International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue IV, April 2018- Available at www.ijraset.com

Potential Factors for Software Risk Management

Dr Rachna Soni[1]
[1]*Associate Professor & Head Department of Computer Science & Applications DAV College For Girls,Yamunanagar*

Abstract: This paper analyses Risk Analysis and Management as a series of steps that helps a software team to understand and manage uncertainty. It analyses the probabilistic, heuristic Macro Model Approaches for assessing software risk by Measuring reliability. This focuses existing software process simulation which is the management of software development risks. It analyses study of software development risk factors for software development risk management, particularly in assessment and planning phases. This discusses the base model for Software Development Risk Factors designed to support decision making of risk management

I. INTRODUCTION

"Large software projects will never be without some risk, but if risks can be brought down to acceptable levels, that will be a good beginning" - Capers Jones, 1998.

As stated by Capers Jones, Chairman Software Productivity Research Inc. [Jon98],software has been the most troubling technology of the 20th century. Major software projects have the highest probability of being cancelled or delayed of any known business activity. Once deployed, software projects often display excessive error densities and low levels of reliability. In addition, software is achieving a very bad public reputation because of several highly publicized disasters. However, it is not a law of nature that software projects will run late, be cancelled, or be unreliable after deployment. A careful program of risk analysis and abatement can reduce the probability of major software disasters, and also shorten average development cycles at the same time.

Peter Kulik [Kul96a]states that awareness of Software Risk Management is growing throughout the industry. Proven industry results from implementation of Software Risk Management include earlier project completion, reduced project cost, more predictable schedules, and improved organizational capabilities. The practice of Software Risk Management includes both top-down and bottom-up perspectives. Projects that implement Software Risk Management become simpler and more focused. Further, a variety of tools are available for easier implementation.

A simple definition of a "risk" is a problem that could cause some loss or threaten the success of a project, but which hasn't happened yet. These potential problems might have an adverse impact on the cost, schedule, or technical success of the project, the quality of our software products, or project team morale. The objectives of software risk management are to identify, address, and eliminate software risk items before they become threats to success or major sources of rework. In general, good project managers are also good managers of risk. It makes good business sense for all software development projects to incorporate risk management as part of project management. There are a number of definitions and uses for the term risk, but there is no universally accepted definition. What all definitions have in common is agreement that risk has two characteristics

Uncertainty: an event may or may not happen an event has unwanted consequences or losses.

A. Risk Management Can

1) Identify potential problems and deal with them when it is easier and cheaper to do so - before they are problems and before a crisis exists.

2) Focus on the project's objective and consciously look for things that may affect quality throughout the production process.

3) Allow the early identification of potential problems (the proactive approach) and provide input into management decisions regarding resource allocation.

4) Involve personnel at all levels of the project; focus their attention on a shared product vision, and provide a mechanism for achieving it.

5) Increase the chances of project success.

Risk management is the application of appropriate tools and procedures to contain risk within acceptable limits. Risk management consists of several sub disciplines Risk assessment is the process of examining a project and identifying areas of potential risk. Risk identification can be facilitated with the help of a checklist of common risk areas for software projects, or by examining the contents of an organizational database of previously identified risks and mitigation strategies (both successful and unsuccessful) Risk

analysis involves examining how project outcomes might change with modification of risk input variables. Risk prioritization helps the project focus on its most severe risks by assessing the risk exposure. Exposure is the product of the probability of incurring a loss due to the risk and the potential magnitude of that loss. This prioritization can be done in a quantitative way, by estimating the probability (0.1 – 1.0) and relative loss, on a scale of 1 to 10. Multiplying these factors together provide an estimation of the risk exposure due to each risk item, which can run from 0.1 through 10 . The higher the exposure, the more aggressively the risk should be tackled. It may be easier to simply estimate both probability and impact as High, Medium, or Low. Those items having at least one dimension rated as High are the ones to worry about first. Risk avoidance is one way to deal with risk: don't do the risky things! You may avoid risks by not undertaking certain projects, or by relying on proven rather than cutting edge technologies. Risk control is the process of managing risks to achieve the desired outcomes. Risk management planning produces a plan for dealing with each significant risk, including mitigation approaches, owners, and timelines. Risk resolution is execution of the plans for dealing with each risk. Finally, risk monitoring involves tracking your progress toward resolving each risk item. The list of evil things that can befall a software project is depressingly long. The enlightened project manager will acquire extensive lists of these risk categories to help the team uncover as many concerns as possible early in the planning process. Possible risks to consider can come from group brainstorming activities, or from a risk factor chart accumulated from previous projects. The Software Engineering Institute has assembled taxonomy of hierarchically-organized risks in 13 major categories, with about 200 thought-provoking questions to help you spot the risks facing your project [Carr, 1993]. Capers Jones has identified the top five risk factors that threaten projects in different application sectors [Jones, 1994]. Table 1 illustrates those factors, and the approximate percent of projects to which they apply, in the management information systems (MIS) and commercial software sectors.

Table 1. Most Common Risk Factors for Various Project Types

Risk Factor	Percent of Projects at Risk
Creeping user requirements	80%
Excessive schedule pressure	65%
Low quality	60%
Cost overruns	55%
Inadequate configuration control	50%
Inadequate user documentation	70%
Low user satisfaction	55%
Excessive time to market	50%
Harmful competitive actions	45%
Litigation expense	30%

B. Assessing Software Risk by Measuring Reliability

A probabilistic approach has successfully assessed the reliability of the product [Lyu 1995,Schneidewind 1975, Musa 1998] . However, this approach addresses the reliability of the product, not the risk of failing to complete the project within budget and schedule constraints. These approaches could be used to assess risks related to failures of software projects. A concern with these approaches is that the resulting assessments arrive too late to economically correct possible faults, because the software product is mostly complete and development resources are mostly gone at the time when reliability of the product can be assessed by testing.

Heuristic approaches: researchers assess the risk from the beginning, in parallel with the development process. However, these approaches are less rigorous, typically subjective and weakly structured. Basically these approaches use lists of practices and checklists [SEI, 1996, Hall 1997, Charette 1997, Jones 1994] or scoring techniques [Karolak1996]. Paradoxically, SEI defines software technical risk as a measure of the probability and severity of adverse effects in developing software that does not meet its intended functions and performance requirements [SEI, 1996]. However, the term "probability" is misleading in this case because the probability distribution is unknown.

Macro Model Approaches: A group of researchers uses well-known estimation Models to assess how risky a project could be. The widely used methods COCOMO[Boehm 1981], and SLIM [Putnam, 1980] both assume that the requirements will remain unchanged, and require an estimation of the size of the final product as input for the models[Londeix 1987]. This size cannot be actually measured until late in the project.

Defining "simulation" is a notoriously difficult task. However, the type of simulation that the readers of Management Science have most often encountered has three features:

1) There are one or more stochastic input processes that are specified via the language of probability and from which synthetic realizations can be generated

2) There is a logical model that completely describes how the system of interest reacts to realizations of the stochastic inputs. The logical model is usually an algorithm that updates the System State upon the occurrence of some discrete events.

3) There are output processes, typically ordered by some concept of time, that represent the system behavior of interest to the analyst.

One of the proposed purposes for software process simulation is the management of software development risks, usually discussed within the category of project planning/management (Kellner et al. 1999) However, modeling and simulation primarily for the purpose of software development risk management has been quite limited. A notable exception is Madachy's (1994) model. It was designed partially for the purpose of risk assessment, which was achieved through two inputs that represented combinations of COCOMO drivers heuristically graded for degrees of risk. This paper describes another approach to simulation for managing software development risks, that of researching common and significant software development risk factors and their effects, then adapting a base model as necessary for simulating the selected factors. This simulator is a tool designed specifically for risk management.

Among the topics often discussed in the literature on software development risk management is the subject of risk factors. A variety of approaches have been used to investigate SDRFs. From them have emerged lists of risk factors, as well as taxonomies, questionnaires, and matrices for assessing software development risk. Some investigators have produced SDRF lists numbering on the order of 150 or more factors. A study of this literature (Houston 2000) revealed the existence of a set of common and significant SDRFs. Twenty-nine (29) of these were identified and selected for further study.

While a small but important body of work identifying SDRFs is accumulating, little work has been undertaken on the potential effects of these factors. Since knowledge of these effects is necessary for modeling and simulating SDRFs, a study was performed in two stages to identify and quantify the potential effects. In the first stage, software project managers participated in a qualitative survey, which utilized causal diagrams, and helped to identify significant potential effects of 29 risk factors. In the second stage, six of the 29 SDRFs were selected for further investigation (based on their importance and interrelationships) and a web-based quantitative survey was used to collect data on the actualized effects of the six factors. This data was used both to substantiate the results of the first survey and to develop quantitative relationships for the simulation model.

Although the data did not substantiate some of the effects suggested by the literature and by the first survey, analysis revealed other statistical relationships not previously discussed in the software development literature. The data analysis also provided probability distributions for the random variables representing risk factor effects.

A base model for the SDRFs was developed by combining features from two published software process simulators, Abdel-Hamid and Madnick (1991) and Tvedt (1996), then updating and extending the combination, particularly in the area of quality management.

The base model represents a waterfall process and incorporates sectors for Planned Staffing, Actual Staffing, Effort Allocation, Project Planning, Project Control, Productivity, Work Flow, and Quality Management.

The model was designed to support decision-making for three kinds of risk management activities: risk mitigation, risk contingency planning, and risk intervention. In typical usage for risk assessment, runs are made for different combinations of probable risks. This also establishes baselines for risk management studies. For risk mitigation study, setting inputs at the outset of a run may simulate various plans. For risk contingency planning, triggers have been added to the model so as to simulate a risk mitigation activity upon the occurrence of a predetermined condition. For risk intervention, outputs provide the user with simulated project states so that a "project" may be paused and inputs may be manipulated, thereby simulating an unplanned change for reducing the effects of an actualizing risk.

This simulator, with its ability to model development project outcomes stochastically, demonstrates that explicit modeling of risk factors and their potential effects provides an additional tool for software development risk management, particularly in the assessment and planning phases. Software project managers can use such a model to study the potential impact of various

combinations of risk factors on project outcomes, then run simulations for risk mitigation plans, risk contingency plans, and interventions, all as means of elucidating their experience and supporting project management decisions.

Research into SDRFs is ongoing; research into their potential effects has begun.Continued research in this area is expected to provide further insights into and data for the development of simulators for software project risk management. Modeling constructs for risk factors is also evolving. Experimentation with risk-oriented simulators may provide insights into the relative influence of various risk factors and into best practices for project risk management. Though some SDRFs are quite common, some appear to be specific to certain software development domains (Jones 1994), which suggests domain-specific simulation models for risk management.

C. Few Software Development Risk Factors Are

1) Unavailability of key staff
2) Reliance on a few key personnel
3) Instability and lack of continuity in project staffing
4) Lack of staff commitment, low morale
5) Low productivity
6) Lack of client support
7) Lack of user support
8) Lack of contact person's competence
9) Inaccurate metrics
10) Lack of organizational maturity
11) Lack of quantitative historical data
12) Inaccurate cost estimating
13) Excessive schedule pressure
14) Inadequate configuration control
15) Excessive reliance on a single development improvement
16) Excessive paperwork
17) Unreliable subproject delivery
18) Creeping user requirements
19) Unnecessary features
20) Immature technology
21) Complex application
22) Large number of complex external interfaces

Risk factors, provide a strong statistical tool for supporting risk management. The model can be designed to support decision-making for three kinds of risk management activities: risk mitigation, risk contingency planning, and risk intervention. In typical usage for risk assessment, runs are made for different combinations of
probable risks.

II. DISCUSSIONS AND CONCLUSIONS

So the study of SDRFs can be undertaken to identify common and significant risk factors for modeling in simulation. Qualitative and quantitative surveys can be used to study the factors and their potential effects. These survey results will provide the basis for new modeling constructs and statistical relationships to incorporate simulator. This simulator, with its ability to model development project outcomes stochastically will demonstrate that explicit modeling of risk factors and their potential effects will provide an additional tool for software development risk management. Software project managers can use such a model to study the potential impact of various combinations of risk factors on project outcomes.

REFEBRENCES

[1] Jones,Capers,[1998],"Minimizing the risks of software" Technical Report CMU/SEI may pp23-24.
[2] Kulik,Peter,[1996]"What is software Risk Management?"Technical reportCMU/SEI October pp34-35.
[3] Carr, Marvin J., Suresh L. Konda, Ira Monarch, F. Carol Ulrich, and Clay F. Walker. "Taxonomy-Based Risk Identification," CMU/SEI-93-TR-006. Pittsburgh: Software Engineering Institute, 1993

[4] C. Jones,[1994] “Assessment and Control of Software Risks”, Yourdon Press Prentice Hall, 1
[5] M. Lyu,[1995] “Software Reliability Engineering”, IEEE Computer Society Press.
[6] Schneidewind[1995],” Analysis of Error Processes in Computer Software”, Proceedings of the International Conference on Reliable Software,IEEE Computer Society, 21-23 April, p 337-346.
[7] J. Musa[1998], “Software Reliability Engineering: More Reliable Software,Faster Development and Testing”, McGraw-Hill.
[8] Software Engineering Institute, Software Risk Management, Technical Report CMU/SEI-96-TR-012, June 1996
[9] E. Hall[1997], Managing Risk, Methods for Software Systems Development, Addison Wesley.
[10] R. Charette, K. Adams, & M. White,[1996] “Managing Risk in Software Maintenance”, IEEE Software, May-June,.
[11] D. Karolak,[1996] “ Software Engineering Management”, IEEE Computer Society Press, 11. B. Boehm,[1981] “ Software Engineering Economics”, Prentice Hall.
[12] L. Putnam,[1980] “Software Cost Estimating and Life-cycle Control: Getting the Software Numbers”, IEEE Computer Society Press
[13] B. Londeix,[1987] “Cost Estimation for Software Development”, Addison-Wesley.
[14] Kellner, Marc I., Raymond J. Madachy, and David M. Raffo (1999) “Software process simulation modeling: Why? What? How?” Journal of Systems and Software 46: 91-105.