factors such as hardware errors, human mistakes during operation and maintenance, interface problems and timing problems of the system-to-be?
☐ Partially ☐ More than partially ☐ Yes

(Q110) Is there any correlation considered from security violation to the software failure?
☐ No ☐ Partially ☐ Yes

(Q111) Are the safety requirements identified and analyzed for the product?
☐ No ☐ Partially ☐ Yes

(Q112) Do the safety requirements consider the result of hazard analysis?
☐ No ☐ Partially ☐ Yes

(Q113) Does the project consider a complete user manual with the software?
☐ Incomplete user manual ☐ Un-verified user manual ☐ User manual will be developed, tested & delivered

(Q114) Does the manual follow any documentation standard (e.g., IEEE 1063-1987) for preparing the manual?
☐ No ☐ Partially ☐ Yes

(Q115) Does the manual preparation consider different levels of target audience (i.e. novice, intermediate, information services staff, engineers, expert and management)?
☐ No ☐ Partially ☐ Yes

Operation

(Q116) Do the users maintain any documented process to implement and deploy the software into their premises?
☐ No ☐ Partially ☐ Yes

(Q117) Are there any acceptance criteria for the system operation?
☐ No ☐ Yes but not clearly ☐ Yes

(Q118) Are all the acceptance tests planned to be performed ?
☐ No ☐ Some but not all ☐ Yes

(Q119) Do the development team plan to perform comprehensive interviews with the users related to the system implementation?
☐ No ☐ Partially ☐ Yes but not comprehensively

(Q120) Do the operational requirements exist in the system specification?
☐ No ☐ Partially ☐ Yes

(Q121) Does the project consider any user support or problem reporting & resolution activity during operation?
☐ No ☐ Partially ☐ Yes

(Q122) Are the users involved during the system deployment identified and are roles defined?
☐ No ☐ Will consider later ☐ Yes

(Q123) Is the users' training planned and scheduled before the new system physically deployed?
☐ No ☐ Will planned later ☐ Yes

(Q124) Will the training plan focus to completely trained the future system users?
☐ No ☐ Only selected user ☐ Yes

(Q125) Will the technical or operational documentation and supporting training manual planned to be available before the actual training takes place?
☐ No ☐ Not sure ☐ Yes

(Q126) Is there any training evaluation as feedback from the user planned to be considered after the training ?
☐ No ☐ Not sure ☐ Yes

(Q127) Will the required resources in the user environment be available to implement the system?
☐ Very limited resource ☐ Not all available ☐ Yes

(Q128) Will the activities related to the product implementation be planned to be synchronized?
☐ No ☐ Partially ☐ Yes

(Q129) Does there any installation procedure exist for the deployment?
☐ No ☐ Partially ☐ Yes

(Q130) Will the existing user's data be migrated and initialized for the new system?
☐ No ☐ Partially ☐ Yes

(Q131) Do there any checklist plan to assess the proper implementation of the system within the user's operating environment?
☐ No ☐ Partially ☐ Yes

Maintenance

(Q132) Does the project scope include maintenance?
☐ Yes ☐ Partially ☐ No

(Q133) Is the scope for maintenance clearly defined?
☐ No ☐ yes but not clearly ☐ Yes

(Q134) Does the project consider a concrete deliverable phase due to the maintenance?
☐ No ☐ yes but not clearly ☐ Yes

(Q135) Does any maintenance specification(i.e. change and requirements) exist for the maintenance?
☐ No ☐ Partially ☐ Yes

(Q136) Is there any emergency release necessary during the maintenance?
☐ Yes ☐ Partially ☐ No

(Q137) Is there any acceptance criteria necessary once the maintenance activities carried out?
☐ No ☐ Partially ☐ Yes

(Q138) Does the project need major product and document release after the maintenance?
☐ Yes ☐ Partially ☐ No

(Q139) Does the customer organization consider maintenance procedure?
☐ No ☐ unclearly ☐ Yes

(Q140) Are the budget, schedule and roles clearly defined for the maintenance?
☐ No ☐ Will plan later ☐ Yes

(Q141) Will the overall complexity of the software be increased (i.e., more complex structure of individual component, or several new components) due to the maintenance?
☐ Yes ☐ Not sure ☐ No

(Q142) Can the planned maintenance operation increase the overall volatility of product?
☐ Yes ☐ Not sure ☐ No

(Q143) Is the consistency between code and documentation considered during the maintenance?
☐ No ☐ Partially ☐ Yes

(Q144) Does the project plan to identify and eliminate the dead and clone code?
☐ No ☐ Not sure ☐ Yes

(Q145) Is there any negative affect of maintenance to the overall code quality?
☐ Yes ☐ Not sure ☐ Yes

(Q146) Do the individual modified system components plan to be retested due to the maintenance?
☐ No ☐ Still no plan ☐ Yes

(Q147) Does the user acceptance plan to be retested before delivering the maintained product?
☐ No ☐ Still no plan ☐ Yes

(Q148) Will the overall product performance and resource utilization be analyzed during the maintenance?
☐ No ☐ Partially ☐ Yes

(Q149) Will the project document and user manual be updated once the maintenance operation is being completed?
☐ No ☐ Partially ☐ Yes

Human

Motivation & Domain Knowledge

(Q150) Are the practitioners motivated and committed to the project?
☐ Some of them ☐ Almost all ☐ Yes

(Q151) How is the overall relevant domain knowledge of the development team?
☐ Not much ☐ Less than adequate ☐ Adequate

(Q152) Are the practitioners already trained and experienced with tools, language and development environment?
☐ Some of them ☐ Most of them ☐ All

(Q153) Are the project members already familiar with the development process?
☐ Some of them ☐ Most of them ☐ All

(Q154) Are the project members already familiar with the risk management process?
☐ Some of them ☐ Most of them ☐ All

(Q155) Does the development team contain adequate technical knowledge of the project?
☐ No ☐ Yes but unsatisfactory ☐ Yes and satisfactory

(Q156) What is the level of technical expertise between the development team and user?
☐ Lower by development team ☐ Almost same ☐ Higher by development team

(Q157) Does the development team contain adequate expertise about the customer business environment?
☐ No ☐ Yes but unsatisfactory ☐ Yes and satisfactory

Customer/user

(Q158) What is the level of involvement of customer / user until now?
☐ Passive ☐ Occasional ☐ Active

(Q159) How do you evaluate the customer/user relevant project domain knowledge?
☐ Inadequate ☐ Less than adequate ☐ Adequate

(Q160) Do the users have adequate IT competency?
☐ Inadequate ☐ Less than adequate ☐ Adequate

(Q161) Are the key users of the different user groups involved in the development activities?
☐ No ☐ Some ☐ Yes

(Q162) Are the users actively involved and effectively collaborated to the requirements engineering?
☐ No ☐ Partially ☐ Yes

(Q163) What is the level of confidence of customer/user over the development team?
☐ Low ☐ Medium ☐ High

(Q164) Do you think customers/ users have realistic expectations about the project?
☐ No ☐ Partially ☐ Yes

(Q165) Do you think customers/ users support to implement the control action of critical risks in their environment?
☐ No ☐ Partially ☐ Yes

Team Work

(Q166) Are all the development team members identified and available?
☐ Some ☐ Most of them ☐ Yes

(Q167) Is the development team adequately balanced of technical, project domain and management expertise?
☐ No ☐ Partially ☐ Completely

(Q168) How did you evaluate the overall development team performance?
☐ Poor with frequent unhealthy arguments ☐ Average and healthy arguments ☐ Satisfactory

(Q169) Do the distributed development sites (if any) maintain adequate coordination?
☐ No ☐ Partially ☐ Yes

(Q170) Is the development team capable to work with uncertain objective and top management?
☐ No ☐ Less than partially ☐ Partially

Communication & Coordination

(Q171) Is there adequate communication between practitioners and users?
☐ No ☐ Partially ☐ Yes

(Q172) Is there frequent miscommunication between practitioners and users?
☐ Yes ☐ Sometimes ☐ No

(Q173) Are the conflicts with the customer or users resolved in a timely manner ?
☐ No ☐ Sometimes ☐ Yes

(Q174) Do the key users often provide feedback to the development team?
☐ Very rare ☐ Sometimes ☐ Yes

(Q175) Are the users considered as an integral part of the project?
☐ No ☐ Partially and not all aspect ☐ Yes

(Q176) Does the development team face problem due to large number of users' involvement?
☐ No ☐ Partially ☐ Yes

Management

(Q177) How much capable is the project manager to lead the project successfully?
☐ Unreliable ☐ Partially reliable ☐ Reliable

(Q178) Are all roles related to project management, risk management, development, & release management identified and assigned?
☐ Some ☐ Not all ☐ Yes

(Q179) Is every staff of the distributed site clear about his/her roles and responsibilities?
☐ Confused ☐ Fairly understood ☐ Clearly understood

(Q180) Does the management has appropriate leadership quality?
☐ Lack ☐ Average ☐ Adequate

(Q181) What is the level of management commitment (e.g. direction & support) to overall project activity and critical situation?
☐ Unreliable ☐ Partially reliable ☐ Reliable

(Q182) How is the confidence of development team on management?
☐ Low ☐ Medium ☐ More than medium

(Q183) Does the management communicate critical problems with the development team?
☐ No ☐ Sometimes ☐ Yes

(Q184) Does the management support the continual improvement of the development processes and development facilities?
☐ No ☐ Less frequently ☐ Frequently

(Q185) Does the management always try to achieve customer satisfaction goal?
☐ No ☐ Not always ☐ Yes always

(Q186) Does the management motivate the practitioner for effective and creative work?
☐ Rarely ☐ Sometimes ☐ Frequently

Internal & External Environment

Organizational Stability

(Q187) Does the management agree to trade additional budget and adequate support for handling project risks?
☐ No ☐ Not willingly ☐ Yes

(Q188) Are the policies and procedure approved and implemented within the organizations?
☐ Some policies exist but not implemented ☐ Policies exist & partially implemented ☐ Yes Completely

(Q189) Is the organizational structure stable and documented?
☐ Frequent change ☐ Change in certain interval ☐ Yes

(Q190) Is the organization always change its operational environment?
☐ Yes ☐ Change in certain interval ☐ Yes but for effective output

(Q191) Does the organization support the project?
☐ No ☐ Partially ☐ Yes

(Q192) Is the top management capable and not conservative in decision making?
☐ Not capable & highly conservative ☐ Partially & moderate conservative
☐ Almost with timely manner & progressive

(Q193) Does the organization has adequate control to the subcontractors & service providers (if any)?
☐ No ☐ Partially ☐ almost full control

(Q194) Do there exist organizational life cycle processes (i.e., management, infrastructure, improvement and training)?
☐ No ☐ Not sure ☐ Partially

(Q195) Does the organization support adequate and frequent training facilities?
☐ No ☐ Infrequently ☐ Yes but not adequately

(Q196) Is the organization highly politically biased ?
☐ Yes ☐ Medium ☐ No

Development & Communication Facility

(Q197) Are there adequate work stations, storage and processing capacity for the software project?
☐ Partially ☐ More than partially ☐ Yes

(Q198) Are all the development and project management tools available ?
☐ Some of them ☐ Almost all of them ☐ Yes

(Q199) Do there adequate infrastructure facilities (e.g. power, work space, Internet, telephone) exist to work and communicate with customer /user & other distributed development site?
☐ No ☐ Partially ☐ Almost

(Q200) Is all the necessary hardware available when needed?

☐ No ☐ Not always ☐ Yes

Thank your once again for your cooperation.

APPENDIX B

Appendix B: Open Questions

The purpose of this interview is to obtain feedback on Goal-driven Software Development Risk Management Model(GSRM). Your comments will be very much vital for this research. Therefore we would like to request you to answer the open questions from your understanding (i.e. positive or negative) about the implementation of GSRM into the project. The collected information will be used only for the research purpose. Thanks for your cooperation

Risk Management Practice

(Q1) What are the techniques commonly used for risk management in the projects you already involved?

(Q2) Is the risk management process integrated into software development process or is it a separated process?

(Q3) What are the main reasons of not having a formal risk management practice in the software project (based on your experience from the project that you have participated)?

(Q4) What are the main reasons of software project failure to meet its project specific goals (based on your experience from the project that you have participated) ?

(Q5) What are the main reasons for the requirements problems (based on your experience from the project that you have participated)?

(Q6) What skill and competence exist by the project personnels in the current project to perform risk management activities?

(Q7) How much effort would you prefer for an effective risk management practice?

(Q8) If the project is by inherent nature high risky, then which part of risk management is the most important from your opinion and which artifacts of the development you focus more?

(Q9) What type of risk culture is followed by your company for software development, deployment, operation and maintenance?

Overall Evaluation of GSRM

(Q10) Is goal-driven approach for risk management useful in software development project?

(Q11) What are the problems you observed because of using goal-driven approach for software risk management?

(Q12) Is it feasible to integrate risk management activities in requirements engineering (Please specify the overhead of the integration) ?

(Q13) Do you think the process used for GSRM is appropriate and adequate for managing risk in software project?

(Q14) Do you think the artifacts of GSRM are appropriate and realistic for controlling the risks in software projects?

(Q15) Do the artifacts clearly represent the goal and risk information and effectively communicate the risk information with the management?

(Q16) Which techniques of individual activities of GSRM are useful or not-useful from your opinion and why?

(Q17) Do you think the closed questions are practical to identify factors which pose for the potential risk event from the perspective of a holistic view?

(Q18) Which part of the GSRM is the most difficult based on your experience in the running project?

(Q19) How much effort is consumed by GSRM to perform a complete risk management practice in the running project?

(Q20) How much reliable are the identified control actions by GSRM to prevent or reduce the identified risk w.r.t the project context?

(Q21) What are the overall impacts of risk management in the software project?

(Q22) What are the complexities caused by GSRM?

(Q23) Can practitioners easily learn the basic of GSRM and its activities and artifacts without having a detail knowledge of goal model?

(Q24) Do you think GSRM can be applicable in any software project domain?

(Q24) How does GSRM differ from your existing risk management practice?

(Q26) Do you think GSRM support to achieve the main goals of the software project?

(Q27) How can the result of GSRM from the running project be effectively used for the upcoming projects?

(Q28) Do you have any recommendation to improve GSRM?

(Q29) Is the integration of risk management into early software project important?

(Q30) What are the criteria necessary to consider the integration of risk management into software development or management process?

APPENDIX C

Appendix C: Checklist for Requirements Errors

This appendix provides a checklist of identifying errors of the elicited requirements. However depending on the project context, product scope, available resources and knowledge, nature of the errors can vary a lot. The checklist consists of root cause examples, test criteria, assessment strategy and mitigation points. The purpose of the checklist is only to provide a guideline to handle the error commonly exist in requirements. This checklist is a part of the goal-driven software development risk management model implementation into the project.

Ambiguity Requirements Table C.1 shows the ambiguity requirements checklist.

Table C.1: Ambiguity requirements checklist

Root causes examples ❑ Overlap, imprecise and insufficient details in requirements which influence to fill the blanks by assumption. ❑ Different interpretation is given for the same requirement. ❑ Subjective meaning of requirement rather than objective. ❑ Some examples of ambiguity phases are: vague subjects refer to multiple things such as pronouns(i.e.,it, they), vague adjectives (i.e., always, soft, hard,strong, weak, sufficient, useful,user friendly), vague prepositions(i.e., above, below, under).
Test criteria ❑ Does the requirement provide different interpretation by different reviewers? ❑ How many places ambiguous phrases are included within the requirement specification?
Assessment approach ❑ (highly, partially, rarely)/ (many, some, very few)
Mitigation points ❒ Ambiguity phrases should be revised from the requirement specification. ❒ Clear and simple language construction should be followed for requirement document. ❒ A common glossary to clearly defined business, technical & domain specific terms (e.g., security, privacy, safety).

Incomplete Requirements Table C.2 shows the incomplete requirements checklist.

Table C.2: Incomplete requirements checklist

Root cause examples
❑ Missing constraints such as linkup with other facilities & requirements. ❑ Missing external interfaces and critical quality attributes(i.e., performance & security) user classes and instances (i.e., use case, derive value) and instance element(i.e., feature). ❑ Lack of effort to fix the incomplete part due to budget and schedule constraint. ❑ Inadequate user involvement and lack of skill within RE activities. ❑ Stakeholders sometime assume some parts of the requirement is obvious without saying or implicit expectation that influences for under-specified requirements.
Test criteria
❑ Are the use cases complete (i.e., valid/invalid inputs, outputs, trigger & post conditions, alternative courses, exceptions and boundary values)? ❑ How many places of the specification are marked with to be determined (TBD) or to be added (TBA)? ❑ Does the specification contain (sections & sub-sections, page number, figures & tables)? ❑ Are all user groups and classes identified?
Assessment approach
❑ (no, partially, yes)/ (many, some, no)
Mitigation points
❒ Use cases should be completed and specify response for valid & invalid inputs and outputs, address trigger & post conditions, normal & alternative courses, exceptions, boundary values by adopting standard use case templates. ❒ Requirements elicitation requires collaboration among domain experts(i.e., maintenance, performance, safety, security) and representatives from all the user groups. ❒ Functional, quality, data and interface requirements are required to contain necessary information about what it should do and should not do. ❒ Section marked with TBD or TBA need to be revised. ❒ All system requirements required to be identified and documented.

Incorrect Requirements Table C.3 shows the incorrect requirements checklist.

Table C.3: Incorrect requirements checklist

Root causes examples ❑ Requirement fails to specify actual needs of stakeholders and business context. ❑ Incorrect assumption, inappropriate bias, insufficient knowledge by user & requirement engineer. ❑ Wrong users' involvement in RE who do not know the actual needs.
Test criteria ❑ Do the requirements align with goals and business specification? ❑How many mistakes or wrong parts exist in specification?
Assessment approach ❑ (yes, partially, no)/ (many, some, very few)
Mitigation points ❒ Scenarios and test cases can be used to verify the requirements. ❒ Active user involvement and effective feedback are essential.

Unverifiable Requirements Table C.4 shows unverifiable requirements checklist

Table C.4: Unverifiable requirements checklist

Root cause examples ❑ Failure to verify whether the system meets the requirement. ❑ Contradictory characteristics within requirements dependency metrics. ❑ Inadequate facilities to meet the key users.
Test criteria ❑ Do there exist possible cost effective attributes to verify the requirements? ❑ Is it possible to generate test cases from individual or group requirements?
Assessment strategy ❑ (no, often, yes)/(no, yes but difficult, yes) **Mitigation points** ❒ Non measurable quantities & ambiguous terms should be revised from the requirement. ❒ High level requirements require adequate refinement to generate test cases.

Unstable Requirements Table C.5 shows the unstable requirements checklist.

Table C.5: Unstable requirement checklist

Root cause examples ❏ Unclear definition of product scope, long development life cycle, errors & inconsistencies within requirements specification and high initial specification. ❏ Nature of requirement (i.e., emergent, assumption, compatibility). ❏ Change of external interfaces (e.g., business needs change, new competitors, new technologies , new or changed laws and legislations and customer priorities). ❏ Failure to manage the change(e.g., change rate too high, scope of change too big, too expensive to implement the change). ❏ New requirements are added at the late phase or numerous user requests. ❏ Project scope grows in an uncontrolled way such as adding new features or process and including more functions.
Test criteria ❏ How much percentage of the requirement changes over a given period of time? ❏ How many place includes variable domain?
Test criteria ❏ (high, medium, low)/(many, some, none)
Mitigation points ❐ Variable domain parts need to be revised. ❐ Well defined policy to support change management. ❐ Project management should accommodate space for any growth of requirements change. ❐ New requirements need to be verified w.r.t. system vision and project scope. ❐ System vision, features, system boundary, success criteria need to be clearly defined within standard template and agreed with the customer.

Data Requirements Table C.6 shows the data requirements checklist.

Table C.6: Data requirements checklist

Root cause examples ❏ Unknown and non standard data require to implement the requirement. ❏ Unavailable data type or object, out of range data within the existing data type.
Test criteria ❏ Are the basic data requirements factors such as data type, possible range, data volume, legitimate operations identified and verified for the requirements?
Assessment strategy ❏ (no, partially, yes)
Mitigation points ❐ Data requirement must be completed with its type, semantics, volume, update frequency, legitimate operations on data and relationship with other data and relevant information.

Untraceable Requirements Table C.7 shows untraceable requirements checklist.

Table C.7: Untraceable requirements checklist

Root cause examples ❑ Unable to trace the requirements to downstream artifacts such as architecture, design elements and test sets. ❑ Unable to trace the requirement back to its origin such as business specification, regulations and product vision. ❑ A large number of requirements make tracing difficult. ❑ Fail to uniquely and correctly identify requirements.
Test criteria ❑Is any tracing element or link identified from requirement to source documents? ❑ Is any tracing element or link identified from requirement to design, code, testing?
Assessment approach ❑ (yes, partially, no)
Mitigation points ❒ Requirements require to be uniquely identified. ❒ Well defined policy and tool support (if possible) for the tracing. ❒ Tracing matrix should clearly specify elements and links to trace from requirements to both downstream and upstream development phases.

Infeasible Requirements Table C.8 shows infeasible requirements checklist.

Table C.8: Infeasible requirements checklist

Root cause examples ❑ Difficult to implement requirement and its related features within available technology (hardware or software) and estimated schedule and cost. ❑ Requirements may be individually feasible but combined with other requirements might be difficult to implement.
Test criteria ❑ Is it feasible to implement the requirement with existing technologies, language, tools and hardware ? ❑ Are experience personnels available to verify the requirements?
Assessment strategy ❑ (no, partially, yes)/ (readily, usually but shared and unavailable)
Mitigation points ❒ Feasibility analysis needs to be performed for the critical requirements . ❒ Test cases might be generated to identify feasibility of the requirements.

Inconsistent Requirements Table C.9 shows inconsistency requirements checklist.

Table C.9: Inconsistent requirements checklist

Root cause examples ❏ Conflicts among requirements and contradictory characteristics by requirements dependency metrics. ❏ Requirements emphasized more on design by stating how to solve the problem. ❏ When a single requirement addresses multiple product or unnecessary features
Test criteria ❏ Does the requirements support possible product features? ❏ Are there any logical conflicts or inconsistency among terms used in the requirements? ❏ Does there exist any conflict between specification with real world objects ?
Assessment strategy ❏ (no, partially, yes)/ (yes, yes to some extent, no)
Mitigation points ❒ Conflicting terms, characters and logics should be modified in the requirements . ❒ Single requirement should not support more than one product feature . ❒ Requirement must initially emphasize on what the system should and should not do, rather than providing any design and implementation solution.

Requirement Quantification Table C.10 shows requirements quantification check list.

Table C.10: Requirement quantification checklist

Root cause examples ❑ Fail to quantify quality attribute such as performance, reliability, usability, & security. ❑ Qualitative terms (i.e., easy, fast, reliable, secure, scalable, efficient and robust) sometimes influence misunderstandings between product and stakeholders. ❑ Project team members reluctant to precisely specify minimum or maximum acceptable amount.
Test criteria ❑ Is the quality profile (e.g., scale, method, minimum and acceptable level, goal and range) identified and analyzed ?
Assessment strategy ❑ (no, partially, yes)
Mitigation points ❒ Factors to quantify requirement such as scale(i.e., measuring unit of quantify statement), meter(i.e., method, frequency, source), must (i.e.,minimum level), wish (i.e.,desirable level to achieve), trend (i.e., historical range), record (i.e., best known achievement) should be identified and evaluated for requirements. ❒ Quality factors identification should derive from user expectation, regulations, business specification and quality goal.

Non-Compliance Requirements Table C.11 shows non-compliance requirements checklist.

Table C.11: Non-compliance requirement checklist

Root causes ❑ Requirement conflicts with key regulatory elements(such as legal rights, correlatives and constraints) and with the existing company policies and procedures. ❑ Requirement influences for non-compliance issues .
Test criteria ❑ Does the requirement conflict with laws and regulation?
Assessment strategy ❑ (yes, partially, no)
Mitigation points ❒ Key elements from relevant regulation,industry standard are required to identify based on product scope. ❒ Elicited requirements should be refined to align with derived key concepts(legal rights, correlatives, obligations and policies) from the regulations.

Bibliography

[ADH$^+$96] C. J. Alberts, A. J. Dorofee, R. Higuera, R. L. Murphy, J. A. Walker, and R. C. Williams. *Continuous Risk Management Guidebook*. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA., 1996.

[AER02] A. I. Antón, J. B. Earp, and A. Reese. Analyzing website privacy requirements using a privacy goal taxonomy. In *RE '02: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, pages 23–31, Washington, DC, USA, 2002. IEEE Computer Society.

[AG06] Y. Asnar and P. Giorgini. Modelling risk and identifying countermeasures in organizations. In *In: Proceedings of 1st International Workshop on Critical Information Infrastructures Security (CRITIS 06). Volume 4347 of LNCS*, pages 55–66. Springer, 2006.

[AG07] Y. Asnar and P. Giorgini. Risk analysis as part of the requirements engineering process. Technical report, Technical Report DIT-07-014, Department of Information and communication Technology, University of Trento, 2007.

[AL04] A.K.T.Hui and D.B. Liu. A bayesian belief network model and tool to evaluate risk and impact in software development projects. In *Annual Symposium on Reliability and Maintainability(RAMS)*, 2004.

[AMN07] N. H. Arshad, A. Mohamed, and Z. M. Nor. Risk factors in software development projects. In *SEPADS'07: Proceedings of the 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, pages 51–56, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

[AMV06] W. Asprayn, F. Mayadas, and M. Y. Vardi. Globalization and offshoring of software: A report of the acm job migration task force. Technical report, ACM, NY, 2006.

[Arr57] K. J. Arrow. Decision theory and operations research. *Operations Research*, 5, No. 6:765–774, December 1957.

[AS499] Standard australia, as/nzs 4360:1999 risk management, 1999.

[BAB09] A guide to the business analysis body of knowledge (babok guide). International Institute of Business Analysis (IIBA), March 2009.

[Ban08] P. L. Bannerman. Risk and risk management in software projects: A reassessment. *The Journal of Systems and Software*, 81(12):2118–2133, 2008.

[Bas] Bangladesh association of software & information services (basis). http://www.basis.org.bd (last accessed October 2010).

[BB01] B. Boehm and V. Basili. Software defect reduction top 10 list. *Computer*, 34(1):135–137, 2001.

[BC01] T. Bedford and R. Cooke. *Probabilistic Risk Analysis. Foundations and Methods*. Cambridge University Press, 2001.

[BEK$^+$98] B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy. Using the winwin spiral model: A case study. *Computer*, 31(7):33–44, 1998.

[BFH08] K. Boness, A. Finkelstein, and R. Harrison. A lightweight technique for assessing risks in requirements analysis. *IET Software*, 2(1):46–57, 2008.

[BFI$^+$09] M. Broy, A. Fleischmann, S. Islam, L. Kof, C. Leuxner, K. Lochmann, D. Mendez-Fernandez, B. Penzenstadler, W. Sitou, and S. Winter. Towards an integrated approach to requirement engineering. Technical report, Technical Report, TUM-I0935,Technische Universitt Mnchen, December 2009.

[Boe81] B. W. Boehm. *Software Engineering Economics*. Prentice Hall PTR, 1981.

[Boe91] Barry W. Boehm. Software risk management: Principles and practices. *IEEE Software*, 8(1):32–41, 1991.

[Bor05] Borland. Mitigating risk with effective requirements engineering. Technical report, White paper, Borland, 2005.

[BP01] B. Boehm and D. Port. Educating software engineering students to manage risk. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 591–600, Washington, DC, USA, 2001. IEEE Computer Society.

[BPG$^+$04] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8:203–236, 2004.

[Bro95] F.P. Brooks. *The Mythical Man-Month: Essays on Software Engineering,*. Addison-Wesley Professional, 1995.

[Bro06] M. Broy. Requirements engineering as a key to holistic software quality. In *Computer and Information Sciences ISCIS 2006, Lecture Notes in Computer Science*, 2006.

[BRT93] H. Barki, , S. Rivard, and J. Talbot. Toward an assessment of software development risk. *Journal of Management Information Systems*, 10(2):203–225, 1993.

[BWHW05] C. Braun, F. Wortmann, M. Hafner, and R. Winter. Method construction - a core approach to organizational engineering. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1295–1299, New York, NY, USA, 2005. ACM.

[Car99] E. Carmel. *Global software teams: collaborating across borders and time zones*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[CAW97] R. N. Charette, K. M. Adams, and M. B. White. Managing risk in software maintenance. *IEEE Software*, 14(3):43–50, 1997.

[CD97] T. Connolly and D. Dean. Decomposed versus holistic estimates of effort required for software writing tasks. *Management Science*, 43(7):1029–1045, 1997.

[CDN88] J. P. Chin, V. A. Diehl, and K. L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–218, New York, NY, USA, 1988. ACM.

[CH93] C. Chittister and Y. Haimes. Risk associated with software development: A holistic framework for assessment and management. *IEEE Transactions on Systems, Man and Cybernetics*, 23, 1993.

[Cha99] R. N. Charette. The competitive edge of risk entrepreneurs. *IT Professional*, 1(4):69–73, 1999.

[Cha05] R.N. Charette. Why software fails? *IEEE Spectrum*, 2005.

[CKM$^+$93] M. Carr, S. Konda, I. Monarch, C. Ulrich, and C. Walker. Taxonomy based risk identification (cmu/sei-93-tr-6, ada266992). Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA., 1993.

[COS04] COSO. Enterprise risk management - integrated framework. Technical report, Committee of Sponsoring Organizations of the Treadway Commission, 2004.

[Dav93] Alan M. Davis. *Software Requirements: Objects, Functions and States*. Prentice Hall PTR, New Jersey, 2nd edition, 1993.

[Dgb] Digital Bangladesh Vision 2021. http://www.boi.gov.bd (last access October 2010).

[DM92] W. H. Delone and E. R. McLean. Information systems success: The quest for the dependent variable. *Information Systems Research*, 3(1):60–95, 1992.

[Dom] Domain software technologies ltd. http://www.domtech.co.uk/ (last access October 2010).

[DvLF93] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.

[DWP07] F. Deissenboeck, S. Wagner, and M. Pizka. An activity-based quality model for maintainability. In *International Conference on Software Maintenance(ICSM07)*, 2007.

[EH78] H. J. Einhorn and R. M. Hogarth. Confidence in judgment: Persistence of the illusion of validity. *Psychological Review*, 85 , no 5:395–416, 1978.

[EH04] J. E.Tomayko and O. Hazzan. *Human Aspects of Software Engineering (Electrical and Computer Engineering Series)*. Charles River Media, Inc., Rockland, MA, USA, 2004.

[EM97] K. Ewusi-Mensah. Critical issues in abandoned information systems development projects. *Communications of the ACM*, 40(9):74–80, 1997.

[ESSD07] S. Easterbrook, J. Singer, M.A. Storey, and D. Damian. *Selecting Empirical Methods for Software Engineering Research*. Springer, 2007.

[FCD02] A. Fuller, P. Croll, and L. Di. A new approach to teaching software risk management with case studies. In *CSEET '02: Proceedings of the 15th Conference on Software Engineering Education and Training*, page 215, Washington, DC, USA, 2002. IEEE Computer Society.

[FcY04] C.-Feng Fan and Y. chang Yu. BBN-based software project risk management. *The Journal of System and Software*, 73(2):193–203, 2004.

[FHK$^+$01] B. Freimut, S. Hartkopf, P. Kaiser, J. Kontio, and W. Kobitzsch. An industrial case study of implementing software risk management. *SIGSOFT Software Engineering Notes*, 26(5):277–287, 2001.

[Fir] D. Firesmith. Open process framework, opf metamodel. http://www.opfro.org/.

[Fir04] D. Firesmith. Creating a project-specific requirements engineering process. *Journal of Object Technology*, 3(5), 2004.

[Fir06] D. Firesmith. Requirements engineering tasks. *Journal of Object Technology*, 5(8), 2006.

[Fis67] M. Fisz. *Probability Theory and Mathematical Statistics*. John Wiley & Sons, New York, 1967.

[FK09] Daniel Mndez Fernndez and Marco Kuhrmann. Artefact-based Requirements Engineering and its Integration into a Process Framework. Forschungsbericht TUM-I0929, Technische Universität München, nov 2009.

[FLN$^+$98] N. Fenton, B. Littlewood, M. Neil, L. Strigini, A. Sutcliffe, and D. Wright. Assessing dependability of safety critical systems using diverse evidence. In *IEE Proceedings Software Engineering*, volume 145, pages 35–39, 1998.

[FM00] S.-Wei Foo and A. Muruganantham. Software risk assessment model. In *Proceedings of the IEEE International Conference on Management of Innovation and Technology (ICMIT 2000)*, 2000.

[FN08] N. Fenton and M. Neil. Visualising your risks, agenarisk. http://www.agena.co.uk/resources, 2008.

[FN09] N. Fenton and M. Neil. Measuring your risks. http://www.agena.co.uk/resources, 2009.

[FR07] R. France and B. Rumpe. Model-driven development of complex software: A research roadmap. In *Future of Software Engineering(FOSE07)*, pages 37–54, Washington, DC, USA, 2007. IEEE Computer Society.

[GBB$^+$06] E. Geisberger, M. Broy, B. Berenbach, J. Kazmeier, D. Paulish, and A. Rudorfer. Requirements engineering reference model (rem). Technical report, Technische Universität Müunchen, 2006.

[Gei95] J. D. Geier. The delphi survey methodology: An approach to determine training needs. In *Proceedings of the 8th SEI CSEE Conference on Software Engineering Education*, pages 389–402, London, UK, 1995. Springer-Verlag.

[GHKM00] L. Goossens, F. Harper, B. Kraan, and H. Meacutetivier. Expert judgement for a probabilistic accident consequence uncertainty analysis. *Radiation Protection and Dosimetry*, 90, no 3:295–303, 2000.

[Gil92] A. C. Gillies. *Software Quality: Theory and Management*. Chapman & Hall, Ltd., London, UK, UK, 1992.

[Gla98] Robert L. Glass. *Software Runaways: Monumental Software Disasters*. Prentice-Hall, 1998.

[Gla99] R. L. Glass. Evolving a new theory of project success. *Communications of ACM*, 42(11):17–19, 1999.

[Gli07] M. Glinz. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference(RE07)*, pages 21–26, 2007.

[GMS99] A. Gray, S. G. MacDonell, and M. J. Shepperd. Factors systematically associated with errors in subjective estimates of software development effort: The stability of expert judgment. In *METRICS '99: Proceedings of the 6th International Symposium on Software Metrics*, page 216, Washington, DC, USA, 1999. IEEE Computer Society.

[Gro95] Standish Group. Unfinished voyages. http://www.standishgroup.com/visitor/voyages.htm, 1995.

[HC99] B. Hughes and M. Cotterell. *Software project management*. The McGraw-Hill Companies, 2nd edition, 1999.

[Her99] D. S. Herrmann. *Software safety and reliability: techniques, approaches, and standards of key industrial sectors*. IEEE Computer Society, Washington, DC, USA, 1999.

[HGBJ05] S.H. Houmb, G. Georg, J. Bieman, and J. Jurjens. Cost-benefit tradeoff analysis using bbn for aspect-oriented risk-driven development. In *ICECCS '05: Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems*, pages 195–204, Washington, DC, USA, 2005. IEEE Computer Society.

[HIK$^+$10] S. H. Houmb, S. Islam, E. Knauss, J. Jürjens, and K. Schneider. Eliciting security requirements and tracing them to design: an integration of common criteria, heuristics, and umlsec. *Requirement Engineering Journal, special issue on security requirements*, 15(1):63–93, 2010.

[Hog87] R.M. Hogarth. *Judgement and Choice*. John Wiley and Sons, New York, USA, 1987.

[Hou07] S. H. Houmb. *Decision Support for Choice of Security Solution- The Aspect-Oriented Risk Driven Development (AORDD) Framework*. PhD thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, 2007.

[HWFP07] C. Hood, S. Wiedemann, S. Fichtinger, and U. Pautz. *Requirements Management: Interface Between Requirements Development and all other Engineering Processes*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[Iba] Ibacsbd limited. http://www.ibacsbd.com, Last access 2009.

[ID08] S. Islam and W. Dong. Human factors in software security risk management. In *LMSA '08: Proceedings of the first international workshop on Leadership and management in software architecture*, pages 13–16, New York, NY, USA, 2008. ACM.

[IEEa] IEEE Std 1219-1998: IEEE standard for software maintenance.

[IEEb] IEEE std 12207.0-1996, industry Implementation of International Standard ISO/IEC 12207:1995 standard for information technology -software life cycle processes.

[IEE98] The institute of electrical and inc. electronics engineers, ieee recommended practice for software requirements specifications,ieee std 830-1998, 1998.

[IEE06] IEEE16085: 2006 Systems and Software Engineering Life Cycle Processes Risk Management. The Institute of Electrical and Electronics Engineers, Inc., 2006.

[IH10] S. Islam and S. H. Houmb. Integrating risk management activities into requirements engineering. In *Proc. of the 4th IEEE Research International Conference on Research Challenges in Information Science(RCIS2010), Nice, France*, 2010.

[IH11] S. Islam and S. H. Houmb. Towards a framework for offshore outsource software development risk management model. *Journal of Software (JSW), Special Issue: Selected Papers of the IEEE International Conference on Computer and Information Technology (ICCIT 2009)*, 2011.

[IHMFJ09] S. Islam, S. H. Houmb, D. Mendez-Fernandez, and Md. M. A. Joarder. Offshore-outsourced software development risk management model. In *In Proc. of the 12th IEEE International Conference on Computer and Information Technology (ICCIT 2009), Dhaka , Bangladesh, DOI: 10.1109/ICCIT.2009.5407292*, pages 514–519, 2009.

[IJH09] S. Islam, M. A. Joarder, and S. H. Houmb. Goal and risk factors in offshore outsourced software development from vendor's viewpoint. In *ICGSE '09: Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering*, pages 347–352, Washington, DC, USA, 2009. IEEE Computer Society.

[IMJ10] S. Islam, H. Mouratidis, and J. Jürjen. A framework to support alignment of secure software engineering with legal regulations. *Journal of Software and Systems Modeling (SoSyM) Theme Section on Non-Functional System Properties in Domain-Specific Modeling Languages (NFPinDSML)*, 2010.

[IN08] C. L. Iacovou and R. Nakatsu. A risk profile of offshore-outsourced development projects. *Communication of the ACM*, 51(6):89–94, 2008.

[Isl09] S. Islam. Software development risk management model: a goal driven approach. In *ESEC/FSE Doctoral Symposium '09: Proceedings of the doctoral symposium for ESEC/FSE on Doctoral symposium*, pages 5–8, New York, NY, USA, 2009. ACM.

[ISOa] Information technology – process assessment – part 2: Performing an assessment, ISO/IEC 15504-2. http://www.iso.org/iso/.

[ISOb] International Standard: Quality Management Systems Requirements, iso 9001:2000. http://www.iso.org/iso.

[ISOc] ISO/IEC 27001:2005 Specification for Information Security Management. International Organization for Standardization (ISO) /International Electrotechnical Commission (IEC).

[Iso02] Risk management-vocabulary-guidelines for use in standards,iso/iec guide 73, 2002.

[Jen96] Finn. V. Jensen. *Introduction to Bayesian Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[JJ02] J. Järvinen and J. Jartti. Is your project ready for time-to-market focus? In *PROFES '02: Proceedings of the 4th International Conference on Product Focused Software Process Improvement*, pages 198–206, London, UK, 2002. Springer-Verlag.

[JK00] J. Jiang and G. Klein. Software development risks to project effectiveness. *The Journal of Systems and Software*, 52(1):3–10, 2000.

[JKD02] J. Jiang, G. Klein, and R. Discenza. Perception differences of software success: provider and user views of system metrics. *The Journal of Systems and Software*, 63(1):17–27, 2002.

[JKL98] G. Gerhard J. Kontio and D. Landes. Experiences in improving risk management processes using the concepts of the riskit method. *SIGSOFT Software Engineering Notes*, 23(6):163–174, 1998.

[Kar95] D. W. Karolak. *Software Engineering Risk Management*. IEEE Computer Society Press, 1995.

[KCLS98] M. Keil, P. E. Cule, K. Lyytinen, and R. C. Schmidt. A framework for identifying software project risks. *Communications of the ACM*, 41, issues 11(11):76–83, 1998.

[KNA09] S.U. Khan, M. Niazi, and R. Ahmad. Critical success factors for offshore software development outsourcing vendors: A systematic literature review. In *ICGSE '09: Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering*, pages 207–216, Washington, DC, USA, 2009. IEEE Computer Society.

[Kon01] J. Kontio. *Software Engineering Risk Management: A Method, Improvement Framework and Empirical Evaluation*. PhD thesis, Helsinki University of Technology, 2001.

[KPP$^+$02] B. A. Kitchenham, S. L. Pfleeger, M. L. Pickard, P. W. Jones, D. C. Hoaglin, K. EI Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8):721–734, 2002.

[KS04] Y. H. Kwak and J. Stoddard. Project risk management: lessons learned from software development environment. *Technovation*, 24(11):915 – 920, 2004.

[Lin99] K. R. Linberg. Software developer perceptions about software project failure: a case study. *The Journal of Systems and Software*, 49(2-3):177–192, 1999.

[Lit] Hugin Lite. Hugin expert the leading decision support tool. http://www.hugin.com/.

[LK95] P. Loucopoulos and V. Karakostas. *System Requirements Engineering*. McGraw-Hill, Inc., New York, NY, USA, 1995.

[LW00] D. Leffingwell and D. Widrig. *Managing software requirements: a unified approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[LWE01] B. Lawrence, K. Wiegers, and C. Ebert. The top risks of requirements engineering. *IEEE Software*, 18(6):62–63, 2001.

[LYM03] L. Liu, E. Yu, and J. Mylopoulos. Security and privacy requirements analysis within a social setting. In *RE '03: Proceedings of the 11th IEEE International Conference on Requirements Engineering*, page 151, Washington, DC, USA, 2003. IEEE Computer Society.

[McC96] S. McConnell. *Rapid Development , Taming wild software schedules*. Microsoft Press, 1996.

[McC98] S. McConnell. *Software project survival guide*. Microsoft Press, Redmond, WA, USA, 1998.

[MCCW93] M.Paulk, B. Curtis, M. Chrissis, and C. Webster. Capability maturity model for software, cmu/sei-93-tr-024. Technical report, Software Engineering Institute, Carnegie Mellon, 1993.

[McM04] John McManus. *Risk Management In Software Development Projects*. Elsevier Science Inc., 2004.

[MG06] H. Mouratidis and P. Giorgini. *Integrating Security and Software Engineering: Advances and Future Vision*. IGI Global, 2006.

[MG07] H. Mouratidis and P. Girogini. Secure tropos: A security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering,World Scientific Publishing Company*, 17(2):285309, 2007.

[Moy97] T. Moynihan. How experienced project managers assess risk. *IEEE Software*, 14(3):35–41, 1997.

[MRW77] J. McCall, P. K. Richards, and G. F. Walters. *Concepts and Definitions of Software Quality, Factors in Software Quality. Volume I.* NTIS, 1977.

[MS87] J. G. March and Z. Shapira. Managerial perspectives on risk and risk taking. *Management Science*, 33, No. 11:1404–1418, November 1987.

[Neu95] P. Neumann. Safeware: System safety and computers. *SIGSOFT Software Engineering Notes*, 20(5):90–91, 1995.

[NI09] R. T. Nakatsu and C.L. Iacovou. A comparative study of important risk factors involved in offshore and domestic outsourcing of software development projects: A two-panel delphi study. *Information and Management*, 46(1):57–68, 2009.

[NKM08] Jaana Nyfjord and Mira Kajko-Mattsson. Integrating risk management with software development: State of practice. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 (IMECS)*, 2008.

[Nuk99] J. & Forsell M. Nukari. Suomen ohjelmistoteollisuuden kasvun strategiat ja haasteet. Teknologian kehittmiskeskus, 1999. Teknologiakatsaus 67/99.

[OBD$^+$06] A. OHagan, C. E. Buck, A. Daneshkhah, J. R. Eiser, P. H. Garthwaite, D. J. Jenkinson, J. E. Oakley, and T. Rakow. *Uncertain Judgements: Eliciting Experts Probabilities. Wiley, 2006.* Wiley, 2006.

[OG09] E. E. Odzaly and P. Des Greer. Software risk management barriers: An empirical study. In *ESEM '09: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 418–421, Washington, DC, USA, 2009. IEEE Computer Society.

[Pea01] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2001.

[Pfl00] S. L. Pfleeger. Risky business: what have we yet to learn about risk management. *The Journal of Systems and Software*, 53(3):265–273, 2000.

[PK01] S. L. Pfleeger and B. A. Kitchenham. Principles of survey research: part 1: turning lemons into lemonade. *SIGSOFT Software Engineering Notes*, 26(6):16–18, 2001.

[PNJE06] R. Prikladnicki, A. Nicolas, L. Jorge, and R. Evaristo. A reference model for global software development: Findings from a case study. In *ICGSE '06: Proceedings of the IEEE international conference on Global Software Engineering*, pages 18–28, Washington, DC, USA, 2006. IEEE Computer Society.

[Pre96] R. Pressman. *Software Engineering: A practitioners Approaches*. McGraw-Hill, 1996.

[Pri09] R. Prikladnicki. Exploring propinquity in global software engineering. In *ICGSE '09: Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering*, pages 133–142, Washington, DC, USA, 2009. IEEE Computer Society.

[PV09] J. D. Procaccino and J. M. Verner. Software developers' views of end-users and project success. *Communication of the ACM*, 52, No 5(5):113–116, 2009.

[PVOD02] J. Drew Procaccino, June M. Verner, Scott P. Overmyer, and Marvin E. Darter. Case study: factors for early prediction of software development success. *Information and Software Technology*, 44(1):53 – 62, 2002.

[PVSG05] J. D. Procaccino, J. M. Verner, K. M. Shelfer, and D. Gefen. What do software practitioners really think about project success: an exploratory study. *The Journal of Systems and Software*, 78(2):194–203, 2005.

[Pyx] Pyxisnet limited. http://www.pyxisnet.com (last access January 2010).

[Ras] Rasis bd limited. http:/www./rasisbd.com(last access January 2010).

[RH09] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[RKC96] H. Raghav Rao, N. Kichan, and A. Chaudhury. Information systems outsourcing. *Communication of the ACM*, 39(7):27–28, 1996.

[RL00] J. Ropponen and K. Lyytinen. Components of software development risk: How to address them? a project manager survey. *IEEE Transactions on Software Engineering*, 26(2):98–112, 2000.

[Rop99] J. Ropponen. Risk assessment and management practices in software development. In *In: Willcocks, L.P., Lester, S. (Eds.), Beyond the IT Productivity Paradox;john Wiley & Sons, Chichester. pp. 247-266.*, 1999.

[Ros97] S. M. Ross. *Introduction to Probability Models*. Academic Press, 1997.

[Roy04] G. Roy. A risk management framework for software engineering practice. In *ASWEC '04: Proceedings of the 2004 Australian Software Engineering Conference*, page 60, Washington, DC, USA, 2004. IEEE Computer Society.

[RPJLNA06] R. Evaristo R. Prikladnicki and M. H. Yamaguti J. L. N. Audy. Risk management in distributed it projects: Integrating strategic, tactical, and operational levels. *International Journal of e-Collaboration*, 2:1–18, 2006.

[Saa96] T. Saarinen. An expanded instrument for evaluating information system success. *Information and Management*, 31(2):103–118, 1996.

[SB03] H. M. Sneed and P. Brössler. Critical success factors in software maintenance-a case study. In *ICSM '03: Proceedings of the International Conference on Software Maintenance*, page 190, Washington, DC, USA, 2003. IEEE Computer Society.

[Sch] B. Schätz. Model-based development of software systems: From models to tools. Habilitation, Technische Universität München, 2008.

[Sch97] Roy C. Schmidt. Managing delphi surveys using nonparametric statistical techniques. *Decision Sciences*, 28(3):763–774, 1997.

[Sch04] K. Schwalbe. *Information Technology Project Management*. Course Technology Press, Boston, MA, United States, 2004.

[Sch08] B. Schneier. The psychology of security. http://www.schneier.com/essay-155.pdf, January 2008.

[SDJ07] Dag I. K. Sjoberg, T. Dyba, and M. Jorgensen. The future of empirical methods in software engineering research. In *FOSE '07: 2007 Future of Software Engineering*, pages 358–378, Washington, DC, USA, 2007. IEEE Computer Society.

[SFI09] S.Wagner, D. Mendez Fernandez, S. Islam, and K. Lochmann. . A security requirements approach for web systems. In *Proc. Workshop Quality Assessment in Web (QAW 2009)*, 2009.

[Sha76] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press,Princeton, NJ, 1976.

[SJ94] F. Sisti and S. Joseph. Software risk evaluation method version 1.0. Technical report, SEI,Carnegie -Mellon University, 1994.

[SLKC01] R. Schmidt, K. Lyytinen, M. Keil, and P. Cule. Identifying software project risks: An international delphi study. *Journal of Management Information Systems*, 17(4):5–36, 2001.

[Smi89] M.J. Smithson. *Ignorance and Uncertainty: Emerging Paradigms*. Springer-Verlag, 1989.

[S.N09] S.N.Geethalakshmi. Non-technical components of software development and their influence on success and failure of software development. In *Proceedings of CONSEG-09: International Conference on Software Engineering*, 2009.

[Sne96] H. M. Sneed. Modelling the maintenance process at zurich life insurance. In *ICSM '96: Proceedings of the 1996 International Conference on Software Maintenance*, page 217, Washington, DC, USA, 1996. IEEE Computer Society.

[SNKT08] Z. Sheng, M. Nakano, S. Kubo, and H. Tsuji. *New Frontiers in Artificial Intelligence, Risk Bias Externalization for Offshore Software Outsourcing by Conjoint Analysis*. Springer Berlin / Heidelberg, 2008.

[Sou] Southtech limited. http://www.southtechlimited.com, last access 2009.

[Spe] Spectrum engineering consortium limited. http://www.spectrum-bd.com (last October 2010).

[SPHP02] B. Schätz, A. Pretschner, F. Huber, and J. Philipps. Model-based development of embedded systems. In *OOIS '02: Proceedings of the Workshops on Advances in Object-Oriented Information Systems*, pages 298–312, London, UK, 2002. Springer-Verlag.

[SS96] S. J. Stephen and D. A. Schkade. A psychological approach to decision support systems. *Management and Science*, 42(1):51–64, 1996.

[SS97] Ian Sommerville and Pete Sawyer. *Requirements Engineering - A Good Practice Guide*. John Wiley and Sons, Inc., 1997.

[TBK99] D.P. Truex, R. Baskerville, and H. Klein. Growing systems in emergent organizations. *Communications of ACM*, 42(8):117–123, 1999.

[Tha02] R. H. Thayer. *The Project Manager's Guide to Software Engineering's Best Practices*. IEEE Computer Society Press, Los Alamitos, CA, USA, 2002.

[TSY$^+$07] H. Tsuji, A. Sakurai, K. Yoshida, A. Tiwana, and A. Bush. Questionnaire-based risk assessment scheme for japanese offshore software outsourcing. In *Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD07)*, page 114127. LNCS 4716, 2007.

[TTL00] D. Tapscott, D. Ticoll, and A. Lowy. *Digital Capital: Harnessing the Power of Business Webs*. Harvard Business School Press, Boston, 2000.

[vL09] A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[vLL00] A. van Lamsweerde and E. Letier. Handling obstacles in goal-oriented requirements engineering. *IEEE Transactions on Software Engineering*, 26:978–1005, 2000.

[VM04] D. Verdon and G. McGraw. Risk analysis in software design. *IEEE Security and Privacy*, 2(4):79–84, 2004.

[Vos00] D. Vose. *Risk Analysis: A Quantitative Guide*. John Wiley & Sons Ltd, 2000.

[WA95] C. Wohlin and M. Ahlgren. Soft factors and their impact on time to market. *Software Quality Journal*, 4:189–205, 1995.

[Wag09] S. Wagner. A bayesian network approach to assess and predict software quality using activity-based quality models. In *PROMISE '09: Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, pages 1–9, New York, NY, USA, 2009. ACM.

[WdOA05] K.P. B. Webster, K. de Oliveira, and N. Anquetil. A risk taxonomy proposal for software maintenance. In *ICSM '05: Proceedings of the 21st IEEE International Conference on Software Maintenance*, pages 453–461, Washington, DC, USA, 2005. IEEE Computer Society.

[WDW08] S. Wagner, F. Deissenboeck, and S. Winter. Managing quality requirements using activity-based quality models. In *WoSQ '08: Proceedings of the 6th international workshop on Software quality*, pages 29–34, New York, NY, USA, 2008. ACM.

[Wie03] Karl Eugene Wiegers. *Software Requirements*. Microsoft Press, Redmond, WA, USA, 2003.

[WK04] L. Wallace and M. Keil. Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4):68–73, 2004.

[WK07] W. Wu and T. Kelly. Combining bayesian belief networks and the goal structuring notation to support architectural reasoning about safety. In *Computer Safety, Reliability, and Security, Lecture Notes in Computer Science*, 2007.

[WKR04] L. Wallace, M. Keil, and A. Rai. Understanding software project risk: a cluster analysis. *Information and Management*, 42(1):115–125, 2004.

[WvMHR00] C. Wohlin, A. von Mayrhauser, M. Hst, and B. Regnell. Subjective evaluation as a tool for learning from software project success. *Information and Software Technology*, 42(14):983 – 992, 2000.

[Yin02] R. K. Yin. *Case Study Research: Design and Methods, Third Edition, Applied Social Research Methods Series*, volume 5. Sage Publications, Inc, 3rd edition, December 2002.

[YLL01] E. S.K. Yu, L. Liu, and Y. Li. Modelling strategic actor relationships to support intellectual property management. In *ER '01: Proceedings of the 20th International Conference on Conceptual Modeling*, pages 164–178, London, UK, 2001. Springer-Verlag.

[Yu96] E. Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, Toronto, Ont., Canada, Canada, 1996.

[Yu97] E. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *RE '97: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, page 226, Washington, DC, USA, 1997. IEEE Computer Society.

[Zad99] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 100:9–34, 1999.