International Journal of Advancement in Engineering Technology, Management & Applied Science

Volume1, Issue 1 June 2014 ISSN No: 2349-3224

Title
Role of Software Metrices in identifying the risk of project

MD. Imran Alam

Sr. Lecturer, Jazan University, Saudi Arabia

Abstract:

Process metrics can be used to improve software development and maintenance. Product metrics describe the product characteristics such as design, complexity, quality level, performance, size and design features. Project metrics describe the project characteristics and execution. This paper defines the role of Metrices in identifying the risk for successful project.

Keywords: Risk, Metrices, Mitigation

Introduction:

Software quality process is associated more closely with process and product metrics than project metrics. Software Quality metrics can be further subdivided into end product quality metrics and in process quality metrics. The essence of software quality is to investigate the relationships among in process metrics, project characteristics and end product quality based on the finding to engineer improvements in both process and product quality. Software quality metrics are the subset of the software metrics that focus on the quality aspects of the product, process and project.

Classification of Software Risk:

The risk management process is a continuous process to address the risk systematically throughout the project/product life-cycle. Risks can be introduced in the very earlier stages of the lifecycle of software. The ability to identify the earlier risks translates into earlier removal of risk at less cost, which promotes high success probability of project. Software process risk includes both technical and management work procedures. In management procedures, process risk is found in activities such as staffing, tracking, configuration management, planning and quality assurance. Software product risk contains final and intermediate characteristics of work product. Technical responsibility of the product risk is found in test specifications, design performance, requirements stability and code complexity. Product risk is critical to manage

because software requirements are perceived often as flexible. The below figure shows the classifications of software risk:

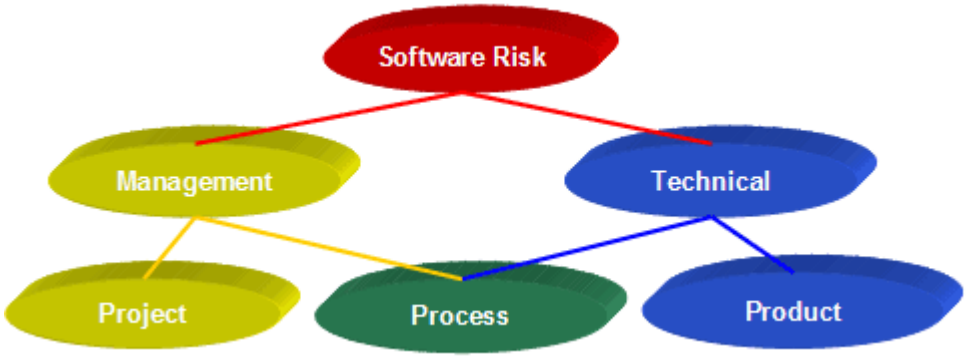

Figure 1: Classifications of software risk

Process of Risk Management:

The process of risk management consists of the following activities, which supports the maintenance, operation, development, acquisition and supply of software services and products. The steps in the risk management are:

- Implementation/Planning risk management process.
- Managing the risk profile of project.
- Performing risk monitoring.
- Performing risk analysis.
- Performing risk treatment and
- Evaluating the process of risk management.

The below figures shows the steps in the process of risk management:

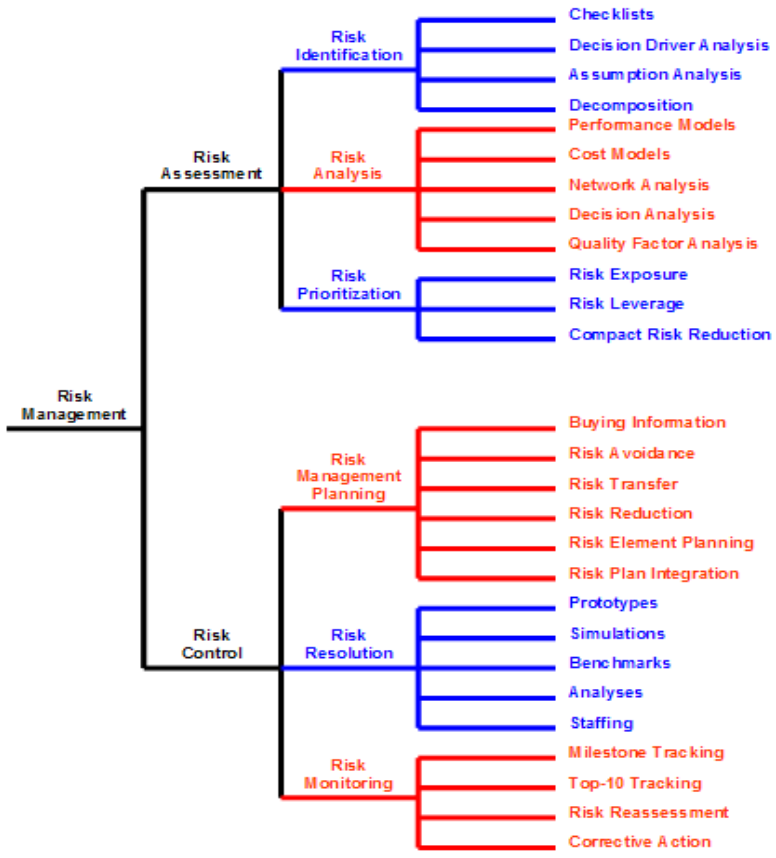

Figure 2: Process of Risk Management

Objectives of RiskManagement:

The objective of Risk Analysis is to identify potential problems that could affect the cost or outcome of the project.The objective of risk assessment is to take control over the potential problems before the problems control you, and remember: “prevention is always better than the cure”.

The following figure shows the activities involved in risk analysis. Each activity will be further discussed below.

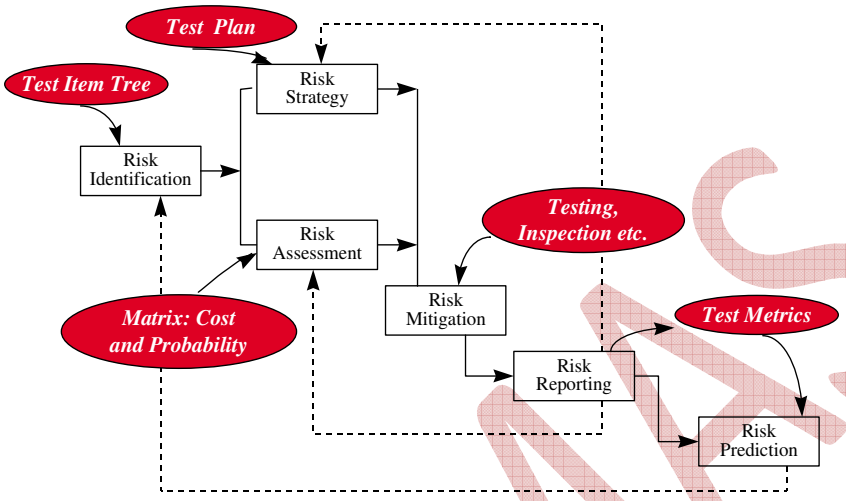

Figure 3:Risk analysis activity model. This model is taken from Karolak's book "Software Engineering Risk Management",

Risk Identification:

Risk identification involves collecting information about the project and classifying it to determine the amount of potential risk in the test phase and in production (in the future).The risk could be related to system complexity (i.e. embedded systems or distributed systems), new technology or methodology involved that could cause problems, limited business knowledge or poor design and code quality.

Risk Mitigation:

The activity of mitigating and avoiding risks is based on information gained from the previous activities of identifying, planning, and assessing risks. Risk mitigation/avoidance activities avoid risks or minimise their impact. The idea is to use inspection and/or focus testing on the critical functions to minimise the impact a failure in this function will have in production.

Risk Reporting:

Risk reporting is based on information obtained from the previous topics (those of identifying and mitigating risks).Risk reporting is very often done in a standard graph like the following:

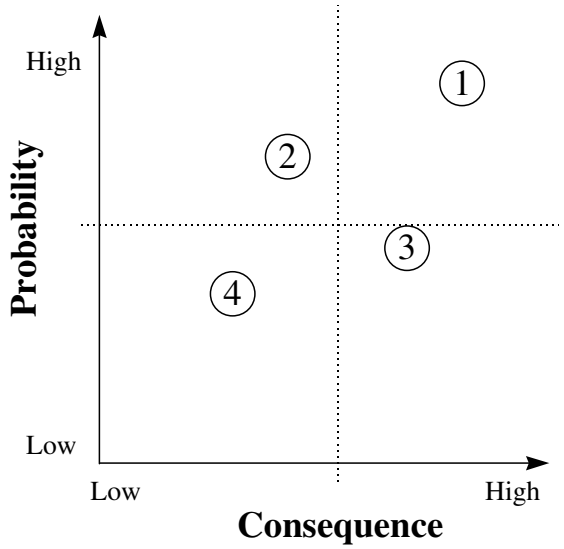

Figure 4: Standard risk reporting -

In the test phase it is important to monitor the number of errors found, number of errors per function, classification of errors, number of hours testing per error, number of hours in fixing per errors etc.

Role of Metrices:

There are several reasons to use metrics, for instance:

- Return on investment (cost / benefit analyses)
- Evaluate choices, compare alternatives, monitor improvement
- Have early warning of problems, make predictions
- Benchmark against a standard or in competition

Metrices for Progress tracking:

Metrics used for measuring progress:

1. the number of tests planned, executed and completed
2. the number of faults per function
3. the number of hours used in testing per fault found
4. the number of hours used in fixing per fault (to correct the error and return the function to re-test)

The metrics were reported graphically and trend analysis applied. For instance the information about "test planned, executed and planned" was compared with information about "faults to be

fixed and actually fixed". The reason was to have an early warning of a resource problem if the number of not completed tests increased at the same time as the number of faults to be fixed were increasing.

Based on the information above, it was possible to calculate "Estimated to Complete" in number of hours, i.e. resource requirements to complete the test project. This was of course very important information in a project based on reducing risk and dynamic resource allocation to the most critical areas.

Metrics to predict probability of faults

A completely different type of metric is used to identify probability of faults in the system. Identifying indicators that were expected to be of importance per function did this. Indicators could be "Changed functionality since previous release", size of function (i.e. number of lines of code), complexity (this could be functional complexity or structural complexity), quality of design documentation etc. A number of 1, 2 or 3 (i.e. low, medium or high) was given to each indicator per function as well as a weight to handle different importance between the indicators.
Now a probability of having a fault could be calculated per function and compared to the other functions in that system. This information should then be combined with information about the consequence of a fault in each function.
Based on this information it will now be possible to "rank" the list of functions based on risk exposure (probability and cost of a fault).

Conclusion:

In this paper we defined the risk involved in project and how metrices identifying this risk.
Finally we conclude that metrices plays a major role in reducing risk and dynamic resource allocation in critical areas and also predict probability of faults.

References:

1. Boehm B W, Software Engineering Economics, Prentice-Hall, 1981.
2. Conte S D, Software Engineering Metrics and Models, 1986.
3. Cushman M A, Software Measurement, Analysis and Control, 1992.
4. Daskalantonakis M K, Practical View of Software Measurement, 1992.
5. Demarcate T, Controlling Software Projects: Yourdon Press, 1982.
6. Fenton N E, Software Metrics, Thomson Computer Press, 1997.
7. Grady R B, Software Metrics: Establishing a Company-Wide Program, Prentice-Hall, 1986.
8. Jones C, Assessment and Control of Software Risks, Yourdon Press, 1994.
9. John S, Applied Software Measurement, McGraw-Hill, 1997.
10. Symons C R, Software Assessments and Best Practices, Addison-Wesley, 2000.
11. Littlewoods B, Risks of Software, 1992.
12. Oman P, Applying Software Metrics, Computer Society Press, 1997.
13. Boris Beizer, "Software Testing Techniques" Second Edition, Van Nostrand Reinhold, 1990.
14. Dale Walter Karolak, "Software Engineering Risk Management", IEEE Computer Society Press, 1996.