European Journal of Scientific Research
ISSN 1450-216X Vol.42 No.4 (2010), pp.629-636
© EuroJournals Publishing, Inc. 2010
http://www.eurojournals.com/ejsr.htm

Risk Assessment Techniques for Software Development

Rasmita Dash
Sr.Lecturer, Department of Computer Science, ITER, SOA University, Bhubaneswar
E-mail: rasmita02@yahoo.co.in

Rajashree Dash
Lecturer, Department of Computer Science, ITER, SOA University, Bhubaneswar
E-mail: rajashree_dash@yahoo.co.in

Abstract

Every software project is exposed to adverse external influences, the so called project risks that affect the cost, duration of the project and possibly the quality of the product but without risk, the project becomes lackluster and sometimes can't be completed to the fullest satisfaction. So with risk analysis, it can be determined for a specific project what the risks are. These risks then should be included in a systematic and formal manner in the project estimate in order to obtain a realistic project plan. This paper focuses on the problem of how to manage risk in the software development. We have discussed how Spiral model deals with the prevention and reduction of risks, continuously access all possible problems, define potential risks, and determine what risks are important and how to deal with them. Finally we have discussed some risk estimation method for software product development. Again we can conclude "the findings of this paper can be a project management tool to assess and tone down the events that might adversely affect a project, thereby increasing the possibility of success".

Keywords: Project management, Risk management, Spiral model, Risk estimation.

1. Introduction

Risk is defined as "Hazard, danger; exposure to mischance or peril". A simple definition of Risk is: "The possibility of loss, injury, disadvantage or destruction." Zardari et al (2009). Risk is concerned with uncertainty. This naturally includes uncertainty about the occurrence of known events, but also events that are not initially identified as impacting on the project. Risk management must therefore be an evolving and learning process, adapting to new and changing knowledge as the project precedes Hall et al (1998).

2. Risk Management

Risk Management consists of the processes, methodologies and tools that are used to deal with risks in the Software Development Life Cycle (SDLC) process of Software Project. Risk Management is defined as the activity that identifies a risk; assesses the risk and defines the policies or strategies to alleviate or lessen the risk. "Risk management is simply a practice of systematically deciding cost effective approaches for minimizing the outcome of threat realization to the organization.

3. Risks in Software Project Management

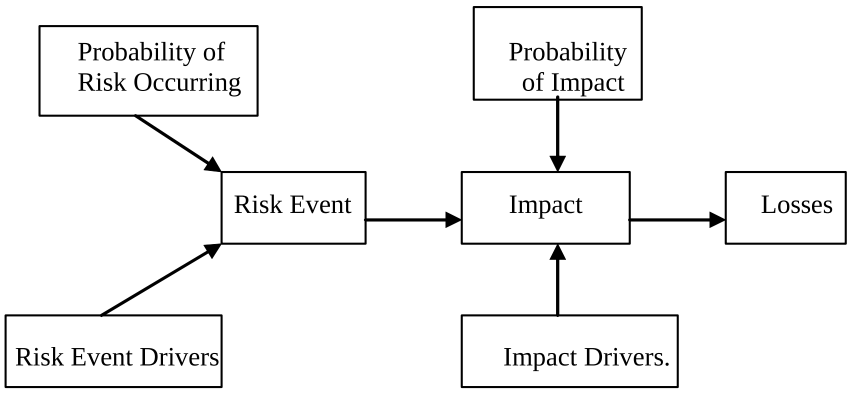
Unlike the hazards of daily living, the dangers in the young and rising field of software engineering must often be learned without the help of lifelong exposure. A more intentional approach is required. Such an approach includes studying the experiences of successful project managers and keeping up with the leading writers and thinkers in the field. The top software risk items can be

1) Personnel Shortfalls
2) Unrealistic schedules and budgets
3) Developing the wrong functions and properties
4) Developing the wrong user interface
5) Gold-plating
6) Continuing stream of requirements changes
7) Shortfalls in externally furnished components
8) Shortfalls in externally performed tasks
9) Real-time performance shortfalls
10) Straining computer-science capabilities

Proactive and Reactive Risk Management

Risk management focuses to assess the probability of risk occurring, risk event drivers, risk events, the probability of impact and the impact drivers before the risk actually takes place. This is in fact called 'proactive risk management'. Fig. 1 shows the flow of Proactive Risk Management

Figure 1: Proactive Risk Management



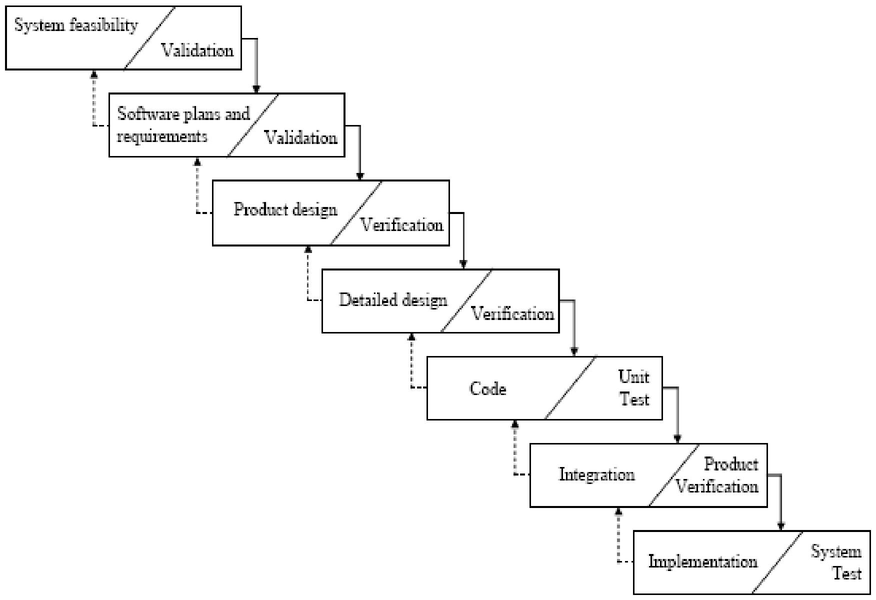
When a Project team responds to risks when they occur then it is called 'reactive risk management'. Failures are fixed; resources are found and applied once the risk strikes.

The risk assessment model, methods and techniques are widely used to control risk in a software development. Effective decision-making required a clearly defined risk assessment and analysis that could show any possible outcomes. The bad outcomes are risks and good outcomes are possibilities to produce good software. One of the challenges to accurately manage risk analysis is to use automated tools to store, organize and process data into meaningful knowledge Boehm et al (1989). Nowadays, software communities are striving to develop commercial software in order to stay viable. Many software projects when deployed displayed an excessive error and very low reliability. Yet, while software industry is actively using software risk management techniques to improve their risk management practices, only few reports on designing visualization tools are available for managing software risks.

4. Software Development using Waterfall Model

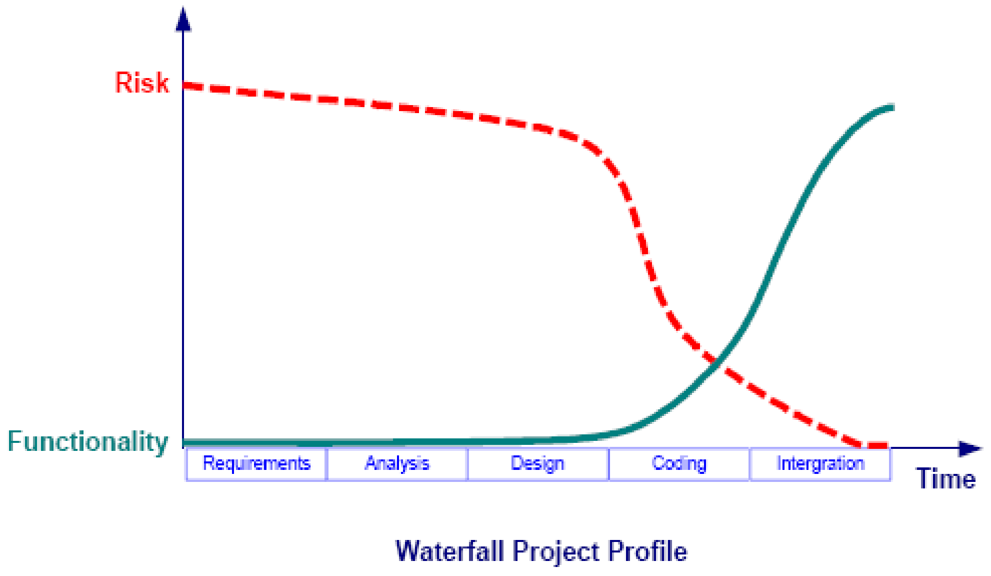
Royce introduced the traditional software development process model– "Waterfall Model" in year 1970. Later, Boehm expands this model in 1981 by adding additional steps. This waterfall model, illustrated in Figure2, allowed to steps back to the previous stages if necessary and has become the basis of most software acquisition standards in governments and industry Boehm et al (1988). The waterfall model has a clear objective where each process takes the input from the previous step.

Figure 2: Waterfall Model



According to Gillian et al (2004), the basic problem of waterfall model is that this model remains a high risk throughout the development lifecycle. This is because a waterfall process assumes that each stage can be fully defined without the need for feedback from the subsequent stages.

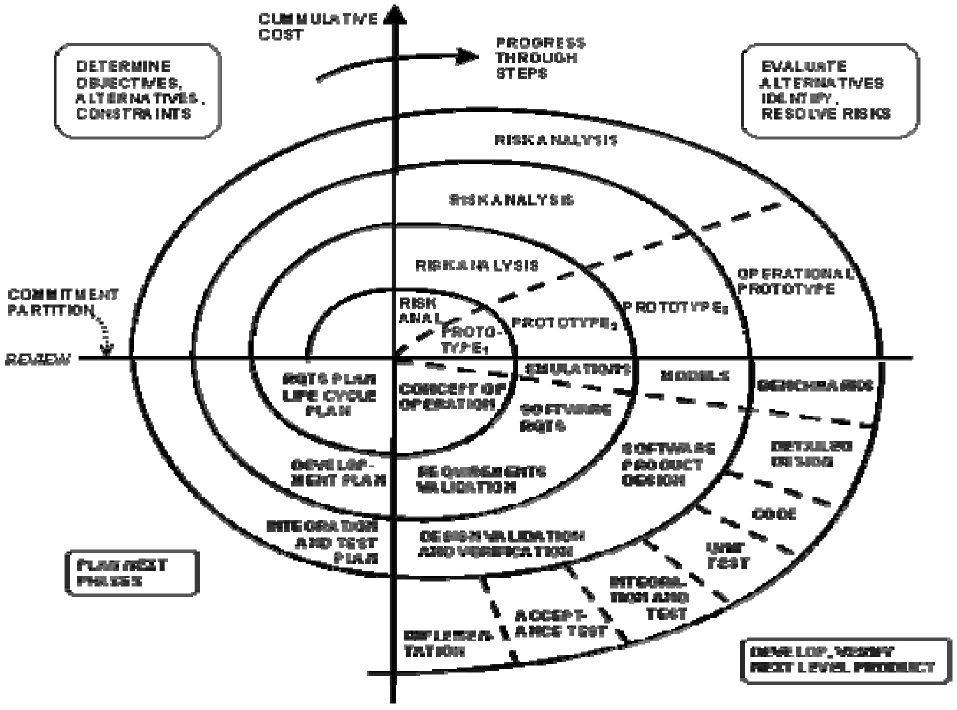
Figure 3: Illustrates a typical waterfall



The X-axis showed the time and Y-axis plots the functionality and the level of project risk. As shown in the graph above, the risk remains high until the last phase of the software development. The risk is gradually decreasing as the coding built, integrated and been testing. In order to produce good software, it is important to take risk into account. The basic problem is that in waterfall model the project risk remains higher throughout the software process development.

5. Spiral Model and Risk Management

Figure 4: Spiral model for software process



The spiral model presented in this article is one candidate for improving the software process model situation. The major distinguishing feature of the spiral model is that it creates a risk-driven approach to the software process rather than a primarily document-driven or code-driven process. It incorporates many of the strengths of other models and resolves many of their difficulties. The spiral model of the software process (see figure-4) has been evolving for several years, based on experience with various refinements of the waterfall model as applied to large government software projects. As will be discussed, the spiral model can accommodate different models (like waterfall model, evolutionary model, prototype model, transform model etc) as special cases and further provides guidance as to which combination of these model best fits a given software situation.

A typical cycle of the Spiral: Each cycle of the spiral begins with the identification of

- The objectives of the portion of the product being elaborated (performance, functionality, ability to accommodate change, etc).
- The alternative means of implementing this portion of the product (design A, design B, reuse, buy, etc); and
- The constraints imposed on the application of the alternatives (cost, schedule, interface, etc.)

The next step is to evaluate the alternative relative to the objectives and constraints. Frequently, this process will identify areas of uncertainty that are significant sources of project risk. If so, the next step should involve the formulation of a cost-effective strategy for resolving the sources of risk. Once the risks are evaluated, the next step is determined by the relative remaining risks. If performance of user-interface risks strongly dominates program development or internal interface-control risks, the next step may be an evolutionary development one: a minimal effort to specify the overall nature of the product, a plan for the next level of prototyping, and the development of a more detailed prototype to continue to resolve the major risk issues. On the other hand, if previous prototyping efforts have already resolved all of the performance or user-interface risks, and program development or interface-control risks dominate, the next step follows the basic waterfall approach modified as appropriate to incorporate incremental development.

The risk driven sub setting of the spiral model steps allows the model to accommodate any appropriate mixture of a specification oriented, prototype oriented simulation oriented, automatic transformation-oriented or other approach to software development. In such cases, considering the relative magnitude of the program risks and the relative effectiveness of the various techniques in resolving the risks chooses the appropriate mixed strategy. In a similar way, risk-management considerations can determine the amount of time and effort that should be devoted to such other project activities as planning, configuration management, quality assurance, formal verification, and testing. An important feature of the spiral model, as with most other models, is that each cycle is completed by a review involving the primary people or organizations concerned with the product.

Advantage

The primary advantage of the spiral model is that its range of options accommodates the good features of existing software process models, while its risk driven approach avoids many of their difficulties. In appropriate situations, the spiral model becomes equivalent to one of the existing process models. In other situations, it provides guidance on the best mix of existing approaches to a given project.

The primary conditions under which the spiral model becomes equivalent to other main process models are summarized as follows:

- If a project has a low risk in such areas as getting the wrong user interface or not meeting stringent performance requirements, and if it has a high risk in budget and schedule predictability and control, then these risk considerations drive the spiral model into equivalence to the waterfall model.
- If a software product’s requirements are very stable (implying a low risk of expensive design and code breakage due to requirements changes during development), and if the presence of errors in the software product constitutes a high risk to the mission it serves, then these risk considerations drive the spiral model to resemble the two-leg model of precise specification and formal deductive program development.
- If a project has a low risk in such areas as losing budget and schedule predictability and control, encountering large-system integration problems, or coping with information sclerosis, and if it has a high risk in such areas as getting the wrong user interface or user decision support requirements, then these risk considerations drive the spiral model into an equivalence to the evolutionary development model.
- If automated software generation capabilities are available, then the spiral model accommodates them either as options for rapid prototyping or for application of the transform model, depending on the risk considerations involved.

The spiral model has a number of additional advantages, summarized as follows:

1. It focuses early attention on options involving the reuse of existing software. The steps involving the identification and evaluation of alternatives encourage these options.
2. It accommodates preparation for life-cycle evolution, growth, and changes of the software product. The major sources of product change are included in the product’s objective, and information hiding approaches are attractive architectural design alternatives in that they reduce the risk of not being able to accommodate the product-charge objectives.

3. It provides a mechanism for incorporating software quality objectives into software product development. This mechanism derives from the emphasis on identifying all types of objectives and constraints during each round of the spiral.
4. It focuses on eliminating errors and unattractive alternatives early. The risk-analysis, validation, and commitment steps cover these considerations.
5. For each of the source of project activity and resource expenditure, it answers the key question, “How much is enough?” Stated another way, “Howe much of requirements analysis, planning, configuration management, quality assurance, testing, formal verification, etc. should a project do?” Using the risk-driven approach, one can see that the answer is not the same for all projects and that the level of risk incurred by not doing enough determines the appropriate level of effort.
6. It does not involve separate approaches for software development and software enhancement (or maintenance). It helps to avoid many of the problems that currently ensue when high-risk enhancement efforts are approached in the same way as routine maintenance efforts.

6. Approaches in Risk Estimation

When discussing project risks with project managers they are convinced that they take them into account. In the following we will discuss how project risks are currently taken into account.

6.1. The Simplistic Approach

In many cases, independent of the method with which a project manager calculated cost and duration of a project, the project risks are taken into account by applying a single factor to the cost and/or the duration. The factor ranges from 1.0to x depending on the gut feeling (sometimes called “experience”) of the project manager. Individual risks are not specified and not assessed and thus there is no way of determining the effect of a specific risk. One obtains two values: using factor 1means no risk will fire (a very optimistic approach), using a factor x meaning that all risks fire (a very pessimistic approach).

6.2. Function Point Analysis

In straightforward Function Point Analysis (FPA) one must assume that the effect of the various project risks, together with other influences on the project, are reflected in the fourteen “General System Characteristics (GSC)”

Table1: General system characteristics (GSC)

General System Characteristics	
Data communications	Online update
Distributed data processing	Complex processing
Performance	Installation ease
Heavily used configuration	Reusability
Transaction rate	Operational ease
Online data entry	Multiple sites
End user efficiency	Facilitate change

Each of these fourteen characteristics is assigned a degree of influence (DI) between 0(no influence) and 5 (strong influence). The overall contingency is expressed in the Value Adjustment Factor (VAF) which is calculated using the sum of all DIs.

$$VAF = 0.65 + 0.01 \sum_{n=1}^{14} DI_n$$

In case all characteristics have a DI of 5, the contingency is 35%, i.e. the number of unadjusted function points increases by 35% to obtain the adjusted function points.

Regardless of whether such a contingency is sufficient to take project characteristics (such as complexity) and project risks into account this approach does no tallow us to determine the effect of as specific risk such as “Change of technology” as there is no known relationship between a specific project risk and the general system characteristics. Neither the impact nor the probability of the risks may be taken into account in any way.

6.3. Cocomo II

COCOMO II takes project risks into account by defining a risk factor characterizing each module to be developed Boehm et al (2000). The total risk (TR) of each module is the sum of the risk levels (RL) of six types of risk (Table 2).

Table 2: Project risks

$$TR = \sum_{n=1}^{6} RI_n$$

Project risks
Schedule Risk
Product Risk
Platform Risk
Personnel Risk
Process Risk
Reuse Risk

The *risk factor* (R) of each module is determined by
$R = (TR/373)*100$

This approach allows us to consider types of project risks at a module level. A risk like “Change of technology”, however, would have to be allocated to more than one of these risk types, which would make it difficult to adjust its threat. Probabilities are not taken into account.

“Expert COCOMO”, an extension to COCOMO II, “aids in project planning by identifying, categorizing, quantifying, and prioritizing project risks”. It is a heuristic method that uses rules to analyze risks on the basis of cost factor information. This means that the project risks are not addressed the cost factors specified for each module. This makes it even more difficult to recognize the influence of a specific project risk and its probability of occurrence as we need a relationship between the project risk and its effect on the cost of a module.

6.4. Other Approaches

In a risk analysis Coombs et al (2003) determines three characteristics of the project risks: the probability of occurrence, the difficulty of detecting whether the risk has occurred and the impact on the project if the risk occurs. In addition, for each risk the maximum person-days and the maximum cost to correct the damage if the risk fires are determined.

Each characteristic is assigned a weight (W_c) recorded as Low, Medium, or High corresponding to the values 1, 3 and 7. The sum of these three weights is the Overall Weight.

$$OverallWeight = \sum_{c=1}^{3} W_c$$

The maximum value of the overall weight, the Overall Weight max, is 21, corresponding to a maximum contingency of 75%. The actual contingency is the proportion of the maximum contingency corresponding to the Overall Weight.

$$Contingency = Overallweight / Overallweight\,\max \times 75[\%]$$

The contingency then is applied to the maximum person-days and to the maximum cost of each risk to obtain the weighted maximum person-days and the weighted maximum cost in order to determine the project risk's effect on the project. With this approach any number of project risks may be considered. Each project risk is taken into account separately and independently. The probability is only used to calculate a contingency. There is no analysis of the probabilities in terms of determining the most probable increase of the project duration.

7. Conclusion

Thus one can conclude that formal risk management analysis and formal project assessment are effective and useful approaches that are starting to add rigor to the phrase "software engineering". Not every risk factor is fully controllable, and several risk factors exceed the authority of software managers. Nonetheless, risk analysis and assessment methods are quite effective in the identification of significant problems. Once problems are identified and examined, solutions can often be developed. We can say in conclusion that, like any other control, proper and timely Risk management control can provide enormous advantages to an organization by cutting down the cost and ensuring proper delivery as per schedule.

References

[1] Abdullah T, Mateen A, Sattar A. R, Mustafa T, "Risk Analysis of various Phases of Software Development Models", European Journal of Scientific Research,vol.40(3),pp.369-376,2010

[2] Boehm Barry W, "A Spiral Model of Software Development and Enhancement", IEEE Computer Society Press, vol.18 (3), pp.1-9, 1988.

[3] Boehm Barry W, "Software Risk Assessment", IEEE Computer Society Press, vol.15 (7), pp.902-916, 1989.

[4] Costaa R, Barros Marcio, "Evolutionary Software Project Portfolio Risks", Journal Systems and Software, vol.80 (1), pp.16-31, 2007.

[5] Gang Xie, Zhan Jin," Risk avoidance in bidding fop Software Projects based on the Cycle Management Theory", International Journal of Project Mangement,vol.12(4),pp.516-521,2006

[6] Geoffrey G.Roy, "A Risk Management Framework for Software Engineering Practice", Australian Software Engineering Conference, pp.60, 2004.

[7] Gillian Adens, "The Role of Risk in a Modern Software Development Process", TASSC Technical Paper, 2008.

[8] Zardari.S,"Software Risk Management", International Conference on Information Management & Engineering IEEE Computer Society, pp.375-379, 2009.

[9] Hall, Elaine M, "Managing Risk – Methods for Software Systems Development", Addison Wesley, 1998.

[10] Boehm B, Abts C, Brown A. W, Chulani S, Clark B. K, Horowitz E, Madachy R, Reifer D. J and Steece B, "Software Cost Estimation with Cocomo II". Prentice-Hall, Inc., 2000.

[11] Coombs P. "IT Project Estimation". Cambridge University Press, 2003.