Integrating Risk Management Activities into Requirements Engineering

Shareeful Islam
Institut für Informatik
Technische Universität München, Germany
islam@in.tum.de

Siv Hilde Houmb
Arena for Service Innovation
Telenor GBD&R, Norway
siv-hilde.houmb@telenor.com

Abstract— **Software projects are often faced with unanticipated problems caused by e.g. changes in the development environment resulting in delays or threatening the ability of the project to succeed. Managing these uncertainties is a challenging task at all phases of the development, but nevertheless crucial in controlling schedule and costs. Therefore software development risks need to be controlled as early as possible. As software development risks are not merely of technical nature it is equally important to tackle non-technical risks. The paper presents a goal-driven software development risk management model (GSRM) that takes a holistic view on development, taking both technical and non-technical development components into consideration. The focus of the paper is on how to integrate GSRM and particularly the holistic risk perspective into requirements engineering. GSRM effectively identifies and makes explicit the critical project goals (for arriving at a successful project) and the risk factors that may obstruct these goals. GSRM also helps in planning how to employ control actions for mitigating risks and by that increase the ability to meet project goals. The integrated requirements engineering risk management model has been applied to an on-going development project in a low-cost development environment (Bangladesh). The result showed it to be relatively trivial to integrate the model into requirements engineering activities and that the model did indeed contribute to the overall project success.**

Paper Category- Technical Solution

Keywords - software development riskt; risk management; goal-driven modelling; risk modelling; requirement engineering.

I. INTRODUCTION

Software development projects have to deal with both generic and project specific risks and particularly those related to delay, stress of entering into a new market, miscommunication among project stakeholders, missing business features, erroneous requirements, and many more. Risk management in software development is challenging, but effectively contributes to control these problems before they occur and certainly improves the overall project outcome. However, the problem though is not that developers and project managers are not aware of the importance of risk management and its positive contribution to project outcomes, but that risk management is not effectively applied in practice [19, 20]. A study showed that, 75% of surveyed project managers did not follow any detailed risk management approach [20]. The cause of most project failure has little to do with technical issues despite of the common tendency among project managers to focus more on these [16]. Failed projects just as often suffer from the poor management of people-related problems [6, 16]. McManus [17] identified that 65% of the project failures are accounted by management issues and 35% by technical issues. Several software risk management approaches emphasize the importance of performing risk management activities as early as possible [1, 13, 15, 18, 20]. However, there is still a lack of comprehensive detailed guidelines describing how to integrate risk management activities explicitly at the early development stage. If risk management activities are merely employed from the design phase on, the result may end up in expensive revision to the design or major rework of the elicited requirements and related artefacts. This may also pose additionally problems later on depending on the competence and ability of developers to tackle and quickly respond to late discovered risks or mistakes in the requirements, inconsistent design, and may also end with passive customer/user involvement.

This paper contributes to integrate Goal-driven Software Development Risk management Model (GSRM) [8, 9] for managing software development risk as part of Requirements Engineering (RE). The model considers goals relating to project success beyond schedule, budget, and quality and recognises the importance of motivating project stakeholders in particular customer/user to take active part during the development. The model focuses on the non-technical components such as project execution constraints, stakeholders, customers/user and project participants' communication, and usage environment, along with the technical components such as development process and tools even before starting with the requirements elicitation. By doing so, we believe GSRM not only contributes to reducing the error rate in the elicited requirements but also to control issues relating to non-technical development factors. This contributes for an effective development process moving steadily towards a successful project. The integration of risk management, here GSRM, into RE follows two perspectives; i.e., artefact and process oriented view. This allows us to specify the dependencies between requirement and risk artefacts along with the underlying activities and tasks. We employed the model in an on-going offshore software development project in Bangladesh as a case

study and to demonstrate the effect of integrating GSRM into requirements engineering activities and by that reducing requirements errors and contributing to increased project success. The case study also evaluated the feasibility of integrating GSRM into RE.

The structure of the paper is as following. Sect. II outlines the early software development components as foundation concept for the Goal-driven Risk Management Model. The framework of the model is introduced in Sect. III. Sect. IV described the fundaments of integrating GSRM into RE. The integrated model is demonstrated at the hand of a case study in Sect. V. Sect. VI gives overview of related works and Sect. VII concludes the paper and points to future work.

II. EARLY SOFTWARE DEVELOPMENT COMPONENTS

To develop a goal based risk management model it is important to understand the basic elements of software development, what it takes to succeed with software development and how to specify project goals and identify and address risk. Therefore, we have investigated the early software development components and project success factors from existing literature and from these specified a set of general project goals and identified a set of often experienced risks to these. Our initial focus was on issues relating to the initial part of a development project, including RE activities, artefacts, and then further to the rest of the development phase. The main task of RE is to produce a number of artefacts towards a comprehensive requirements specification document that aims to describe the problem space of the future system-to-be or system-to-be-next. Integration of GSRM at RE stage facilitates to manage any change in particular relating to cost and schedule rather easily. For example, a study found that cost relates to fixing errors during the testing phase is twenty times more than the cost of fixing these in the requirements phase [2]. Moreover requirements errors are the most expensive software errors that persist throughout the system life cycle [15]. There are several reasons for requirement problems, such as developers failing to address requirements because they consider requirement specification as being the responsibility of the customers. However, customers rarely have a clear conception of their problem domain about the system-as-is and are often not able to state their requirements explicitly, but expect the end-product to meet all their needs and supports the business demands. Developers when involved in RE may not have adequate project specific domain knowledge. There may also be infectivity in the activities used to elicit, analyze, and validate the user and system requirements. Practitioners of the development team commonly focus more on solution oriented view of the system-to-be rather than detailed analyses of the existing problem space. Project may not support adequate schedule and budget for requirement engineering. These are the problems during RE that pose major risk to successful development. Therefore, if factors relating to these issues are addressed up-front, even before the actual elicitation of requirements, it can effectively contribute not only to reduce requirement errors but also to increase the ability for the project success.

GSRM provides a greater understanding of the early technical and non-technical software development components and how these relates to RE from the perspective of project success. However the perception of success and successful project differ significantly among the various stakeholders including customer/user, software practitioner, project manager, and senior/executive management. The reasons are that each of these groups has different backgrounds, responsibilities, expectations, and understanding to evaluate project success. Generally accepted industry standard organizational/managerial definition of projects success is: having met agreed upon business objectives, been completed on time and within budget, meets all customer/user requirements, has effective project management and achieve user satisfaction [6, 14, 18, 20]. The user satisfaction is the single most widely cited measure of the system success [10]. On the other hand, practitioners tend to focus more on the micro-level project view (details of design, cool coding, etc.) compared to project management such that ensuring that requirements are technically realistic, realistic estimation of schedule and effort, effective leaders, diverse and synergistic development team, employee motivation, and adequate development facilities [14, 16, 18]. These are important success factors in respect to the development process, associate management, project constraints, and overall product. Furthermore, these factors combine both technical and non-technical aspects of the development. .

According to Boehm [2, 3] and McConnell [16] effective and efficient software development and ultimate project success can be framed in terms of people, process, product and technology. Procaccino et al. [18] further categorise seven factors such as management, customers and users, requirements, estimations and scheduling, the project manager, the software development process, and development personnel that contribute to the success and failure of the software systems. Several other researches also emphasize development environment and project management related issues as critical components that directly influence project success [6, 10]. Based on our investigation from the existing literature, we categorise software development components into five dimensions(e.g. as shown in Fig. 1). These are: project execution constraints, development process, product, human, and finally environment (internal & external). These components are all based on a set of elements that are essential for the component. The elements may further be characterised by terms of single or multiple factors. Thus elements and factors collectively represent the characteristics, artefacts, methods, and activities required for the development components. Generally, the elements are intertwined, interdependent, and contribute combindely to attain one or more development goals that influence for the project success. The component-element-factor hierarchy focuses on both technical (i.e. hardware and software) and non-technical (i.e.

human factors, project management, and environment) aspects of software development. However, managing non-technical issues is rather difficult and challenging compared to the technical ones. Unfortunately, project managers tend to neglect these factors as it requires certain time, experience, and quality to attain these factors at a reasonable level. However, experience has shown that these factors indeed play a critical role for the success or failure of software development [6, 16, 17]. A brief overview of the components is given below:

Project execution constraints: This component considers relevant elements for the project execution such as project planning and control including factors like budget, schedule, roles and project management, project scope including factors like success criteria, boundary, and contract and technical issues including tools, hardware and software and complexity. Therefore the component consists of three elements which all are further categorized into factors.

Processes: The activities, tasks, and methods for the development and risk management process, their usage during the development, and tool support are considered under the process.

Product: This component is concerned with the early artefacts of the business and requirement specification such as business goals, business process, business domain, system vision, user, system and architectural requirements. Furthermore, it also focuses on the requirement faults, documentation, priority, traceability, and product quality factors as elements and factors for the component.

Human: This component mainly deals with the non-technical issues relating to the practitioner, customer/user, and management that directly or indirectly influence the development. For instance, practitioner's knowledge, skill, motivation, customer/user's involvement, team overall performance, coordination, management supports are considered by this component.

Environment (internal & external): This component deals with the development project environment, including in-house sourcing or outsourced, development facilities, corporate environment are main consideration by this component.

GSRM requires a detailed elaboration of these components so that expectations from these components can be mapped with the issues relating to the project success. GSRM considers them as goals of the development component and this further eases to identify risk factors that obstruct these goals. Therefore component-element-factor hierarchy allows us to identify and category the goals and risk factors during the development. For instance, *requirements specification* is an element under the *product component* and *error free requirement* is an important expectation from the element for any software project. On other hand, *requirement errors* certainly obstruct this goal to attain. This hierarchy supports to focus on holistic view of the development. For instance, elements and factors of the human and environment component focus more on the non-technical issues, but product and process components, on the other hand, focus more on the technical issues, but GSRM analyses them combindely for the software development risk management.

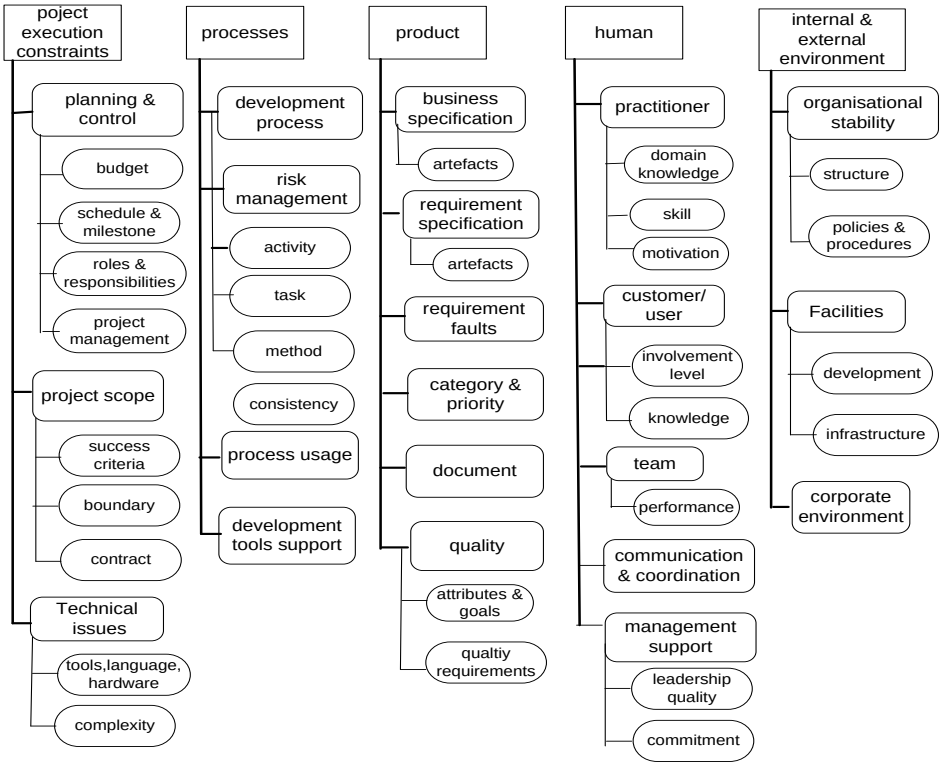

Figure 1. Early development component-element-factor hierarcy

III. GOAL-DRIVEN SOFTWARE DEVELOPMENT RISK MANAGEMENT MODEL (GSRM)

We propose GSRM to follow the existing goals modelling techniques to accommodate the risk management activities. Goal provides anchor for risk analysis and facilitates to model and trace the risk factors that obstruct the goals and countermeasure that satisfy the goals [15]. Goal modelling language such as KAOS, i* , and Tropos has long been recognized in the RE community as useful to elicit, analyze, negotiate, document, and modify requirements. GSRM extends KAOS to support risk management activities during RE. KAOS defines obstacle as a construct that can be used to identify undesirable behaviour against the strategic interest of a stakeholder [15]. GSRM adopts this construct and defines software risk factors and their consequence as obstacles that contribute negatively to fulfilling the specific development goals [8]. These risks must be analysed and controlled and GSRM does this by assigning suitable treatment actions. Thus GSRM adopts goal and obstacle concept from the KAOS and further extends these with risk assessment and treatment for modelling and managing software development risk. This facilitates the reasoning and tracing of treatment actions and their ability to mitigate risks, and hence, to fulfil goals. This is done using the four layer modelling structure of GSRM: (i) Goal layer, (ii) Risk-obstacle layer, (iii) Assessment layer, and (iv) Treatment layer.

Goal layer

GSRM starts with identifying, elaborating, and modelling the goals from the components-element-factor hierarchy. These goals are the objective, constraints, and expectations from the development components. The initial identified goals can be higher level representations of the abstract expectation from the components. Therefore elicited goals is refined using AND or OR refinements into sub-goals to provide a concrete meaning for their satisfaction. Thus goal refinement supports different levels of abstraction ranging from higher level coarse grained to lower level finer-grained sub-goals.

Generally sub-goals contribute to the parent goals by including *contribution link* from the sub-goal to the related parent goal. This goal refinement together makes up the *goal model*. Most of the goals in software development are soft in type as they specify several alternatives to satisfy the main objective. However some times the goals are also behavioural (i.e. known as hard) to specify certain clear cut objective of any property. For instance, every project should *maintain [EstimatedBudget ThroughoutDevelopment]*, which represents a clear cut goal, on the other hand, *improve [Customer/userParticipation]* during development cannot be specified in strict sense. This is because customer/ user may not have adequate time to actively participate in the development but expect the project to be finished within budget, on time and to meet their implicit expectations. Goals are represented in natural language through a precise meaning describing the purpose of the goal. GSRM also follows informal temporal pattern as stated in KAOS [12] to represent the goal. However whatever syntax is used for the goal representation, i.e., temporal pattern or natural language, it must precisely state its meaning in an explicit manner.

Risk-obstacle layer

Risk obstacles are the causes that reduce the ability to satisfy a single or multiple goals. This layer is used to identify the risk factors that influence the undesirable events that may occurred during a development project. To ease the risk identification in the early requirements phase, GSRM provides a set of general risk factor structured according to the goal categories along the components-element-factor hierarchy. For instance, if a goal is to *improve overall team* performance then this layer focuses on the factors that could deteriorate the overall team performance such as *frequent conflicts among the team members*, *negative team attitude*, *incompetence staff*, and so on. We provide *obstruction link* from the risk factor to the goal and this allows constructing the *goal-risk model*.

The same risk factor can obstruct more than one goals and this is important to capture this obstacle, as it is crucial information when later considering treatment options. Risk factors that cross-cut several goals are in general more effective to treat, as the effect of a treatment in such cases often propagates to goals that are not directly linked to the particular risk factor. In GSRM, we follow a set of questionnaires (i.e. such as those in Karolak's SERIM method [5]) based on the state of early development components to identify the risk obstacles. The Questionnaires consist of 82 close questions and arranged sequentially based on the component-element-factor hierarchy. Overview of the questions is given in section V. We also recommend using brainstorming session with key project members to review and categorize the risk factors from the answer of the questionnaires.

Assessment layer

The main role of the assessment layer is to provide more insight into each individual risk factor. This includes identifying any resulting event of the risk factors. E.g. risk event. Each risk event is characterized using the two properties: (a) likelihood and (b) impact. *Likelihood* specifies the rate of occurrence of a risk event and is modelled as a property of the risk event itself. *Impact* is a measure over the negative consequence of a risk event to the goals. Therefore this layer quantifies the individual risk level through risk event likelihood and impact. GSRM only allows risk factors that directly obstruct goal or that in some way cause problems in executing development activities. Thus, a risk event is defined as an undesirable circumstance of the early development environment. What is important to take into consideration when working on the assessment layer is that the same risk factor may leads to more than one risk event and that the same risk event can obstructs more than one goal. Such representations allow capturing situations where an event is influenced by more than one risk factor and where both factors and event combindely impact negatively to single or multiple goal. The value of likelihood and impact estimates the risk

level for specific goals. We use a qualitative scale (i.e. high, medium and low) to estimate the risk level, likelihood, and impact.

This layer models the risk events by following the casual relationship from the risk factors to the related risk events. Thus risk factors as causes are refined to risk event and further mapped with the consequences as goal negation. We follow Bayesian Belief Network [11] to construct a *casual relationships model* from the risk factor to the risk event. Furthermore, this layer also enhances the goal-model by including contribution link from the risk factor to the risk event and obstruction link from risk event to the related single or multiple goals. This allows tracing the obstacles to the goals. The risk assessment layer finally prioritises the risk based on the risk level derived from the likelihood and impact values.

Treatment layer

The fourth and final layer of the GSRM is the treatment layer which models the possible control actions and chooses the most suitable ones to mitigate the risks. Once the goals, risk factors, and events are identified and analysed by the goal, risk obstacle and the assessment layers, then it is crucial to identify, plan and then quickly implement cost effective countermeasures. Thus the aim of this layer is to gain control of the software development risks as early as possible and preferable in the earliest stages of RE by assigning appropriate countermeasures. Risk treatment further requires monitoring the status of individual risks throughout the development. Thus, it evaluates the effectiveness of the implemented control action and identifies any new risks during the course of the development. The initial consideration should be the risk factors that influence several risk events as well as obstruct several goals. E.g. high prioritised risk factors and associated events. Note that, there can always be alternative countermeasures to the obstacles, but treatment layer should select the most potential ones for the risk mitigation. Every treatment action requires evaluating based on several criteria such as schedule, cost, resource availability, and goals for its implementation. Furthermore, project context is also important in identify and select the suitable countermeasures.

This layer includes three different links; contribution link from the control action to the goals, obstruction link from control action to the risk event and finally responsibility link from the control action to the agent that is responsible to prevent, reduce or avoid the risk. This allows tracing and reasoning the treatment action to the goal satisfaction and obstacle obstruction.

Fig. 2 shows the modelling framework of GSRM. Note that GSRM uses the same notations for goals (parallelogram) and obstacles (reverse parallelogram) as the KAOS model. On top is the goal layer which refined parent goal through AND and OR refinement depending on the goal context. The two middle layers collectively represent the software development risks as obstacle which directly obstructs the goals. Therefore top three layers combindely produces the goal-risk model. The bottom level is the treatment layer which initially contains goals as prevent, reduce and avoid risk and assigns responsibilities to agents i.e. resource such as project participant and specific tool, that contributes to control the risk to satisfy the goal. Therefore, treatment layer includes contribution, obstruction, and responsibility link to the top three layers.

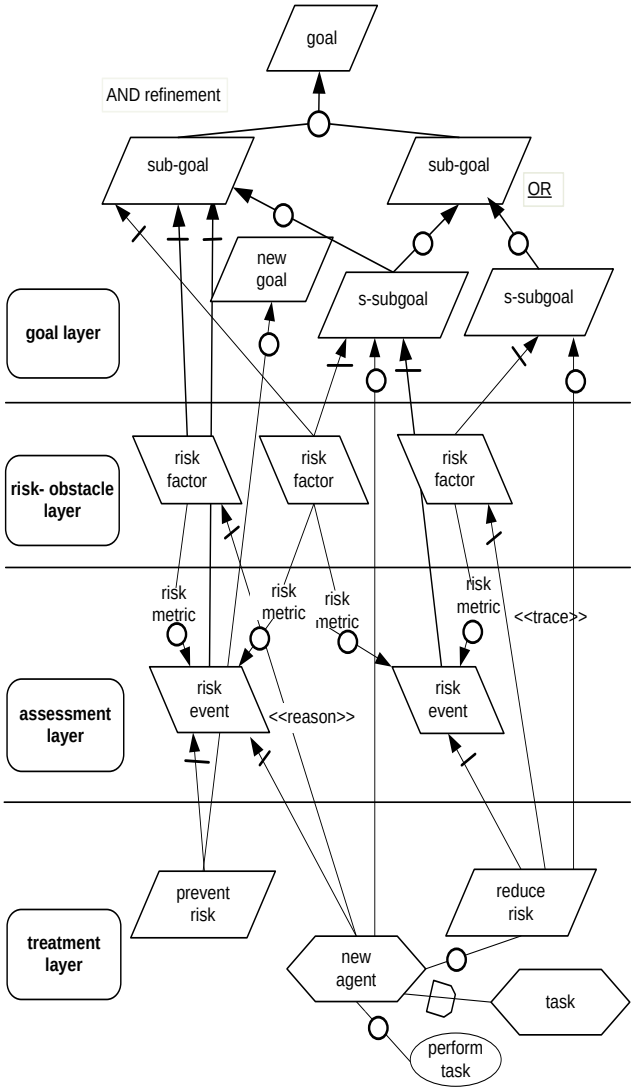

Figure 2. Overview of GSRM

IV. USING GSRM TO SUPPORT REQUIREMENTS ENGINEERING

As stated, we follow artefact and process orientation view to understand the background foundation regarding the integration of GSRM into RE time. A short overview of these two principals is given below.

Artefact oriented view

Artefact oriented requirement engineering is a systematic methodology that describes the problem space of the system-as-is as comprehensively as possible towards complete, consistent, and rigid requirement specification document. The artefact orientation combines both structure and content of the artefacts and incorporates techniques and notions for producing the consistent and complete result. Artefact oriented requirement engineering in particular for the business information domain mainly covers two main artefacts types, i.e., *business specification* and *requirements specification*, considering system-to-be or system-to-be-next [5]. The business specification contains several content items such as business vision, business domains, business goals and restrictions, business roles and capability and requirement specification with system vision and user, organisational and integrational requirements [5]. Artefacts rely on concepts to describe the content of the artefact and syntax to represent the concept through textually or graphically representation. GSRM also focuses on the artefact oriented view as work product by the underlying activities and tasks. The main artefact type of GSRM is the *risk specification* that consists of risk management plan, goal detailed, risk detailed and risk status report. These risk management concepts mainly represent through highly structured text by following the natural language. On the other hand, modelling concepts about software development risk such as goal-risk model, causal relationship model are generally represented graphically. The requirement specification artefacts provide limited visualisation support by following use case, activity, or sequence diagram when representing the user requirements or scenario.

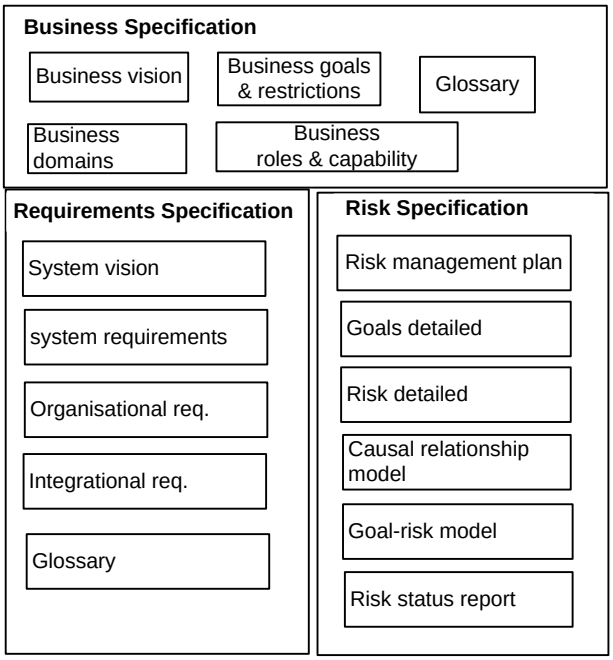

Figure 3: Overview of the artefact types

Both requirement and risk artefacts are interdependent upon each other. Fig. 3 shows an overview of the requirement and risk artefacts. *Goals* are one of the main initial elementary artefacts that support to create business, requirement, and risk specification. Several goals such as business goals, stakeholder expectations, constraints, and problems of the system-to-be are identified and reviewed to elicit user and system requirements. Risks are identified by analysing the negation of the identified goal in particular those relating to the development component-element-factor. The more the goals refine the easier it is to assess and manage the software development risks. Risk controls actions also introduce new goals in terms of reduction, prevention, and avoidance of risk from the development environment. Goals support tracing and rational from higher stakeholder expectation, business needs, and system objective to the refined system requirements and further to the control action for the goal satisfaction. Requirement artefacts are among one of the elementary inputs for risk identification. On the one hand, quality of requirements highly influences to attain goals relating to schedule, budget, quality, and error free requirements. In fact, *reduce project risk* is a critical requirement for any project situation. On the other hand, complete requirement specification document is highly desirable for any software development project. Requirement errors are one of the most expensive software development risks [5, 15]. Therefore risk control actions such as include competence practitioner to the development team, increase customer/user active participation, adequate budget for requirements engineering, adequate domain analysis, and so on certainly contribute to attain complete requirements specification document.

Process oriented view

The *process oriented view* deals with the underlying activities of both the requirements engineering and GSRM. Requirement engineering is comprised of elicit, analyze, validate, and management activities separated into several fine-grained tasks and sub-tasks. In GSRM, we consider several activities for the software development risk management, such as plan risk management planning, identify and model goals and obstacle, and assess and treat risks. Requirement elicitation techniques commonly rely on background study of specific type of artefacts 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 system. Furthermore, the elicitation also focuses on stakeholder-driven processes such as structured and unstructured interview and workshop-like activities. Risk management planning, in particular specifying the risk context, development component goals, and identify risk obstacle, also focus on the preliminary analysis of the system-as-is, running project information, project domain analysis, and the requirement artefacts. Taxonomy based questionnaires and brainstorming session with stakeholder are also very effective techniques for risk identification. This means that the techniques used as well as the input artefacts require for goal, requirement and risk identification are similar. Furthermore, risk monitoring are similar to requirement validation and management with being a continuous activity

throughout the development life cycle. Thus both requirements engineering and risk management are iterative processes.

GSRM focuses a holistic view of the overall development environment. Activities and tasks under the development process require certain responsibilities deemed *roles* where roles are the active entity(ies) that performs the activities. *Customer/user representative* in particular members of user groups play important role to elicit both requirements and risks. *Business analyst* with particular domain knowledge relating to certain customer domain such as financial sector or insurance is responsible for creating business specification. *Requirement engineer* is the key responsible person that creates and manages the requirement specification by aligning the business needs with the needs towards the software-to-be. *Risk manager* is mainly responsible for the risk assessment and management activities. But in real project situation, in particular for small or medium size project, there may not have any risk manager due to budget constraints. Therefore, project manager concerning the overall project execution also performs the role of a risk manager. This means that the project manager needs adequate experience with both project execution and risk management, in cases where he/she is responsible for risk management.

We propose to start with the goal and risk identification activities of the GSRM in parallel to requirement elicitation activities. This is because it is beneficial to carry out these activities as part of preparing artefacts such as business vision, business processes and system vision for customer approval. Therefore, goals and risks relating to the business needs and project scope can be easily and effectively identified at this stage. Although, note that if required, certain goals and risks from the elements and factors of the project execution constraints, human, and environment are analysed before the elicitation of the business specification and system vision, i.e. prior to user or system requirements elicitation. For instance, goal and risk factors relating to project schedule and budget, staffing, tools support, customer/user involvement, project participants knowledge, management commitment, organisational stability, and development facilitates. In particular, focusing on these aspects early on allows us to capture non-technical project risks up-front, even before any requirements have been identified. To effectively tackle risks at an early stage and to reduce errors or wrong requirements; it is important to align the risk management plan with the project scope and system vision. As a minimum, the risk management plan shall define the scope, schedule, and pre-conditions of the risk identification, analysis and evaluation activities and align these with the requirement engineering activities, such as requirements elicitation. The framework is also flexible and can be tailored to the particular project such that it fits with the project scope, budget and development timeframe.

V. DEMONSTRATION OF GSRM FOR REQUIREMENTS ENGINEERING

We employed GSRM as part of requirements engineering to an on-going offshore software development project in Bangladesh. A short overview of the case study and its results is presented in this section.

A. Context

The company was a software development house in Bangladesh established in 1998. From 2003, the company expanded their business strategy to include offshore customers and since 2007 they have completed several 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, one of the co-authors, a former part-time employee of the company, obtained consent from the managing director to perform the risk management activities into a running project. Four Master in Information Technology students of a university mainly took part in the case study. They are the project students of the co-author and have obtained adequate knowledge about software risk management through courses and about GSRM through tutorial. Moreover, two of them have gained experienced by working in three different software projects..

The development team was on a tight schedule and therefore not interested in following a detailed tutorial on software development risk management and GSRM. Our team therefore decided to give a high-level overview of GSRM and rather take active part in the risk management activities themselves. The situation is similar to action research, but required an even tighter communication with the developers to be successful. The project context was development of application software with a set of common features such as data tracking, searching and filtering, and reporting linked to external components offered by other systems at the customer site. The development team consisted of 9 members, including project manager, requirements engineering, software architects, developers, and testers with approximate duration of ten months. Due to confidentiality restrictions, we cannot provide more information about the project. 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.

B. Case study objective

The main objective from our side was to analyse the effectiveness of the 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 in requirements engineering time using GSRM. For evaluation purposes, we identified a set of hypothesis to evaluate the observed results. These are:

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

C. Instrument

Our team initially attempted to identify the goals and risk factors from the development components. To support them in this activity, the developers reviewed documents like information about the project and development team, project business context, and so on. 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 use of GSRM. The evaluation was performed using a mix of structured interviews, brainstorming sessions, and an offline analyze of the initial artefacts. The data collection was done in a two-steps manner. First step consisted of two different parts: (i) interview with the project team members using our interview template of 82 close questions, and (ii) brainstorming sessions was conducted with the project manager and requirement engineer. The interview results were used as input to the brainstorming sessions with the purpose to identify project goals and risk factors. The brainstorming sessions was also used to plan for risk control actions and their implementation. The final step of the evaluation consisted of 25 open questions asked to the interview participants. The goal of this step was to obtain feedback on the integration of risk management in requirement engineering and on GSRM.

D. GSRM activity and tasks into the running project

Goal identification and elaboration

The project participants identified an initial set of goals linked to particular business goals and the user expectations as part of the requirements engineering activities. We executed an offline review of the initial project documents to elaborate the goals based on project constraints, process, product, human and the environment. We completed the goal identification and modelling together with the project managers, requirement engineers and one customer representative via a series of conference calls. Note, however, that there was only one customer representative available for the GSRM activities.

Risk obstacle identification

An interview template with 82 close questions was used to identify the initial raw risk-obstacles from the project that obstruct the goals. A brainstorming session was also conducted together with the project manager and requirement engineers to review the raw risk factors and cluster them into groups according to components and elements. At this stage goals and risk factors are modelled and their detailed are documented. The interview template focused on the issues that obstruct the project goals in particular relating to budget, schedule, requirements, human factor and so on based on the development components. A short overview of the close questions is given below:

Project constraints (Budget, project scope)

[Q] Are all distinct milestones including estimated duration realistically identified & agreed with the customer?
[ ] Not at all [ ] Partially but not sufficient [ ] Distinct agreed milestone for each development phase
[Q] Up to now how much is the variation of the estimated schedule and cost compare to actual one?
[ ] high [ ] Medium [ ] Not at all
[Q-15]Is the project success criteria clearly defined?
[ ] Partial [ ] More than partial [ ] Full

Process (development activity)

[Q] Does the development activity adequate for every development phase?
[ ] Not adequate [ ] Partially adequate &documented [ ] Adequate & documented
[Q] Are all project members aware & trained with the development methodology?
[ ] Some are trained with some portion [ ] All trained with some portion [ ] All trained with all portion

Product (Requirements)

[Q] Are the requirements provided different ambiguous interpretations or lack of support for rational?
[ ] Highly [ ] Partially [ ] Rarely
[Q] Are the requirements categorised and prioritised?
[ ] Less than some [ ] Some [ ] Almost all
[Q] Do you follow any standard (template, notations, and checklist) for producing the requirement specification?
[ ] No [ ] Partially [ ] Yes

Human (competence practitioner)

[Q] What is the overall relevant domain knowledge of the development team?
[ ] Not much [ ] Less than adequate [ ] Adequate
[Q] How much capable is the project manager?
[ ] Unreliable [ ] Reliable [ ] Much reliable
[Q] What is the level of involvement of customer / user up to now?
[ ] Passively [ ] Occasionally [ ] Actively

Environment (internal)

[Q] Are there adequate infrastructure facility (e.g. power, space, internet, telephone) exists relating to communicate with customer / client or other distributed development site?
[ ] Not at all [ ] Partially [ ] Adequate
[Q] Is there any legal disputes considering data privacy, intellectual property rights of product & development artefacts with customer?
[ ] Yes [ ] Partially [ ] No

Risk assessment and treatment

Risks level is estimated by identifying the likelihood of risk event occurrence and impact of the occurrence towards the goal negation. The risks are prioritised 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 risk. Note that as GSRM focuses on effective use of time and resources the project manager was more concerned to *prevent* (if possible) or *reduce* the risk.

Therefore, our team focused more on the control actions that can prevent or reduce the risks. The selection of appropriate control action for the prioritised risks also depends upon the agent who is responsible for implementing the action. For instance the agent may be a practitioner that is responsible to countermeasure the risk. This means that the project manager role is also important when selecting the suitable risk control action. At this stage, our team documented details on the risk and the state of the risk status reports and the project manager was assigned the responsibility to monitor the risk throughout the development despite of the tight schedule pressure.

Feedback about the effectiveness of GSRM

Regarding the effectiveness of GSRM in requirements engineering, our team used 25 open-ended questions to structurally collect comments from the project practitioner. It was mainly the project manager, requirement engineering and one developer that participated in this last feedback-loop. The questions also help them to form their opinion about GSRM as goal-driven risk management approach in general and its contribution to requirements engineering in particular. A short overview of the close questions is given below:

[Q]What are the generic advantages/limitations of performing risk management into RE?
[Q] Do you think risk management at RE significantly contribute to reduce error from requirements?
[Q] Are there any dependencies between requirements and risk artefacts?
[Q] Is there any conflict situation arise between risk management and RE activities while performing the tasks under the activities within RE time?
[Q] Is software development risk management based on goal-driven a useful technique for risk management?
[Q] What are the main reasons to informally follow the risk management activities at your software development projects?
[Q]For each task under the GSRM, what are the advantages/problems from your opinion?
[Q] For each artefact of GSRM, what type of problem can arise in terms of its creation and maintenance at development in particular within RE?

E. Results

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

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 complete requirement specification. GSRM only consumed 14% of these efforts.

Goal and risk

There were several goals identified and agreed with the project manager and other practitioner of relevance to project success. Some of the goals are outline in Table I. These goals are important and desirable for any software development project.

TABLE I. LIST OF THE IDENTIFIED HIGH LEVEL GOALS

Project constraints
Improve[RealisticBudgetEstimation]
Maintain[EstimatedBudgetThroughoutDevelopment]
Improve[RealisticScheduleEstimation]
Maintain[EstimateScheduleThroughoutDevelopment]
ClearRolesAndResponsibilitiesAssignment
ContractApprovalWithCustomer
ClearProjectSuccessCriteriaAndBoundary
Minimize[TechnicalComplexity]
Process
Improve[AdequacyOfTasksAndMethods]
Improve[ProjectManagementCapability]
Improve[FormalRiskManagementPractice]
Product
Attain[CompleteBusinessSpecification]
Reduce[ErrorFromRequirements]
Improve[CompletenessInRequirementSpecificationDocument]
Human
Improve[CompetencyOfTeamMembers]
Improve[Customer/UserParticipation]
Reduce[Customer/UserDissatisfaction]
Improve[OverAllTeamPerformance]
Improve[EffectiveCommunicationAndCoordination]
Improve[ManagementCommitment]
Environment
Improve[StabilityOfTheOrganization]
Improve[AdequateDevelopmentFacilities]

Risk factors identified from the project context that directly obstruct the goals are also outlined in Table I. Our team observed that some factors influences several risk events and obstruct more than one goal compare to other risk factors. These factors are important and require extra attention to control as early as possible. Table II shows the high prioritised risk factors and associate event identified from the project.

TABLE II: HIGH PRIORITISED RISK FACTOR AND EVENT

Risk factor	Event
• Under-specified, unstable, incorrect, and infeasible requirements • Incomplete requirement specification document • High level technical complexity • Software-to-be demands several external links with other parts of the customer • Unclear business process • Practitioner inadequate domain knowledge • Customer/user passive participation • Local environmental problems • Missing information from the demanded legislation • New development platform	• ErroneousRequirements, • Technical Infeasibility, • ProjectComplexity • IncompencePractit-ioner, • UnclearSystemVision, • IneffectiveCommunicati on • PassiveCustomer/UserIn volvement • Customer/UserDissatisfa ction • BudgetOverruns • ScheduleOverruns

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. Therefore, 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. Therefore 35 out of the 165, i.e., *approximately 22% of the system requirement were erroneous*. There are several causes for these requirement errors such that the project was inherently complex due to the large number of links among several components within the system under development, as well as with external system components, and the lack of domain and system knowledge among the project members. Besides requirements errors, some other risk factors were observed such as customer/user representatives passive involvement during requirement elicitation process, information regarding regulatory compliance was partially missing, and new development platform was required to support the specific device for the project. There were also some local environmental risk factors: power shortage and interrupted internet bandwidth. Therefore, 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 with the project manager and requirement engineer. The project manager mainly focused on the *human* as an agent to resolve these risks, because most of the identified risks are caused by humans. Initially the focus was to prevent the risk completely (if possible), otherwise reduce it as much as possible to satisfy the goals. Unfortunately, due to the inherent nature, all risks were not resolved. This is because some of the requirements were unclear by both customer and developer site. The project manager considered it as being a common situation in offshore projects. However, due to the schedule pressure, these requirements can pose sever problems later on. But *no immediate actions* were taken in respect to these requirements. It was rather decided to obtain more information in particular about the component dependencies and the specific legislation context. However, some of the requirements errors are recovered by reviewing the goals and system vision together with the end user. Two requirements were removed due to their technical infeasibility after approval from the user. Therefore, out of the 35 requirements, 15 requirement errors were completely solved. The remaining requirements required further analysis, e.g., in the later development phases. In addition to the requirements error some other risks such as bandwidth problem, inadequate knowledge about programming platform were also resolved. E.g., the project manager recommended to assigning additionally one or two new members with expertise on the system-to-be require for the project. The project manager further recommended to the customer/user to get more actively involved in rest of the requirements engineering tasks, as well as in later stages of the development.

F. Discussion

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

Integration of Risk Management into Requirement Engineering

There are indeed strong dependencies among requirements and risk artefacts. In particular, business specification, system vision, and requirements closely support goal and risk identification activities. Risk management as part of requirements engineering contributed to producing a complete requirement specification document. Furthermore, controlling human and environmental factors such as practitioner domain knowledge, customer/user participation, adequate development facilities for the effectively completion of the development activities. Activities of GSRM did not introduce any conflicts or significant unnecessary burden to the requirements engineering activities, as well as not consuming much extra efforts. A project manager with some background knowledge and experience in risk management were able to perform the risk management activities to a sufficient level. Furthermore, the requirement engineers also contributed to the goal and risk identification and later on also to the risk control and monitor activities, in particular in reducing requirement 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 directly solved*) with the support of GSRM We observed that risk assessment results help to prioritise requirements so that high prioritised requirements get early attention to the later development phase. Therefore, we can conclude that the observed results support hypotheses *H1* and *H3*. Moreover, in the evaluation process, we considered risks from both the technical and non-technical perspectives and similarly the risk control and monitoring actions where all executed in a holistic view. Thus the result of the case study also supports hypothesis *H2*.

Overall observation of GSRM from the case project

GSRM is a goal-driven approach and therefore eases the practical execution of risk assessment and 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 [ErrorFromRequirements] and maintain [EstimateBudgetThroughoutDevelopment]. However impact of the event to the goals is different. Furthermore, our team also observed that the same risk event can be a risk factor in another context. For instance ErroneousRequirements as a consequence of requirements faults such as under-specified, unstable, incorrect and infeasible requirements and further being risk

factors for the schedule or budget overruns. Therefore, the consequences and causes of a risk event may vary from context to context.

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

- The close questions and the brainstorming sessions are effective techniques for the risk identification
- Development component-element-factor hierarchy eases to identify and categorise the goals and risk factors.
- Goal refinement is difficult as there may be several sub-goals under one parent goal. Huge number of sub-goals can increase complexity for handling it through assessment and treatment. Risk assessment is complex and this makes a project manager averse. Therefore, simplified estimation technique is desirable for software development.
- The effort involved in developing risk artefacts, e.g. the risk monitor sheet 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 artefacts in particular for projects with tight schedule and high budget pressure.

We treat the last two remarks as limitations of GSRM. These factors can increase the overall risk management effort in requirements engineering. At the early stages of requirements engineering, it is also not possible to plan and control all identified risk due to inadequate knowledge of the problem space of the system-to-be and uncertainty about the future project activities. Additionally, if a project contains many risk factors, then modelling the obstacle and maintain risk status report would consume more time in the project. 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 close and open questions and the brainstorming sessions. The result of the identified risks was compared with published risk factors [7, 21] from similar development environment to augment to our limited experience data. The risk factors and the consequences from the case project coincide with the published survey risk factors of offshore project. E.g., requirement errors, in particular unstable and incorrect requirement, inadequate project domain knowledge, are also highly ranked by other research results. The local environmental context highly influences the risk factors; therefore we do realize that project risks are cultural dependent [21], which is also observed in related research.

VI. RELATED WORKS

Several works in the literature already contributed to the area of software risk management. The core initial contribution of risk management into a single framework was done by Boehm [3] in his spiral model. Following the spiral model there were many contributions each describing well-documented risk management approaches, such as Karolak's SERIM method [12] and Konito's Riskit [13]. Researches also have contributed to identify software risk factors in particular in offshore development environment [7, 21]. All contributions put emphasize on performing risk management as early as possible, but comprehensive detailed guidelines are still missing. Thus far, some works have tackled the problem of considering risk management as part of early development activities [1, 13]. Ansar et al [1] contribute by introducing organizational setting besides requirement risk by extending Tropos and focusing more on the early stages of requirements engineering. Procaccino et al., [18] as stated identified seven early development factors and discussed how these contribute to the success or failure of a software project. Ropponen et al. [20] conducted a survey to investigate six software development risk components and showed how to provide assistance in addressing these components.

In the area of goal oriented requirement engineering, goal models generally shows the system's functional and non-functional goals that contribute to each other through refinement towards software requirements and environmental assumption as constraint to support the goals. Requirements are the lower level goals under the responsibility of a single agent of the system-to-be. Goal oriented requirements engineering already recognised as an essential component for all phases of requirements engineering life cycle. KAOS (Keep All Objective Satisfied) aims to model not only what and how aspect of requirements but also why, who, and when [15]. The model also includes obstacle as unintended risks that associates with undesirable behaviour and anti goal as intended risk that associates with intended risk. Other goal model such as i*, Tropos [4] models and analyses requirements both the system-to-be and its organisational environment by using concept of actor, goal, task, resource, and social relationships to capture stakeholders' intentions in an organisation.

In GSRM [8, 9], we follow the basic concepts from KAOS. Note that KAOS also includes risk management activities within requirements evaluation with main focus on ensuring the completeness of the requirement specification. But GSRM focuses comprehensive detailed on software development risk management in particular from the early development components where requirement completeness is one of the main goals. Our main focus is to integrate risk management activities into early requirements engineering activities. The result from the case study showed that risk management can indeed be well-integrate into requirements engineering.

VII. CONCLUSIONS AND FUTURE WORKS

The paper presents GSRM, a modelling framework to manage software development risk in the early stages of requirement engineering. The model was implemented in a running offshore software project to analyse the effectiveness

of GSRM. The results showed that GSRM can be well-integrated with requirement engineering activities and effectively contributes to reduce requirements errors. GSRM is particularly beneficial at the early phases of the development because at this stage the project generally focuses on formulating and understanding the core goals for the system-to-be. 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 (Bangladesh). 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 observation from the participants within the case study.

ACKNOWLEDGMENT

The work is partly supported by the German Academic Exchange Service (DAAD), Germany, and by the Telenor COLAB – A part of the Telenor Connected Objects Project.

REFERENCES

[1] Ansar, Y. and Georgina, P., Modeling Risk and Identifying Countermeasure in Organizations, In Proc. of the first International Workshop on Critical Information Infrastructures Security, Springer, 2006.

[2] Boehm B., Software Engineering Economics, 1981.

[3] Boehm, B., Software Risk Management: Principles and Practices, IEEE Software, Vol. 8, pp. 32-41, January 1991.

[4] Bresciani, P. Perini, A. Giorgini, P. Giunchiglia, F. and Mylopoulos, J., Tropos: An Agent-Oriented Software Development Methodology. Journal of Autonomous Agents and Multi-Agent Systems, 8(3):203–236, 2004. ISSN 1387-2532.

[5] Fernández, D. and Kuhrmann, M., Artefact-based Requirements Engineering and its Integration into a Process Framework, Technical Report, number TUM-I0929, Technische Universität München, 2009.

[6] Glass. R. , Software Runaways: Monumental Software Disasters. Prentice-Hall, 1998.

[7] Iacovou, C. L. and Nakatsu, R., A Risk Profile of Offshore-outsourced development project, Communication of the ACM, Vol 51, No. 6, June 2008.

[8] Islam, S. , Software Development Risk Management Model-a goal Driven Approach, Doctoral Symposium, In Proc. of the 7th ESEC/FSE, Amsterdam, The Netherlands, 2009.

[9] Islam, S., Joarder, M.A. and Houmb, S.H., Goal and Risk Factors in Offshore Outsourced Software Development from Vendor's Viewpoint, Proceedings of the Fourth IEEE International Conference on Global Software Engineering, Limerick, Ireland , 2009.

[10] Jiang, J. and Klein, G., Software Development Risks To Project Effectiveness, Journal ofSystems and Software, Volume 52, Number 1 (2000) p. 3-10.

[11] Jensen, F., An introduction to Bayesian Network. University College London: UCL Press, 1996.

[12] Karolak, D., Software Engineering Risk Management, IEEE Computer Society Press, 1996.

[13] Kontio, J., Software Engineering Risk Management: A Method, Improvement Framework, and Empirical Evaluation. PhD thesis, Helsinki University of Technology, 2001.

[14] Linberg, R., Software Developer Perceptions About Software Project Failure: A Case Study, The Journal of Systems and Software, Vol. 49, Issue 2/3, 1999.

[15] Lamsweerde van A., Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009.

[16] McConnell, S., Rapid Development. Microsoft Press, 1996.

[17] McManus J. , Risk Management In Software Development Projects, Elsevier Science Inc, 2004.

[18] Procaccino, J. D, and Verner, J. M. Case Study: Factors for Early prediction of software development success; Information and Software Technology; Vol. 44, 2002.

[19] Pfleeger, S. L., Risky business: what we have yet to learn about risk management, Journal of Systems and Software, Volume 53, Issue 3, 15 September 2000, Pages 265-273.

[20] Ropponen, J. and Lyytinen, K. Component of Software Development Risk: How to address them? A project manager survey, IEEE Transactions on Software Engineering, Vol.26, Issue: 2, Feb 2000, 98-112.

[21] Sheng Z., Nakano M., Kubo S., and Tsuji H., Risk Bias Externalization for Offshore Software Outsourcing by Conjoint Analysis, New Frontiers in A. I , Volume 4914, 2008.