Software Risk Evaluation and Assessment using Hybrid Approach

Sneh Prabha1, R.L.Ujjawal2

Student M.Tech(IT) GGSIPU,Delhi1
Assistant Professor,GGSIPU2

ABSTRACT
The complexity and the associated risk increases as the size of the project increases. In this paper we have proposed a technique to evaluate the risk based on the source code as well as on the changes in the requirements of the user. Because it is not possible to test exhaustively each and every path in the code so some of the faults are left which can become the future risks. The risk assessment is based on the code is calculated by considering the conditions, variable and predicates in the code. Because there are always some changes in the project, major or minor. These changes increase the chances of the risk. So this proposed model considers the impact of changes on the risk.

Keywords
Software risk, Risk exposure, Risk management, Risk Assessment.

I. INTRODUCTION
Identification and management of the risk is one of the most important tasks of software development. As the software size, complexity, content and changes associated with the project increases the risk associated with the project also increases. Software projects are the high risk activities. It will be very helpful if we can identify all the potential risk at the beginning of the project .This will be helpful in minimizing the occurrence and impact of the various risks. The process of risk management includes set of the activities including identification, analysis, planning, tracking, controlling the risk. Risk is a probability of occurrence of some unwanted and harmful event to the project. These events can result in delay, over budget, wrong functionality or termination of the project, degradation in product functionality & quality and high maintainability and reusability cost. Different techniques have been proposed to handle various type of risk. Some of these are used in the various phases of the development while others are applicable on the code itself. Risk management partly means reducing uncertainty [1].

There are three dimensions of software risk. (i) Technical Risk (ii) Organizational Risk (iii) Environmental Risk. The technical dimension is results of uncertainty in the task and procedure. The organizational dimension is related to poor communication and organizational structure. Because of the change in environment and external factors the environmental risk occurs. All these factors affect the software in different ways [2]. Risk management uses the various steps for managing the risk. It includes the risk identification, risk assessment, risk mitigation, avoidance, acceptance and transference. Risk avoidance is to avoid the risk before it occurs to prevent and control the damage caused by the risk. Risk transference is done by transferring the responsibility of risk handling to some third party. Risk transference can be done by outsourcing, contracts, warranties and insurance. Risk acceptance is to accept the presence of risk in the software. Further steps can be taken based on the severity of the risk. Risk identification includes the process of identifying the risk associated. Risk assessment is used for rating the various risks and the probability and impact of the risk. Risk mitigation is used to prepare a plan for handling and minimizing the adverse effect of the risk. It can be done by using controlling, avoiding or transferring the risk. Risk mitigation consist of the various activities including planning risk control measures, implementing risk control measures, monitoring the risk, controlling the risk, learning on risk.

II. BACKGROUND AND RELATED WORK
Various models have been developed for various types of projects for the risk management based on their different needs and conditions. Software risk management is not a onetime activity and it is a continuous process that has to be followed throughout the life of a project. Approach for the risk management can be based on the traditional approach or it can be based on the proactive such that each project is studied individually to find out the related risk and their management. In traditional approach focus is on the risks which are common in all the projects. These are easy to identify and control. SEI has defined six paradigms of risk management. These are identification, analyzing, planning, tracking, controlling, and communicating the risk. Details of these is best describes in [3]. SEI's software risk management is supported by three groups Software risk evaluation, Continuous risk management and team risk management. The objective of risk management strategies is to prevent, mitigate, correct and ensure system failure. The goal of this model was to identify and resolve risk in early stage and to develop risk strategies to handle all these risk. There is another model called Model of risk assessment of Software Project based on grey theory defines grey comprehensive evaluation model of the risk management. It combines AHP and entropy method to confirm the weight of risk index and calculates the grey approach degree by using improved grey correlation degree as decision making unit. Then software risk can be ranked according to grey approach degree [4]. Another model for risk assessment and evaluation is for the projects based on the fuzzy analytical hierarchal process. For this related risk factors are identified and then expert qualitative judgments about these factors are acquired. These judgments are translated into fuzzy numbers and used as a input to FAHP. After this risk factors are ranked and prioritized by FAHP in order to make project managers aware of important risk and to enable them to adopt measures to deal with these highly devastating risks. It suggested the risk identification. FAHP establishes the hierarchical structure and creates the fuzzy judgment matrix using pair-wise comparisons [5]. Various approaches are used for the risk management. They can be based on the process model, checklist, analytical framework or risk response strategies. In checklist method a list of the risks is

prepared and it is verified for the project. . A checklist is a referential list of typical risk factors of project failure compiled from the experiences of some past projects. Checklists usually have the form of a questionnaire or a risk list. A questionnaire consists of a set of questions that ask for the current state of the project. Some questionnaires use only yes-no questions,. It is a fast, easy and low cost process for risk identification. But decision about which list to use can be very tough. Qualitative risk estimation is used to estimate the risk exposure in two dimensions by considering the likelihood of occurrence, loss due to the risk. It determines the impact or severity of consequences).Analytical framework is closely related to the checklist. It is non-process based analytical framework provides an alternative way to think and manage software risks. Various some include closed test questions and some other employ open questions check list can be some standard list or it can be project specific framework have been proposed based on the source of the risk, lifestyle based risk or based on the elements and their relationships. Process models specify the stepwise task for managing the risk. Process model specify the activities to manage the risk. Boehm's and PMI's PMBOK Guide are example of process models. There is another model called Software Risk Assessment Model (SRAM). This model considers the nine critical risk elements (i) complexity of the software; (ii) staff involved in the projects (iii) targeted reliability (iv) product requirement (v) method of estimation (vi) method of monitoring (vii) development process adopted (viii) Usability of software (ix) tools [6].

III. PROPOSED RISK ESTIMATION STRATEGY

Although so many models have been proposed but no model is perfect for different circumstances. Each model has its strong point and its weakness also and each is defined for a particular category of projects. Risk estimation can be based on the code of the project or it can be model based. Code based approach is useful for calculating risk associated with the code. It can be based on various errors.

In the proposed approach we measure the total phase-wise risk then it is added to the risk calculated from the source code. So if there is any change in the code after the completion we can calculate the risk based on code as well as on the changes that has been implemented. This model will be more useful and better as we are considering the impact of the changes. So the risk calculation will consider the risk due to code as well as the risk due to changes in the code at each phase of development.

These are the errors affecting the risk measurement in the project. Risk measurement can be done in different ways. Risk measurement can be done by considering the various risk factors and it can also be based on the code. The earlier is beneficial as it can measure the risk before the coding is done and it will be very helpful in risk management activities. Risk can also be measured by using the function point.

Each method has its own benefits and drawbacks. In this paper we are going to study risk measurement model which takes the benefits of both code based risk analysis and risk factors based risk analysis. It will be more suitable to the projects which are used over a long period of time. These projects require regular maintenance and updates so the need to keep the record of various risks is a very important activity. It uses actual values of parameters and variables. So the result will be more accurate. These errors can be categorized as follow

1) Measurement Error. The measurement error occurs if some of the input variable in model has inherent accuracy limitation. For example, as a result of Chris Kemerer's work, 3 function points are assumed to be at least 12 percent inaccurate. Thus, if you estimate a product size of 1,000 function points, measurement error could mean that the real size is anywhere between 880 and 1,120 function points. So applying a model of 0.2 person- days per function point means your estimate will have a range of uncertainty between 176 and 224 person- days, with a most likely value of 200 person-days [7].

2) Model Error There are many numbers of factors that affect the effort for the software project. But all of these cannot be included in the model. Model errors occur when all the factors that affect the effort required to produce the project are not included in the model. Some values in the projects are based on the data taken from past projects. For Example if we get the 0.3 person per day function point from past project same value may not be applicable to some another project. So if we are using some past project data then we should calculate the associated inaccuracy by using, for example, the mean magnitude relative error [7]. Thus if estimation model is with an inherent 20 percent inaccuracy and your product is 1,000 function points in size, estimation will be between 140 and 260 person-days. Measurement inaccuracy and model inaccuracy are additive.

3) Scope Error. The scope error occurs when the when project is outside your estimating model's domain. The estimation model will be suitable for the same domain. It will not be applicable to the other domain. If model is used for some another domain then results will not be accurate. It will be difficult to quantify the impact of the
scope error. If your estimation models or methods are completely out of scope, you cannot produce a meaningful effort estimate. In such circumstances any estimate should not be done. We can choose some another technique to check feasibility and risk exposure.

4) Assumption Error. Assumption errors occurs when we make incorrect assumptions about a model's input parameters. For example, if we assess the product size of 1300 function point by assuming that we have correctly identified all the customer requirements. If all assumptions are identified then we can investigate the affect of being invalid by assessing both the probability that an assumption is incorrect and the resulting impact on the estimate. This is the form of risk analysis. For example you believe that there is a 0.3 probability that the requirement complexity has been underestimated and, if it has, you estimate another 100 function point. At this point the concept of risk exposure is used to calculate the effective current cost of a risk and can be used to prioritize risk that requires countermeasure [8]. Risk exposure and model error are independent. Identifying the impact of a wrong assumption does not increase or decrease the estimate uncertainty due to model or measurement error. Assumption errors are very helpful in estimating the risk exposure. Effective current cost of a risk can be calculated by using the risk exposure and it can be used to prioritize risk that requires countermeasure. Risk exposure can be calculated by multiplying the probability with the total loss due to the risk. These three errors are used to estimate the risk exposure. The risk can be calculated as there is change in the requirements like addition of requirements, modification of requirements, or deletion of the requirements. So total risk can be computed as

(Risk) changes = [b/a] i= 1 to n +K [A [c/b] i +B [d/b]i +G [e/b] i] (1)

Where [b/a] i= (Number of mission critical requirements)/ (Total number of requirements) at the input of phase number i. Ki is the penalty for adding, modifying or deleting of requirements during phase number i [10]. Here A, B, C are penalty for adding, modifying, deleting the mission critical requirements. Penalty for adding will be more as compared to the penalty for modification and deletion. This is because addition of a requirement will require more efforts. Also the c, d, e are number of mission critical requirements added, modified during the phase i and b is the total number of mission critical requirements. This risk is added to the risk calculated from the source code. This result will give the total risk after the each change in the code.

The Source code-based software risk assessing model [9] is a static and dynamic model in which the risk estimation is based on the code of the project. The static structure is based on the metric related to the static structure of the code. It considers number of c-uses (condition-uses), p-uses (predicate-uses), definitions, decisions and function calls whereas the dynamic model uses additional dynamic test coverage of the code such as decision, c-use and p-use coverage to calibrate the metric values used in the model. it can be based on the product of the selected metrics. Risk increases with the complexity of the code. As code becomes Users can choose all or some of these metric components with appropriate weighting factors for the construction. Modeling scheme can be either based on the summation of the selected metrics or more and more complex the associated risk also increases. With the rigorous testing of code the chances of occurrence of risk decreases. Thus while constructing the risk model these are important factors. If the summation scheme are selected the corresponding static risk model can be expressed as

(Risk) code=V * □ □+ F * □ □+ D * □ □+ C * □ □+ P * □□ □ □□

Where all five metric components are number of variable definitions (V), number of function calls (F), number of decisions (D), number of c-uses (C) and number of p-uses (P)) Here □, □, □, □, and □ □are the weighting factors. These factors are used to give either more or less emphasis to the metric components in computing the risk. . Weights are given special values given by experts who have very detailed understanding of the system being analyzed. If such information is not available, a possible choice is to use a weighting factor of 1 for all components.

We can now calculate the total risk by adding the risk due to the source code and risk due to changes in the requirements. The total risk will include the risk measured from the source code and the risk measured from the changes in the mission critical requirements.

Total risk = (Risk) changes + (Risk) code

So this total risk defines the final risk associated. Whenever there is a change in the code and the change in the requirements we can use this model. So this approach is applicable to all the phases of the software as well as to the final code produced. It will be applicable at every stage of the development.

IV. CONCLUSION

In this paper we have proposed a new approach for risk measurement. This approach considers the impact of changes after the project is complete. Risk measurement is based on the risk measurement based on the source code as well as on the requirement changes in the project. Because of the changes in the source code the parameters will change as well as the associated risk also changes. So we calculate the risk based on these two factors of code and change in requirements.

V. REFERENCES

[1] Roger L. Van Scoy, "Software Development Risk: Opportunity, not problem", Technical report, September 1992.

[2] Mohd. Sadiq, Mohd. Wazih Ahmad, Md. Khalid Imam Rahmani, Sher Jung Software Risk Assessment and Evaluation Process (SRAEP) using Model Based Approach, IEEE 2010 International Conference on Networking and Information Technology, pp171-177.

[3] Ray C. Williams, George J. Pandelios, and Sandra G. Behrens, "Software Risk Evaluation (SRE) method description (Version-2.o), Technical report December-I 999.

[4] Pang Qinghua ,"A Model of Risk Assessment of Software Project Based on Grey Theory",Proceedings of 2009 4th International Conference on Computer Science and Education IEEE ,pp 538-541.

[5] H. Iranmanesh , S. Nazari Shirkouhi ,M. R. Skandari ,"Risk evaluation of Information Technology Projects Based on Fuzzy Analytical Hierarchical Process",World Academy of Science Engineering and Technology 40 2008.

[6] Say-Wei Foo , Armugam Muruganatham, "Software Risk assessment Model", ICMIT 2000, IEEE, pp-536-544.

[7] 7.BARBARA KITCHENHAMand STEPHEN LINKMAN, Estimates,Uncertainty, and Risk IEEE SOFTWARE 1997,pp 69-74.

[8] Daya Gupta, Mohd. Sadiq , Software Risk Assessment and Estimation Model , International Conference on Computer Science and Information Technology 2008 IEEE 2008,pp 963-967.

[9] W. Eric Wong, Yu Qi, and Kendra Cooper,Source Code-Based Software Risk Assessing, 2005 ACM Symposium on Applied Computing, pp 1485-1490.

[10] Mohd. Sadiq, Abdul Rahman, Shabbir Ahmad Mohammad Asim, Javed Ahmad, esrcTool: A Tool to Estimate the Software Risk and Cost, Second International Conference on Computer Research and Development, IEEE 2010, pp 886-890.