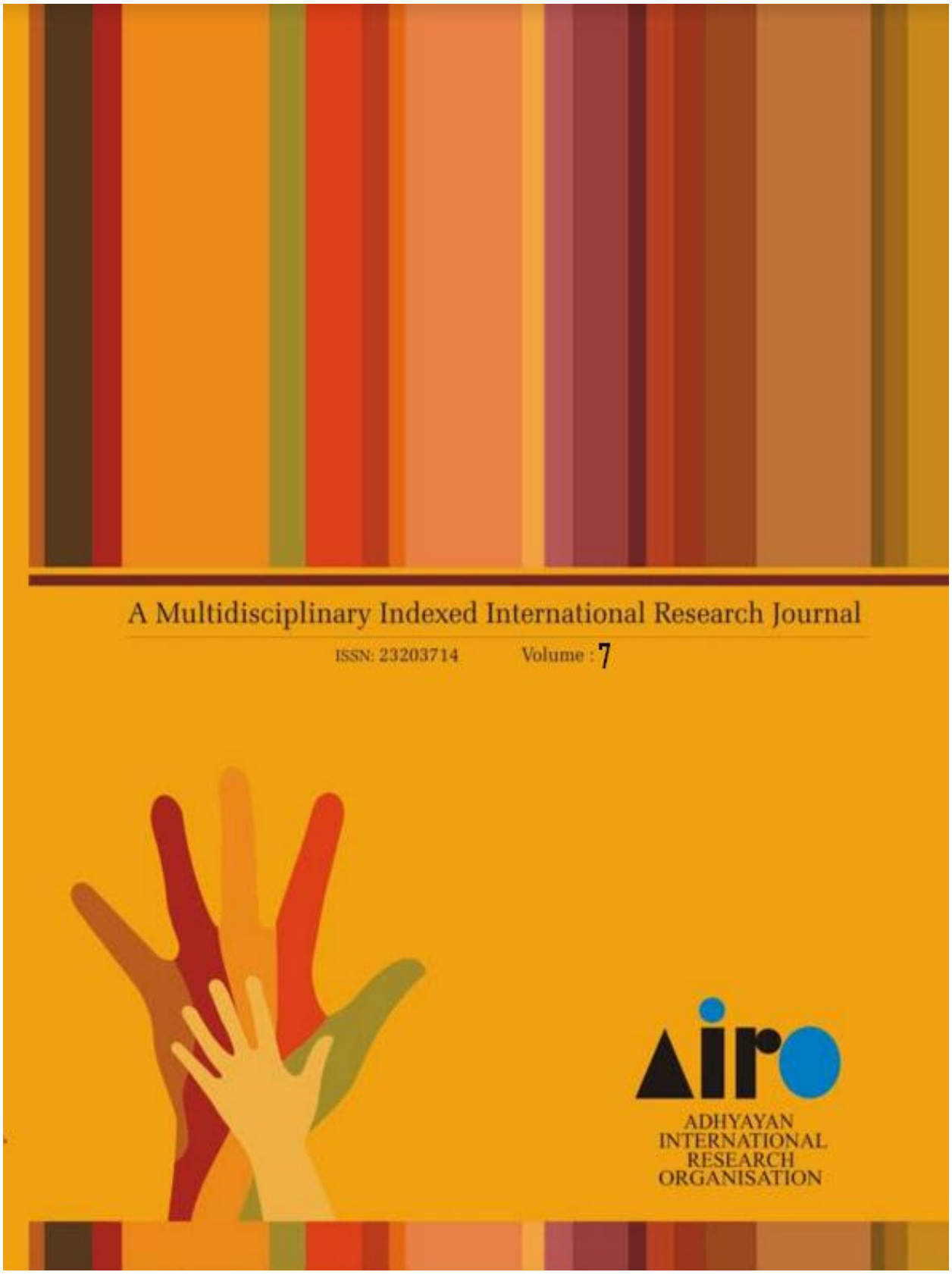

A Multidisciplinary Indexed International Research Journal
ISSN: 23203714 Volume : 7
Airo
ADHYAYAN
INTERNATIONAL
RESEARCH
ORGANISATION

SIGNIFICANCE OF RISK MANAGEMENT IN SOFTWARE DEVELOPMENT LIFE CYCLE

G. RAJESH

Research Scholar, Satya sai University

Abstract

The proactive management of risks throughout the software development lifecycle is important for project success. A risk is a potential future harm that may arise from some present action such as a schedule slip or a cost overrun. The loss is often considered in terms of direct financial loss, but also can be a loss in terms of credibility, future business, and loss of property or life. Risk management is a series of steps whose objectives are to identify, address, and eliminate software risk items before they become either threats to successful software operation or a major source of expensive rework.

Keywords:

Risk, Project, Software

Introduction

The software industry is fraught with failed and delayed projects, most of which far exceed their original budget. The Standish Group reported that only 28 percent of software projects are completed on time and on budget. Over 23 percent of software projects are cancelled before they ever get completed, and 49 percent of projects cost 145 percent of their original estimates. (Standish, 1995) In hindsight, many of these companies indicated that their problems could have been avoided or strongly reduced if there had been an explicit early warning of the high-risk elements of the project. Many projects fail either because simple problems were reported too late or because the wrong problem was addressed. (Bruegge and Dutoit, 2000).

The risk management process can be broken down into two interrelated phases, risk assessment and risk control, as outlined in Figure 1. These phases are further broken

down. Risk assessment involves risk identification, risk analysis, and risk prioritization. Risk control involves risk planning, risk mitigation, and risk monitoring.(Boehm, 1989) Each of these will be discussed in this section. It is essential that risk management be done iteratively, throughout the project, as a part of the team's project management routine

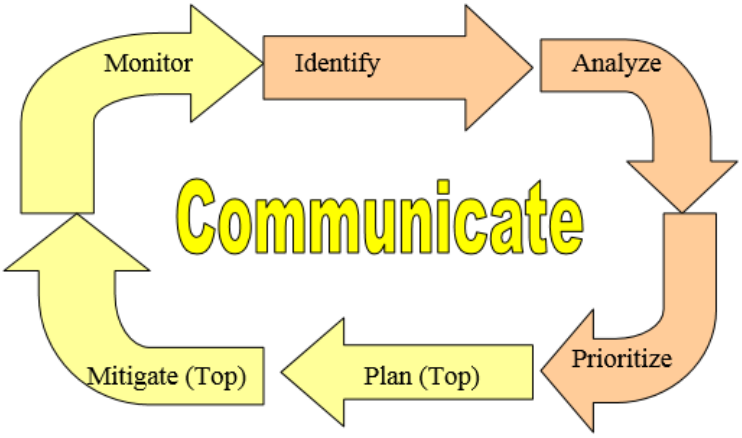


Figure 1: The Risk Management Cycle.

Generic risks are potential threats to every software project. Some examples of generic risks are changing requirements, losing key personnel, or bankruptcy of the software company or of the customer. It is advisable for a development organization to keep a checklist of these types of risks. Teams can then assess the extent to which these risks are a factor for their project based upon the known set of programmers, managers, customers, and policies. Product-specific risks can be distinguished from generic risks because they can only be identified by those with a clear understanding of the technology, the people, and the environment of the specific product. An example of a product-specific risk is the availability of a complex network necessary for testing.

Generic and product-specific risks can be further divided into project, product, and business risks. Project risks are those that affect the project schedule or the resources (personnel or budgets) dedicated to the project. Product risks are those that affect the quality or performance of the software being developed. Finally, business risks are those that threaten the viability of the software, such as building an excellent product no one wants or building a product that no longer fits into the overall business strategy of the company.

Project participants can be reluctant to communicate potential failures or shortcomings and can be too optimistic about the future. It is essential that all participants are encouraged to report risks so they can be monitored and managed. Participants should be rewarded for identifying risks and problems as early as possible.

It is recommended that risks should be stated using the condition-transition-consequence (CTC) format (Gluch, 1994):

Given that <condition> then there is a concern that (possibly) <transition> <consequence>.

• Condition is a description of the current conditions prompting concern. • Transition is the part that involves change (time). • Consequence is a description of the potential outcome. For example, given that no one in our team has ever developed a product in Prolog, then there is a concern that (possibly) the project will take two months longer than has been estimated.

Perceived risk can be reduced by obtaining more information through investigation. For example, in a project in which the use of a new technology has created risk, the team can invest some money to learn about the technology. Throw-away prototypes can be developed using the new technology to educate some of the staff on the new technology and to assess the fit of the new technology for the product.

• Contingency plans. A contingency plan is a plan that describes what to do if certain risks materialize. By planning ahead with such a plan, you are prepared and have a strategy in place do deal with the issue.

• Risk reduction. For example, if the team is concerned that the use of a new programming language may cause a schedule delay, the budget might contain a line item entitled “potential schedule” to cover a potential schedule slip. Because the budget already covers the potential slip, the financial risk to the organization is reduced. Alternately, the team can plan to employ inspections to reduce the risk of quality problems.

• Risk acceptance. Sometimes the organization consciously chooses to live with the consequences of the risk (Hall, 1998) and the results of the potential loss. In this case, no action is planned.

Risk planning and risk mitigation actions often come with an associated cost. The team must do a cost/benefit analysis to

decide whether the benefits accrued by the risk management steps outweigh the costs associated with implementing them. This calculation can involve the calculation of risk leverage (Pfleeger, 1998).

Risk Leverage = (risk exposure before reduction – risk exposure after reduction)/cost of risk reduction

If risk leverage value, rl, is ≤ 1, clearly the benefit of applying risk reduction is not worth its cost. If rl is only slightly > 1, still the benefit is very questionable, because these computations are based on probabilistic estimates and not on actual data. Therefore, rl is usually multiplied by a risk discount factor $\rho < 1$. If ρ rl > 1, then the benefit of applying risk reduction is considered worth its cost. If the discounted leveraged valued is not high enough to justify the action, the team should look for other, less costly or more effective, reduction techniques.

After risks are identified, analyzed, and prioritized, and actions are established, it is essential that the team regularly monitor the progress of the product and the resolution of the risk items, taking corrective action when necessary. This monitoring can be done as part of the team project management activities or via explicit risk management activities. Often teams regularly monitor their "Top 10 risks."

Risks need to be revisited at regular intervals for the team to reevaluate each risk to determine when new circumstances caused its probability and/or impact to change. At each interval, some risks may be added to the list and others taken away. Risks need to be reprioritized to see which are moved "above the line" and need to have action plans and which move "below the line" and no longer need action plans. A key to successful risk management is that proactive actions are owned by individuals and are monitored. (Larman, 2004)

As time passes and more is learned about the project, the information gained over time may alter the risk profile considerably. Additionally, time may make it possible to refine the risk into a set of more detailed risks. These refined risks may be easier to mitigate, monitor, and manage.

On-going and effective communication between management, the development team, marketing, and customer representatives about project risks is essential for effective risk management. This communication enables the sharing of all

information and is the cornerstone of effective risk management.

The three stakeholders are involved in risk management.

• The developer must systematically and continually enumerate all the possible risks related to technical capability and making the schedule.

• The manager must lead the team to follow the risk management process to proactively manage the project risks. The manager must also allocate resources for proactive risks management.

• The customer must participate in the continual identification of risks.

None of these stakeholders is empowered to manage business risks, i.e. what we called organizational and managerial risks, and sales and support risks in the "Risk Identification " section above. This kind of risk must be managed by upper management and marketing department of the firm.

Research Study

After risks have been identified and enumerated, the next step is risk analysis. Through risk analysis, we transform the risks that were identified into decision-making information. In turn, each risk is considered and a judgment made about the probability and the seriousness of the risk. For each risk, the team must do the following:

• Assess the probability of a loss occurring. Some risks are very likely to occur. Others are very unlikely. Establish and utilize a scale that reflects the perceived likelihood of a risk. Depending upon the degree of detail desired and/or possible, the scale can be numeric, based on a percentage scale, such as "10 percent likely to lose a key team member" or based on categories, such as: very improbable, improbable, probable, or frequent. In the case that a categorical assignment is used, the team should establish a set numerical probability for each qualitative value (e.g. very improbable= 10 percent, improbable = 25 percent). • Assess the impact of the loss if the loss were to occur. Delineate the consequences of the risk, and estimate the impact of the risk on the project and the product. Similar to the probability discussion above, the team can choose to assign numerical monetary values to the magnitude of loss, such as $10,000 for a two-week delay in schedule. Alternately, categories may be used and assigned values, such as 1=negligible, 2=marginal, 3=critical, or 4=catastrophic.

Determining the probability and the magnitude of the risk can be difficult and can seem to be arbitrarily chosen. One means of determining the risk probability is for each team member to estimate each of these values individually. Then, the input of individual team members is collected in a round robin fashion and reported to the group. Sometimes the collection and reporting is done anonymously. Team members debate the logic behind the submitted estimates. The individuals then re-estimate and iterate on the estimate until assessment of risk probability and impact begins to converge. This means of converging on the probability and estimate is called the Delphi Technique (Gupta and Clarke, 1996).

The Delphi Technique is a group consensus method that is often used when the factors under consideration are subjective.

If numerical values were given for probability (percentage) and impact (monetary), the risk exposure can be calculated. Risk exposure is calculated as follows (Boehm, 1989):

Risk Exposure (RE) = P × C

where P = probability of occurrence for a risk and C is the impact of the loss to the product should the risk occur. For example, if the probability of a risk is 10 percent and the impact of the risk is $10,000, the risk exposure = (0.1)($10,000) = $1,000. If RE is calculated for each risk, the prioritization is based upon a numerical ranking of the risk exposures.

Risk Management for Software Development Model Selection

In this section, we explain a risk-driven approach to making the selection between an agile, a plan-driven, or a hybrid software development model. The five-step method was developed by Barry Boehm and Richard Turner (Boehm and Turner, 2003; Boehm and Turner, June 2003). Boehm and Turner developed the method so that software developers can enjoy the benefits of both agile and plan-driven methods, while mitigating many of their drawbacks. The guidance given by their method is important because every development practice has its situation-dependent shortcomings and its home ground (the situations for which each is best suited). Agile methodologies promise increased customer satisfaction, lower defect rates, faster development times, and a solution to rapidly changing requirements. Agile methods are highly iterative in nature – meaning that partial working product is

delivered to customers often. Iteration is a prudent risk mitigation strategy because the partial deliverables uncover risks while there is still time to alleviate them. Plan-driven approaches promise predictability, stability, and high assurance. It's all about picking the right model for the job depending upon the most important consideration of the project.

Personal Characteristics of Team

Some background is necessary before describing Boehm and Turner's method. To start, Boehm and Turner believe the personal characteristics of the people who make up the software development team are a key factor in determining whether to use an agile or plan-driven approach. Think about it. A team made up of very experienced team members is very different from a team that consists of all new people to the technology and the domain. The technology is the programming language, hardware platform, and so forth. The domain is the subject area of the program (for example, medical software or networking software). To classify individual skill level, Boehm and Turner adopted and then adapted the classification scheme of Alistair Cockburn (Cockburn, 2001).

It is important to consider both technology and domain expertise when considering a person's skill level. A Level 3 expert in an object-oriented language such as Java developing software for the retail industry might temporarily revert back to being a Level 1B if moved to an assignment like developing a compiler in a functional language such as Haskell. This person's prior expertise enables him to fairly rapidly advance through the skill levels, most likely to the old Level 3. However, it is important to consider the person's current (not potential) skill level when considering the make up of the team relative to agile and plan-driven methods.

Agile and Plan-Driven Home Grounds

Boehm and Turner have observed projects succeed that have used purely an agile approach, they have observed projects succeed with purely plan-driven methods, and they have observed projects succeed with hybrid methods. Based on these experiences, they share the project characteristics of agile "home grounds" and plan-driven "home grounds" where home ground is defined as the situation for which each is best suited.

Step One: Risk Analysis Three different areas of risk are analyzed: environmental, agile, and plan-driven. Each of these areas is now defined. • Environmental risks – risks that result from the project's general environment, as discussed in the earlier sections of this chapter and enumerated in Table 1. • Agile risks – risks that are specific to the use of agile methods. Some of these are issues related to the ability of agile methods to scale to larger teams and projects and to handle the reliability needs of critical projects. Additionally, there are agile risks associated with not thoroughly documenting prior to coding, with the potential of personnel turnover/churn, and with having enough skilled people. • Plan-driven risks – risks that are specific to the use of plan-driven methods. Some of these issue relate to the ability of plan-driven methods to handle rapid technology and/or requirements change, the need to deliver rapid results, and/or having enough team members skilled in plan-driven methods.

If not enough information is known about any of these risks, some resources can be spent to obtain some information about the project's aspects until the team feels more confident about the project risks.

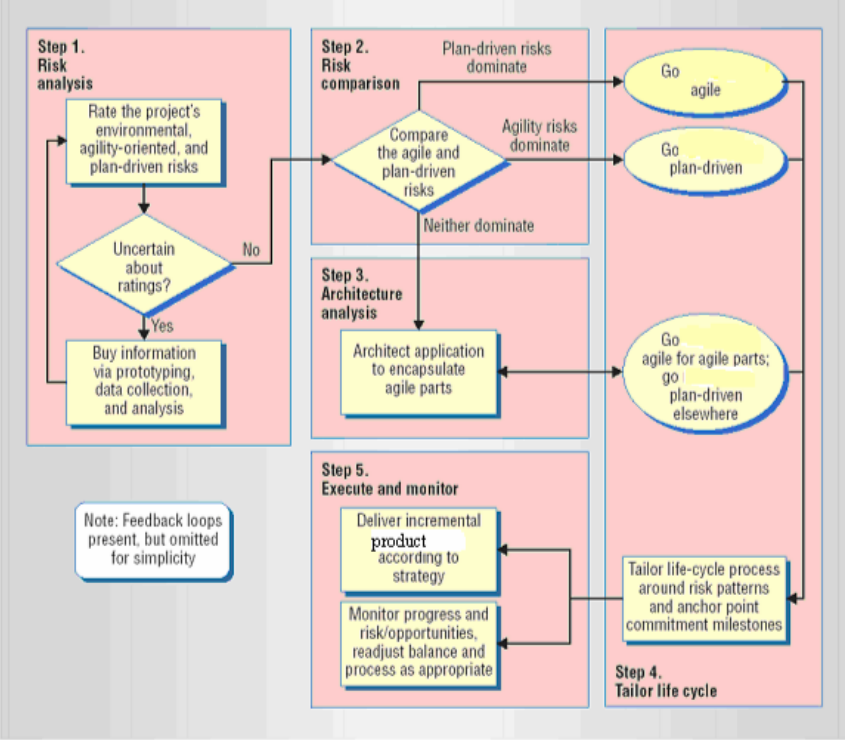
Step Two: Risk Comparison After the risks are identified, the team assesses and compares them. If the plan-driven risks outweigh the agile risks (meaning the issues related to using a plan-driven methodology are more concerning), then the team should adopt an agile method and proceed to Step Four. If the agile risks outweigh the plan-driven risks, then the team should adopt a plan-driven method and proceed to Step Four. If neither dominates – and the project characteristics do not clearly lie in the agile or plan-driven home ground – then the team should proceed to Step Three.

Step Three: Architecture Analysis The optional Step Three is done when the project characteristics do not clearly lie in either the agile or plan-driven home ground or when parts of the system lie in an agile home ground and other parts of the system lie in the plan-driven home ground. If possible, the team develops a system architecture so that the team is able to use agile methods on the parts of the system where their strengths can be best applied. The remainder of the system is developed via plan-driven methods.

Step Four: Tailor Life Cycle A project strategy is developed to address the risks identified in Step One, as was discussed

earlier in the chapter. The life-cycle process is tailored around the identified risk patterns.

Step Five: Execute and Monitor Consistent with the need to consistently monitor risk items, as discussed earlier in the chapter – the team must consistently reassess the risks related to agile and plan-driven methods. If the risk profile changes, the team should consider their choice of process model.



Conclusion

In the risk management cycle, product and project risks are identified, analyzed, and prioritized. The top-ranking risks are planned and mitigated. All risks are monitored. It is important for a project to focus on its critical success factors while keeping an eye on its risk factors. Risk management practices enable the team to find the opportunity in the risk items.

References

1. Boehm, B. (1989). Software Risk Management. Washington, DC, IEEE Computer Society Press.

2. Boehm, B. (January 1991). "Software Risk Management: Principles and Practices." IEEE Software: 32-41.

3. Boehm, B. and R. Turner (2003). Balancing Agility and Discipline: A Guide for the Perplexed. Boston, MA, Addison Wesley.

4. Boehm, B. and R. Turner (June 2003). "Using Risk to Balance Agile and Plan-Driven Methods." IEEE Computer 36(6): 57-66.

5. Bruegge, B. and A. H. Dutoit (2000). Object-Oriented Software Engineering: Conquering Complex and Changing Systems. Upper Saddle River, NJ, Prentice Hall.

6. Cockburn, A. (2001). Agile Software Development. Reading, Massachusetts, Addison Wesley Longman. Gluch, D. P., "A Construct for Describing Software Development Risks," Software Engineering Institute, Pittsburgh, PA CMU/SEI-94-TR-14. Gupta, U. G. and R. E.

7. Clarke (1996). "Theory and Applications of the Delphi Technique: A bibliography (1975-1994)." Technological Forecasting and Social Change 53: 185-211.

8. Hall, E. M. (1998). Managing Risk: Methods for Software Systems Development, Addison Wesley.

9. Larman, C. (2004). Agile and Iterative Development: A Manager's Guide. Boston, Addison Wesley.

10. Pfleeger, S. L. (1998). Software Engineering: Theory and Practice. Upper Saddle River, NJ, Prentice Hall.

11. Standish (1995). "The Chaos Report." Van Scoy, R. L., "Software Development Risk: Opportunity, Not Problem," Software Engineering Institute, Pittsburgh, PA CMU/SEI-92-TR-030.