Software Architecture Analysis for Risk Management

Byeongdo Kang*, Roger Y. Lee**

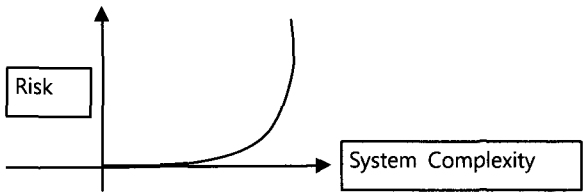
Abstract Management of risks is critical issue in the project management and it is important to ensure that risk management is done in a sensible way. Risk analysis is an activity geared towards risk mitigation in risk management technique. Many techniques to manage, analyze and reduce risks have been done previously but only few have addressed the design analysis to reduce risk and none have attempted to analyze architecture to manage risks. In this paper we try to find a solution through various analyzing various software architectural design concepts. We follow Pressman's method of analyzing architecture design, and then alter it to identify risks which are used in risk analysis process further in risk management process. The risks assessed are analyzed later in the risk management cycle.

Key Words : Risk Analysis, Architecture Design, Risk Management Process

1. Introduction

The management of risks is a central issue in the planning and management of any venture. In the field of Software, Risk management is a critical discipline. The process of risk management embodies the identification, analysis, planning, tracking, controlling, and communication of risk. It gives us a structured mechanism to provide visibility into threats to project success. Risk management is a discipline for living with the possibility that future events may cause adverse effects[1]. Risk management partly means reducing uncertainty. Of course, reducing uncertainty has a cost associated with it. We need to balance such costs we could incur if the risk is not addressed. It may not be cost-effective to reduce uncertainty too much. If we are trying for a risk free system or risk reduction then the cost incurred would be more. So we should manage risk instead of getting rid of them. The need to manage risk increases with system complexity. Figure 1 demonstrates this concept by indicating that as the complexity of the system increases, both technical and non-technical (cost and schedule) risks increase[2]. Several methods to manage risks have been developed, but most of them focus on three basic constructs for software risk management developed at Risk Management Paradigm, Risk Taxonomy, Risk Clinic, and Risk Management Guidebooks by the Software Engineering Institute. The popular SRE is an initiator for continuous risk management(CRM) and team risk management(TRM)[3].

* 대구대학교 컴퓨터·IT공학부 교수
** Professor, Dept. of Computer Science, CMU, USA.



<Figure 1> Risk versus system complexity

2. Risk Management Techniques

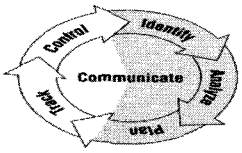
In this section we provide details about the other techniques we investigated. Section 2.1 describes our summary over Team Risk Management approach. Section 2.2 describes Software Risk Evaluation approach for Risk Management. Section 2.3 presents continuous risk management approach for Risk Management.

2.1 Team Risk Management Approach

Team Risk Management[5] is a new paradigm developed by The Software Engineering Institute(SEI), a federally funded research and development center and part of Carnegie Mellon University in Pittsburgh, Pennsylvania. Team Risk Management is a paradigm for program or project management by developing a shared product vision, focused on results, and using the principles and tools of risk management to cooperatively manage risks and opportunities. Team Risk Management establishes an environment built on a set of processes, methods, and tools that enable the customer and supplier to work together cooperatively continuously managing risks throughout the life cycle of a software dependent development program. Team Risk Management implements SEI Risk Paradigm functions for risk management: adding the principles of shared product vision and teamwork to make up the functions of Team Risk Management. Team Risk Management adds two new functions, **Initiate** and **Team**, to recognize both the required cultural paradigm shift and the emphasis on teamwork. SRE's risk paradigm is shown in figure 2, Team Risk Management model will have all the phases of that in figure 2 and also additional components customer and supplier in the identification phase. Each function has a set of activities that are backed by processes, methods to improve communication and teamwork.

2.2 Software Risk Evaluation Service Approach

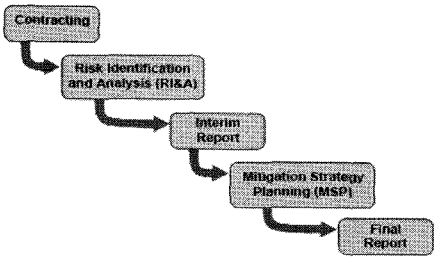
The SRE process has been in evolutionary development at the SEI since 1992 and has been used on over 50 Department of Defense (DoD) and civil (federal and state) contractors and program offices. The SRE addresses the identification, analysis, planning, and communication elements of the SEI Risk Paradigm. Figure 2 show the SRE paradigm. The SRE, while not the only identification method available, is typically the initial and most prominent one used on a project. The analysis element is also covered fully by SRE activities. Planning elements are partially addressed through the construction of high level mitigation strategy plans. The SRE also contributes significantly to the communication element. The remaining elements of the paradigm, tracking and control, are not addressed during an SRE. Figure 3 shows the SRE's five phases－Contracting, Risk Identification and Analysis, Interim, Report, Mitigation Strategy Planning(MSP), and Final Report. By implementing a Software Risk Evaluation, management improves its ability to assure success through the creation of a proactive risk management methodology.



<Figure 2> SRE's risk paradigm

2.3 Continuous Risk Management

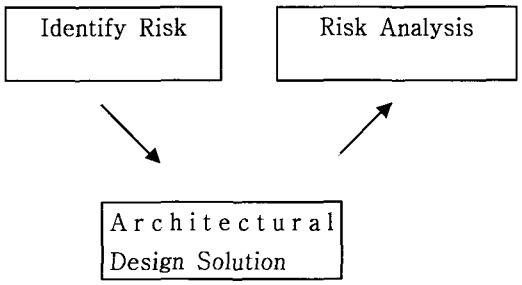
Continuous Risk Management is a software engineering practice with processes, methods, and tools for managing risks in a project[4]. It provides a disciplined environment for proactive decision-making. Continuous risk management is built on set of principles that if followed gives an effective approach of managing risk. Figure 3 shows the seven principles of CRM's risk management. The SEI SRE establishes a baseline set of risks, for the start phase of Continuous Risk Management and extends it to customer-supplier relationships. When using continuous Risk Management, risks are assessed continuously and used for decision-making in all phases of a project. Risk are carried forward and dealt with until they are resolved or they turn into problems that are easy to handle.



<Figure 3> Phases in SRE

3. Our proposed Approach to Risk Management and Analysis.

Figure 4 shows the overview of our approach. Each step and sub-steps are explained further.



<Figure 4> Overview of our approach

3.1 Identify Risks

Before risks can be managed, they must be identified. We should identify risks before they cause problems. Identification of risks could be done by any approach. But here we are using SEI's techniques for surfacing risks. Preferably Identification of risk should give us a risk statement or a component so that we can apply our next step of constructing architecture risk pattern, so the component should derive directly or indirectly risk patterns. We are basically concerned with managing risks and not much concerned about identification risks. Risk identification phase includes following steps:

Risk checklists. Identified risks are broken down into a structure of checklist. Risks identified at each level of software engineering process can be subdivided and added to respective phase like risks in requirements, design, coding, testing and integration etc. This may include prioritizing risks by categorizing them into likely, less likely unlikely etc.

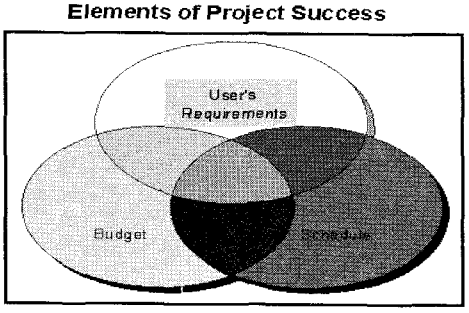
Risk assessment. The risk assessment is done by assessment team. It reviews the

project profile by project manager conducts the interviews. After the interview session the risks are recorded.

3.2 Risk Assessment

Risk Assessment in a project is the most difficult phase of all to be carried out. From the definition we gave elsewhere, a risk is a combination of uncertainty and constraint. Constraints are usually difficult to remove, though it is important to understand them. Figure 5 shows three circles which indicate the most important quality measurements of the project. And the most important of these is **meeting the users' requirements.** How many projects do we know where the team has not bothered with ensuring that the users understand what is proposed, but has got on with the interesting part of the project - actually implementing it.

Existing methods for software architecture analysis of flexibility looks for scenarios that are likely to happen and then assess their impact. Our approach is based on building on the definition of a number of categories of scenarios: 1) Scenarios that have external effects, 2) Scenarios which in turn affect the system itself, 3) Scenarios that change the relationship, 4) Scenarios that changes the internal structure of the system and 5) Scenarios that introduce conflicts.



<Figure 5> Risk Assessment

In the first case, the owner of the system initiates the scenarios and in second case, the scenarios are initiated by the owner of another system. The reasons we have decided to view a system's software architecture in term of scenarios are 1) The role of the system within the environment and 2) The internal structure of the system.

This brings us to the third and fourth categories, which are the scenarios that require adaptations. Most radical type of changes not only do they affect the internals of a number of components, but they also affect the way in which these components collaborate. The fifth and final category we have defined are the scenarios that introduce conflicts. This problem may occur in a situation where components are shared by several systems. When a system then requires changes to one of these components, different problems may be introduced. This means that the corresponding scenario cannot be implemented as such. The changes that are incurred by scenarios may come from a number of sources. We distinguish the following four sources of changes:

- Functional requirements
- Quality requirements
- External components used
- Technical environment

The first source of changes is the set of functional requirements. Examples of changes in the functional requirements are features that have to be added or unwanted functionality that has to be deleted. The second source of changes is the set of quality requirements. Changes that can occur in the quality requirements are, for instance, the need for increased performance or the need for increased security. The third source of changes is the set of external components

used. When these components change, the system may have to be adapted. This situation often occurs when a system makes use of components of another system, or when generic components are shared by a number of systems. The main problem is that these components are often owned by others, which means that the owner of the system concerned does not have full control over them. As a result, these changes are sometimes forced upon the system and its owner. Something similar applies to the fourth source of changes, namely the technical environment. In more and more organizations the technical environment is shared by several systems. So, just like the external components it could happen that the system has to be adapted to changes that are initiated by others than the owner of the system concerned. This is something that has to be taken into account when analyzing a system's inflexibility. The best way to use this diagram is to start by identifying scenarios by reading documentation and interviewing stakeholder. This leads to an initial set of scenarios. The next step then is to classify these scenarios. This classification provides insight into the completeness of our initial set of scenarios.

Estimate scenario risk factor:

Use the Rational Rose tool as front end tool for transforming scenarios into UML models. The tool automatically constructs the Markov chain that represents the control flow graph of the active components and connectors in a specific scenario based on the textual representation of the UML sequence diagrams. The scenario risk factor for each severity level is computed using this Markov chain and the estimated values of component and connector risk factors.

Prioritize the scenarios:

Establish which of those Risk Scenarios should be eliminated completely, because of potential extreme impact, which should have usual management attention, and which are minor to avoid detailed management attention.

Estimate use case and overall system risk factors:

The risk factors of each scenario in a specific use case are aggregated to calculate the use case risk factor. Using the risk factor for each use case, the tool calculates the overall system risk factor.

4. Case Study

We have selected a case study of an ATM to discuss the applicability of the proposed methodology. Transaction on ATM machines is a critical real-time application. An error in the software operation of the device can cause loss transactions, money, and lead dangerous outcomes. Therefore, it is necessary to model its architecture in an executable form to validate the timing and deadline constraints. This executable architecture are also used, based on the proposed methodology, to conduct risk analysis.

5. Validity

This method involves the existing risk assessment and analysis method through architectural design it might prove to expensive since more resource would be spent on recognizing and analysis use case scenarios and estimating risk factor based on that. This also requires skilled architects and

analyst to build and analyze risk factors which is required for later phases. Though this might prove expensive compared to the cost of managing risk without analyzing architectural design, this would help in managing more risks as each scenario gives us idea of handling common risks and different risks. The architecture analysis which includes analyzing scenarios not only helps in risk management but also further in testing phases and other projects with same risks. In projects with few risks this method would expensive to implement.

6. Results and Conclusions

In this paper, we present architectural level risk analysis based on scenarios. The prototype enables early assessment of risk and hence makes it possible for the analyst to identify critical components/connectors and scenarios/use cases early in the software life cycle. The output of this method is development and testing effort based on critical use cases, scenarios, components, and connectors. Our future plan is to further extend the tool so that it computes and interprets quality metrics, even though the result maybe not as sensitive for early risk assessment. In addition, we plan to integrate the requirement analysis methodology into our prototype along with risks to allows precise estimation of the severity level for each architectural design.

Acknowledgements

This work is supported by Daegu University Grant(2007).

References

[1] H.F. Kloman, "Risk Management Agonists," Risk Analysis 10, 2(1990): pp. 201-205.

[2] "Know about the Latest & Hot Strategies in technology," http://technologystrategies.blogspot.com/, Retrieved March 27, 2006.

[3] R.C. Williams, G. J. Pandelios, S.G. Behrens, Software Risk Evaluation (SRE) Method Description(Version 2.0), December, 1999.

[4] R.L. Murphy, C. J. Alberts, R.C. Williams, R.P. Higuera, A.J. Dorofee, J. A. Walker, Continuous Risk Management Guidebook, SEI Carnegie Mellon University, 1996.

[5] R.P. Higuera, D.P. Gluch, A.J. Dorofee, R.L. Murphy, Julie A. Walker, R. C. Williams, An Introduction to Team Risk Management (Version 1.0), Special Report CMU/SEI-94-SR-1, May, 1994.

강 병 도 (Byeongdo Kang)

- 종신회원
- 1995년 서울대학교 이학박사 (전산과학 전공)
- 1988년~1998년 한국전자통신연구원 선임연구원
- 2004년 미국 CMU Research Associate
- 1998년~현재 대구대학교 교수
- 관심분야: 소프트웨어구조, 소프트웨어 프로세스, 소프트웨어 개발방법론

Roger Y. Lee

Dr. Roger Lee is the Director of Software Engineering & Information Institute and Professor of Computer Science at Central Michigan University, Mount Pleasant, Michigan, U.S.A. He received his Ph.D. in Computer Science from Shizuoka University, Japan and M.S. and Ph.D. in Computer Science from The University of Southern Mississippi, U.S.A. His current research interest areas include Requirements Engineering, Software Architecture, and Component-Based Development. Dr. Lee published more than 120 technical papers in international journals and conference proceedings. Dr. Lee's contributions to the field include the establishment of the International Association for Computer and Information Science (ACIS) and the International Journal of Computer and Information Science (IJCIS). He is currently serving as the Editor-in-Chief of International Journal of Computer & Information Science. Dr. Lee has served as conference chairs and program chairs and reviewed papers for many international conferences. He is a member of ACM, IEEE, and ACIS.