Software Development Risk Management Model – A Goal Driven Approach

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

ABSTRACT
Software development project is often faced with unanticipated problems which pose any potential risks within the development environment. Controlling these risks arises from both the technical and non-technical development components already from the early stages of the development is crucial to arrive at a successful project. Therefore, software development risk management is becoming recognized as a best practice in the software industry for reducing these risks before they occur. This thesis contributes for a goal-driven software development risk management model to assess and manage software development risk within requirement engineering phase.

Categories and Subject Descriptors
D.2.9 **[Software Engineering]**: Management| Life cycle, Programming teams, Software quality assurance

General Terms
Software development

Keywords
Software development risk, risk management, goal-driven modeling, and risk modeling.

1. PROBLEM STATEMENT
Despite the advancements in technology, software development projects face similar problems repeatedly. A study found that 20% of software projects failed and that 46% experienced cost and schedule overruns or significantly reduced functionality [11]. The software system is constantly error prone which poses for several problems including constant expansion of the system scope, missed business needs, cost and schedule overruns and even project failure. Research suggests that failed projects suffer from the poor management of people related problems rather than technical problems [11]. Humans involve during every link of software development activities incur error, make wrong assumption, show poor team performance, etc certainly influence for any potential risk. End-user involvement is one of the most important contributors to successful project development. It is imperative that risk management need to be considered a holistic view that spans both technical and non-technical dimensions based on the development components [7].

Copyright is held by the author/owner(s)..
ESEC/FSE'09, August 24–28, 2009, Amsterdam, The Netherlands.
ACM 978-1-60558-001-2/09/08.

Software risk management generally focuses on goals relating to schedule, cost, and quality. Nevertheless, certain goals such as offshore and co-ordination projects work within different cultures and locations, supporting critical business process, compliance with the demanded regulations, security and safety have gained importance recently. Though there are several contributions in the area of software risk management, still a lot need to be done for integrating in the development process. Risk management is usually performed during design or later development phase. But in that case, counter measures may introduce revision of the whole design or alteration of the elicited system requirements and related artifacts. These may lead unanticipated problems during the development and jeopardy to the project success. Considering risk management since the early phase can avoid such problems and contributes to mitigate these risks. However, comprehensive details are still missing in the literature regarding the integration of the risk management during requirement engineering phase. We summaries the following research questions:

a) How risk management can systematically integrate at early development stage to significantly improve the overall software project outcomes?

b) What are the main goals require to be attained during early stage from the perspective of project success?

c) How risks that obstruct the goals assess, trace and control from the early technical and non-technical development components?

2. RESEARCH CONTRIBUTIONS
To answer these questions, the aim of this research is to propose a modeling framework to support software development risk management, considering both technical and non-technical components, during the early development stage. The research contributes a goal- driven software development risk management modeling (GSRM) framework to assess, reason, control, and trace software development risk. The main focus is to integrate risk management activities within Requirement Engineering (RE) phase so that risks are identified and controlled from the early stage. We delimit scope for this research within the context of business information system focusing elicited business, user and system requirements, project constraints (e.g. schedule, budget), development process, resulting software product, and human & organizational factors. The reasons for considering the approach within RE are that poor requirements are one of the main causes of the project failure [6], cost relates to fix errors during the testing phase is fifty times more than the cost of fixing in RE phase [3]. We strongly believe that if software development risks manage during

the RE phase then it can effectively contribute for the completion of the software project.

The work prefers to consider the existing modeling techniques to accommodate the risk management activities rather than developing a new one. We have chosen goal modeling language for software development risk management. Goal modeling language such as KAOS, i*, and Tropos has long been recognized in Requirement Engineering community for elicit, analyze, negotiate, document, and modify user and system requirements. But the methodologies do not consider software development risks during the requirements phase. However some recent contributions such as [2], [5] focus on risk management activities in RE phase. In [2], Ansar et al. contribute towards a Tropos goal risk framework by extending Tropos methodology. However the approach does not consider software development risk factors from the early development components rather only consider risk relating to the requirements. Similarly in [5], Boness et al. also considered risk relating to requirements during later stage of RE. Our approach consider beyond on these concepts. We extend KAOS (Keep All Objective Satisfied) goal model methodology to accommodate risk management considering both technical and non-technical early development components. KAOS defines obstacle as a construct that can be used to identify undesirable behavior against the strategic interest of a stakeholder [15]. GSRM adopts this construct and defines software risks as obstacles that contribute negatively to fulfill specific development goals. GSRM undertakes goal and obstacle concept from the KAOS and further extends with risk assessment and treatment for managing software development risk.

3. PROPOSED APPROACH

3.1 Goal-Driven Software Development Risk Management Model (GSRM)

GSRM is a combination of four layers to manage risk in software development risk [8]. The advantage of using layer based concept is that any techniques can be applied in any layer to perform its task without affecting the other layers. This section provides a short overview of these four layers.

Goal layer. GSRM starts with identifying, elaborating and modeling the goals based on the early development components from the perspective of project success. There are several directions to define project success including project that meet agreed business objectives and complete on time and within budget, satisfy user, technically realistic requirements, realistic estimation of schedule and cost, etc [1, 13]. Therefore, to attain project success, goals require identifying and elaborating from the early technical and non-technical development components. In GSRM, we identify goal based on these definitions such as clear business needs and project scope, realistic time and budget estimation, error free user and system requirements, and so on. The goals involved in the development activities must be achieved, maintained, ensured, managed, improved and reduced depending on its nature to carry out a successful development project [15] such as ensure [clear business needs and project scope], maintain [realistic schedule], maintain [stay under budget], manage [human factors], reduce [errors from requirements], etc. These goals are sometimes high level and if require, can be stated at different levels of abstraction from higher level coarse grained to lower level finer-grained goal. Satisfaction of these sub-goals certainly attains the main goal. This allows to model early development components where the goal fulfillment provides strong support within the development environment.

Risk-obstacle layer. The risk-obstacle layer identifies the potential software development risk factors as obstacles from the early development components that negatively influence the goals. Obstacles are the dual notation to the goals (e.g. undesirable ones) [15]. Same obstacle obstructs more then one goal such as misinformation, human errors, requirement error, ineffective development process obstruct goal such as maintain [realistic schedule], reduce [errors from requirements]. Generally, these risk-obstacles identification is done through checklist, questionnaires and brainstorming session with the stake holders. In GSRM, we follow a set of questionnaires based on the early development components as well as brainstorming session to identify these risks. The identified risks are analysis further through the assessment layer.

Assessment layer. The assessment layer is used to precisely annotate the individual risk obstacle. The main purpose is to analyze the risk event caused by the identified risk factors. Each risk event characterized with two properties: likelihood and severity. Likelihood specifies the possibility of a risk event occurrence and models as a property of the risk event. And severity quantifies the negative impact by the risk event. Same risk factor can pose more than one risk event as well as same risk event can obstruct more than one goal. This representation allows to model situation where an event influences by more than one risk factor and at the same time negatively impacts on single or several goals. An obstruction link is established from risk event to the specific goal it obstructs. The layer considers risk metric values to identify the likelihood of the risk event by measuring the risk factors. Same measurement level follows for the risk metric relating to risk factors, risk event likelihood and risk severity. It makes the whole risk analysis process simple and effective during early stage. We use Bayesian subjective probability to determine the likelihood of the individual risk event caused by single or several independent risk factors. In GSRM, risk analysis explicitly considers the risk events having only negative impact to the goals. Note that we do not consider any event that has positive influence to the goal. Therefore, if the risk events are improbable then it implies that the confidence of the related goal fulfillment is high. Hence, risk event likelihood and severity give us certain belief about the dissatisfaction (DSAT) and satisfaction (SAT) of the goal fulfillment within development environment. The risk assessment layer finally prioritized the risk based on the likelihood, severity and its influence towards goal dissatisfaction through obstruction link.

Treatment layer. Finally, the treatment layer identifies the possible control actions and selects the most suitable ones to mitigate the risk and there by to attain the goal. Once the goals, risk factors and risk events are identified and analyzed by goal, risk -obstacle and assessment layer, then GSRM focuses to implement the suitable cost effective counter measure as early as possible. Therefore control actions are agent within the development environment such as human, tools, etc define as active system components perform specific role to satisfy the goal [15]. Different mitigation strategies follow to control the risk. Additionally it is also necessary to analyze

the cost-benefits before implementing a suitable control action. In GSRM, we allow relation among treatment, risk -obstacle and goal layer. The link establishes from control action to goal is called contribution link facilitates tracing from control action to the goal. Therefore, it is useful to model, reason, and trace situation within the development environment where a control action adopts to mitigate a risk and contributes positively to attain the goal. Figure 1 depicts different layers of GSRM. Note that, we follow the same notations for goal, obstacles and treatment agent in GSRM as in the KAOS model [15].

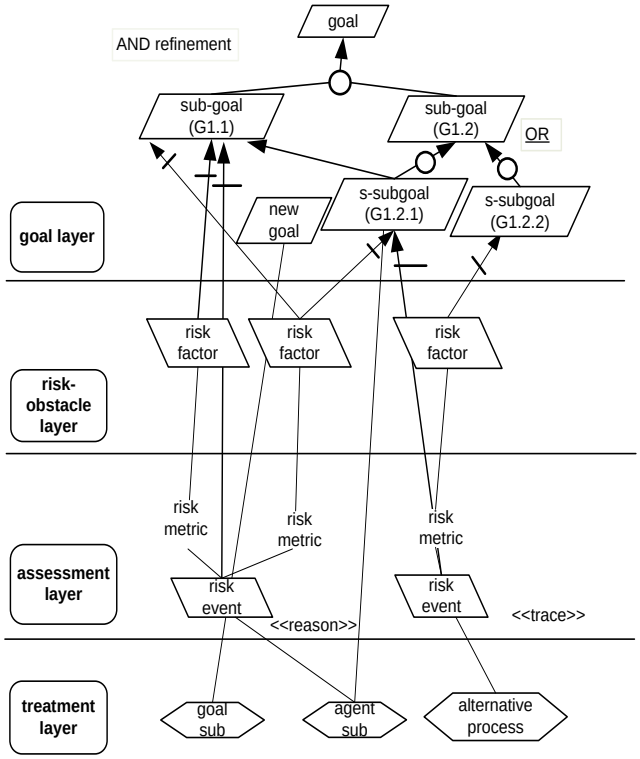

Figure 1. Goal-driven software development risk management model

3.2 GSRM within the context of RE

We propose to begin the goal and risk identification activities nearly in parallel to the requirement elicitation activities. More specifically when business needs identify through business modeling and system vision prepares for customer approval, then GSRM starts with the goal identification and elaboration. Generally, system vision summarizes the elementary artifacts of business specification including overviews of business rules, domains, business goals, business process, project scope, and related features. Therefore goals and risks relating to the business needs and project scope identify at this stage. Although, note that if require, certain goals and associate risk factors, especially from the early non-technical development components identify before elicitation of the business specification and system vision. It allows to identify risk before the definition of the user and system requirements. As mentioned, the activities involve within GSRM are iterative, depending on the input artifacts, several iterations can carry out within requirement elicitation, analysis and validation. However, at the end of RE activities when requirement specification continues for the subsequent development phase, then certain relevant goals such as errors free requirements, accurate and competence project team members, adequate development facilities, and so on are attained. Figure 2 shows the model within the context of RE. During the initial iteration, goals and risks identify from the business specification, system vision as well as from the other non-technical development components. Further iterations identify risk from the elicited user and system requirements and other relevant artifacts.

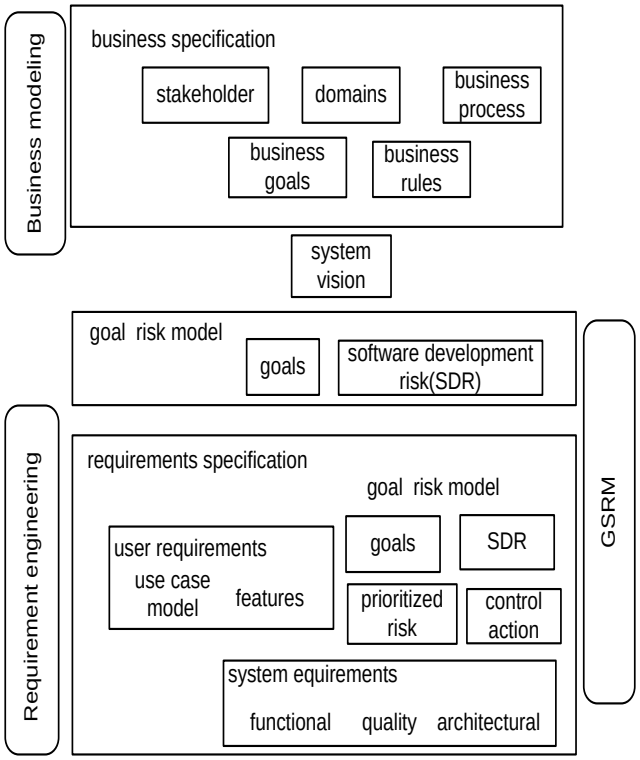

Figure 2. GSRM within the context of RE

3.3 Expected Outcome

The proposed model provides quantitative evidence that certain goals within the early development components such as error free requirement, active customer/user involvement through out the development, realistic estimation of schedule and budget, clear business needs and project scope, and manage human & organizational factors are fulfilled. These facilitate to reduce unanticipated problems within subsequent development phases. However, we need to validate the effectiveness and feasibility of the proposed framework to manage risk during at early development stage.

4. PROGRESS

We have identified early development components, goals and sub-goals based on the published experience paper considering the perspective of project success. Furthermore, a set of questionnaires were already developed to identify the risk factors that obstructs these goals. These questionnaires will be used to conduct a survey study to the experienced software practitioners. Currently, we are planning to conduct a survey study within the context of offshore outsourced software development. Initially, the survey context is from a developing country with limited IT infrastructure facilities.

At this stage of the research, we have chosen eight Bangladeshi software companies as vendor that produce software for its offshore clients for the survey purpose. It facilitates to revise the goals and identify risk factors from a different culture with less advancement in software development infrastructure. Our survey study is based on Delphi survey process to obtain the possible risk factors and rank the top ten risk factors considering early development components. Identified risk factors are analysis through the assessment layer and control actions are proposed through the treatment layer. Afterwards, we would like to apply the model in running software development projects to determine the feasibility and validity of the approach.

5. RELATED WORKS

Lots of works have already been done in the area of risk identification, analysis and the overall software risk management. Short overviews of these works that are relevant to our work are given below. Boehm, one of the pioneers in the area of software risk management, proposed risk driven spiral model in 1991, consisting of an iterative set of activities [4]. Since then, several works contributed around the theme. Karolak proposed Software Engineering Risk Management (SERIM) by four interconnected risk tree based on 81 risk factors with three main risk elements technology, cost and schedule [9]. Kontio emphasized on effectiveness of group work (including the brainstorming sessions) by the Riskit methodology to identify the stakeholder goals and risks that threaten the goal [10]. There is however a consensus that the risk management must comprise two general phases risks assessment and control. In GSRM, we include goal identification and elaboration step, similar as Riskit, before risk assessment and control. Islam et al. provide the short overview of the GSRM in [8]. It is generally agreed that, to be successful, the activities should perform iteratively involving repeated risk assessment and project-wide risk mitigation. In GSRM, it is also possible to perform several iterations of software development risk management depending on the nature of the input artifacts.

There are also several contributions on risk identification and analysis. Well known top-ten risk list are provided by Boehm [4] in 1991 and more extensive list published by Schmidt et al. [14] in 2001. These lists are usually compiled from the surveys of the experienced stake holders. Research also showed that perception of risk varies between stake holders, overtime, within project context and between cultures [14] [16]. However most of these studies were conducted in developed countries with sophisticated IT infrastructure facilities. But, because of the rapid increase of the offshore outsourced software development the survey requires to focus on the risk factors from the developing countries with limited IT infrastructure facilities [12]. Some researches have already contributed to identify the risk factors from the developing country like China and India [12] [16]. We focus to identify the early software development risk factors from Bangladesh having limited IT infrastructure facilities. Furthermore, little work has been undertaken on the potential effects of these risk factors. To address this issue, our survey study not only identifies the risk factors but also quantify the potential effects of these factors. Furthermore we will also implement the proposed model to running software development projects.

6. REFERENCES

[1] The Standish group report chaos http:// www. standishgroup. com.

[2] Y. Asnar and P. Giorgini. Modelling risk and identifying countermeasures in organizations. In Proceedings of 1st In. Workshop on Critical Information Infrastructures Security, 2006.

[3] B. Boehm. Software Engineering Economics. Prentice Hall PTR, 1981.

[4] B. Boehm. Software risk management: Principles and practices. IEEE Software, 8(1):32{41}, 1991.

[5] K. Boness, A. Finkelstein, and R. Harrison. A lightweight technique for assessing risks in requirements analysis. In The Institution of Engineering and Technology, 2008.

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

[7] S. Islam and W. Dong. Human factors in software security risk management. In Proceedings of the first international workshop on Leadership and management in software architecture (LMSA), in ICSE08, 2008.

[8] S. Islam, M. A. Joarder, and S. H. Houmb, Goal and risk factors in offshore outsourced software development from vendor's viewpoint. In Proceedings of the empirical experiences, metrics and tools for project management in globally distributed software development (EGSD) in ICGSE 2009.

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

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

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

[12] K. Na, J. Simpson, T. James, X. Li, T. Singh, and K. Kim. Software development risk and project performance measurement: Evidence in Korea. J. System. Software, 80(4):596{605), 2007.

[13] J. D. Procaccino, J. M. Verner, S. P. Overmyer, and M. E. Darter. Case study: factors for early prediction of software development success. Information and Software Technology, 44(1):53 {62}, 2002.

[14] R. Schmidt, K. Lyytinen, M. Keil, and P. Cule, Identifying software project risks: An international delphi study. J. Manage. Inf. Syst., 17(4):536, 2001.

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

[16] H. Tsuji, A. Sakurani, K. Yoshida, A. Tiwana, and A. Bush, Questionnaire-based risk assessment scheme for Japanese offshore software outsourcing. In SEAFOOD 2007, LNCS 4716.