A STUDY OF SOFTWARE DEVELOPMENT PROJECT RISK MANAGEMENT

Ye Tao
Management School
Hangzhou DianziUniversity, Hangzhou, ZheJiang, 310018, China
taoye3790@163.com

Abstract

Most software development projects confront great risks and risks might occur in the whole development process. Success of the projects demands effective management of the risks. Five main reasons are analyzed for a complete and clear understanding of the risks. The reasons include the use of new and unproven technologies, system requirements, system architecture, system performance, and organizational or non-functional affairs. The risks can be avoided, confined, mitigated and monitored under effective management. Software development project risk management should focus on reduction and prevention of risks, continuously assess possible problems, define potential risks, determine what risks are important and deal with them. Four steps and three countermeasures are put forward to cope with various risks in software development projects. The steps are risk identification, assessing the level of exposure for each risk, risk mitigation and risk conclusion. The countermeasures are careful, typical and flexible countermeasures.

1. Introduction

Most software development projects confront great risks and risks might occur in the whole development process. Many projects try to advance current software capabilities and achieve something that has not been done before while the opportunity for advancement cannot be achieved without taking risks [1]. So the risks must be lessened for the success of a project, especially considering that current software market often demands updated software developed as quickly as possible.

2. RISKS IN SOFTWARE DEVELOPMENT PROJECTS

Risk can be described as project variables that designate project success. It defines the probability that a software development project experiences unwanted and inadmissible events, such as termination, delays in schedule and overrun of resources. Risk corresponds to possibility of poor quality of the software solution, increased costs, failure, or delayed completion. In most software development projects risks occur due to five main reasons: the use of new and unproven technologies, system requirements, system architecture, system performance, and organizational or non-functional affairs.

Most software development projects apply new technologies. An improper use of new technologies usually leads to project failure, in which the main reason comes from knowledge. In order to implement new technologies, a project team should have sufficient relative knowledge. But either the importance of this knowledge is often ignored or the team members lack of enough knowledge.

Software requirements of users guide the entire development process. It is very hard to obtain the right software solution that absolutely meets the users' expectations [2]. To obtain a satisfactory solution, a project team must find out the whole requirements. Identifying and defining the requirements is long and complicated, and the requirements tend to change during the development. This change can be dangerous because the least change may lead to the failure of the whole project.

Software architecture, defined in the early development stage, is a set of key decisions about the organization and components of a software solution. The architecture possibly does not satisfy all the requirements on solution, and this can be verified only with a software prototype realized in later development stages. So the risks appear.

Software should have satisfactory performance for the users' non-functional requirements. The performance is tested only on a real and realized solution, which makes it necessary to predict the performance in the early stages [1]. But these predictions are very hard to make precisely, which implies the risks.

Risks due to organizational or non-functional affairs are related to project resources and schedule. Organizational problems may affect a software solution since only an efficient organization ensures a successful project [3]. A defined schedule can be a risk because of many unwanted events that may cause a delay in the

978-0-7695-3480-0/08 $25.00 © 2008 IEEE
DOI 10.1109/FITME.2008.125

solution. It is a management problem to define a schedule to satisfy both customers and developers.

Besides identification and specification of risks, risks in software development must be addressed and managed. There are many choices to deal with the risks. They can be avoided, confined, mitigated and monitored. The best way is completely avoiding them. Many risks can be avoided by the use of different technologies, changing requirements or project plans. The second way is confining risks that cannot be avoided. A risk can be confined so that its impact area affects only a small part of a software development project. In order to reduce the impact area, the risk must be identified and the project should be changed regarding the technology and resources used. For risks that cannot be avoided or confined, the proper way is mitigating or monitoring them. Some of the risks can be mitigated by the realization of a software prototype in order to try the risks and perceive if they will materialize or not. It is better that the risks materialize on a software prototype than on a real software solution because a software development team can learn on a software prototype and discover a way to avoid or confine materialized risks. If risks cannot be mitigated, they should be constantly monitored in order to track their materialization. For the risks, contingency plans are needed to define activities to be taken if they materialize. The risks that cannot be mitigated introduce a serious threat to a software development project, so they should be considered as highly important. After the materialization of such risks, a complete project assessment and decision about the project future should be made [2]. Risks can be identified and addressed in different phases of a software development project, but it is essential to identify risks as early as possible and address them promptly because the cost connected with exposed risks might be enormous.

3. RISK MANAGEMENT STEPS

Software development project risk management should focus on reduction and prevention of risks, continuously assess possible problems, define potential risks, determine what risks are important and deal with them. So a whole project picture is required for successful risk management. Four basic steps are needed.

The first step is risk identification to discover all factors that could lead to project failure. These factors are connected with the technology used in the software development process and organizational areas. They are observed and assessed in order to discover all potential risks. It is necessary to capture details of the discovered risk, like risk description, probability of risk occurrence, costs connected with materialized risk and possible risk solutions and avoidance countermeasures.

The second step is assessing the level of exposure for each risk. Risks are ranked in levels according to their impact and the order for being solved is determined. Risks with a fatal impact are ranked higher than risks with a gentle impact. This is necessary because the former risks are considered in the early stages, while the cost of risk materialization and project failure is smaller than in later stages [4].

The next step is risk mitigation. This step attempts to avoid or prevent risk occurrence. Risks can be mitigated by avoidance, reduction and transfer. Avoidance is the best and needs completely reorganizing the development, but sometimes it is very difficult and even impossible to do. Reduction means re-planning a project to reduce the probability of risk occurrence. Transfer is forwarding risk to where it would cause less damage.

The final step is risk conclusion. This step is after the definition of a mitigation plan and includes all the actions for avoidance and reduction. The actions in the conclusion should be defined in the contingency plan that describes actions taken once a risk becomes a reality.

4. COUNTERMEASURES FOR EFFECTIVE RISK MANAGEMENT

Three countermeasures can be used for effective risk management in software development projects according to the amount of risk.

The first is careful countermeasure. It fits for fresh and inexperienced organizations whose software development projects are connected with new and unproven technology. This countermeasure may be described as high priority risk management that requires rigorous risk analysis on multiple levels. Risks should be analyzed and traced on individual, team and organizational levels [5]. To identify important risks as soon as possible and on a wide problem area, every project team member takes part in the risk management. In the implementation of the careful countermeasure, risk management roles are defined to choose some project members whose primary activities are with risk management. Since careful countermeasure deals with an extremely high number of risks, an organization formally defines risk interpretation and ranking policy to help separating the important risks from the unimportant ones in order and address the former quickly. Project team members continuously trace

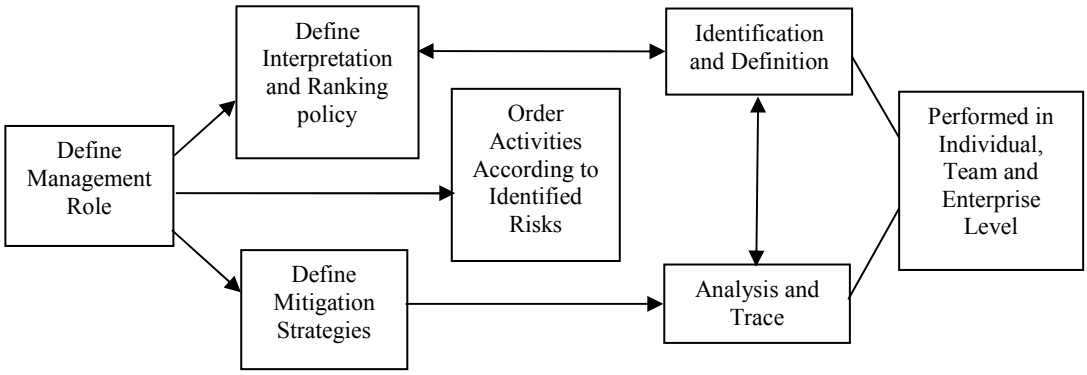

Figure 1 Activities of careful countermeasure

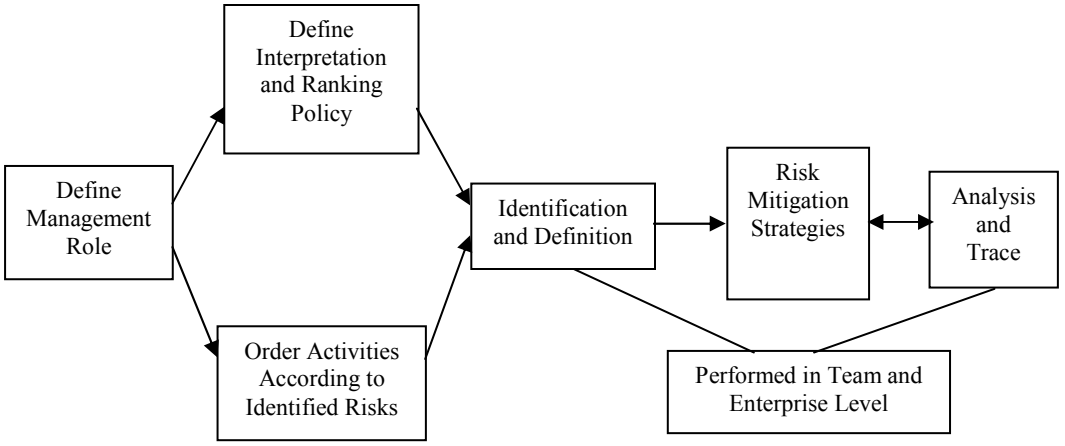

Figure 2 Activities of typical countermeasure

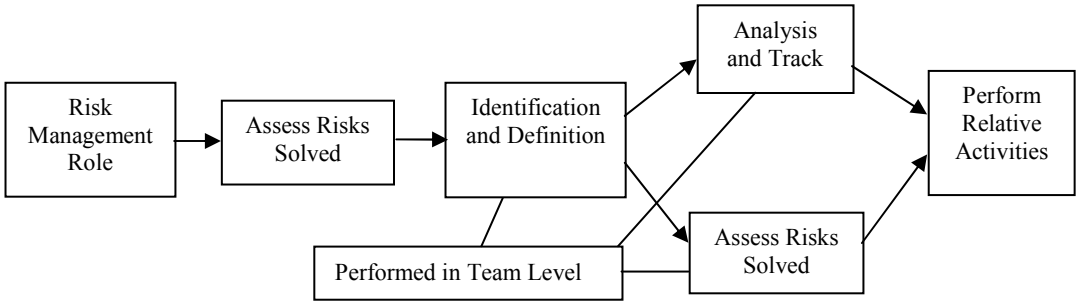

Figure 3 Activities of flexible countermeasure

risks and actions about risks. The development order is according to project risks so that risks with high importance are solved first. Activities of the careful riskmanagement countermeasure are presented in Figure 1.

The second is typical countermeasure. It fits for mature and experienced software development organizations. The level of acceptable risks for this countermeasure is medium and risks are identified in steps according to project progress. The typical countermeasure assumes that most risks can be easily addressed. Risk management is often practiced in the team level and organizational level because risks on a well-defined project are connected with the entire project. This countermeasure does not require project roles for risk management because risks are identified by the project management team and sometimes by team members [6]. Risks also guide software development in order to early address important risks, but because only a relatively small amount of risks matter, project plans are stable and only occasionally changed. Risk interpretation and ranking policy is less formal since risks are defined and ranked by the team. Activities of the typical risk management countermeasure are presented in Figure 2.

The third is flexible countermeasure. It fits for a mature organization with formally defined software development projects and demands the use of known and proven technology with efficient organization. This countermeasure assumes a small number of acceptable risks. Flexible countermeasure requires a relatively informal risk definition. Risk mitigation and contingency plans are defined only for a few important risks, and the definition is to minimize the work for risk management [7]. This countermeasure is based on comparing currently identified risks to previously encountered ones and risk management is practiced mostly in the organizational level. There is no risk interpretation and ranking policy and risks are identified at the beginning of the project and occasionally during the projects. Initial project plans are made according to identified risks, but final risk materialization does not change project plans. Activities of the flexible risk management countermeasure are presented in Figure 3.

5. Conclusions

Risks exist in every software development project and the success of a project is closely connected with effective risk management. In most projects risks occur due to: the use of new and unproven technologies, system requirements, system architecture, system performance, and organizational or non-functional affairs.

Four basic steps are needed to manage risk in software development project, including risk identification, assessing the level of exposure for each risk, risk mitigation, risk conclusion. Three countermeasures (careful, typical and flexible) can be used for effective risk management in software development project according to the amount of risk.

References

[1] Kwak Y H, Stoddardb J (2004). “Project risk management: lessons learned from software development environment”. *Technovation*, Vol. 24, No.11, pp.915-920.

[2] Dan X.Houston, Gerald Tod (2001). “Stochastic simulation of risk factor potential effects for software development risk management”. *Journal of Systems and Software*, Vol. 59, No.3, pp.247-258.

[3] Morcio Oliveira Barros (2004). “Supporting risks in software project management”. *Journal of Systems and Software*, Vol. 70, No.1, pp.21-35.

[4] Gang Xie, Jin Zhan (2006). “Risk avoidance in bidding for software projects based on life cycle management theory”. *International Journal of Project Management*, Vol. 12, No.4, pp.516-521.

[5] John S. Osmundson, James B. Michael (2003). “Quality management metrics for software development”. *Information & Management*, Vol. 40, No.8, pp.799-812.

[6] Wen-Ming Hana, Sun-Jen Huang (2001). “An empirical analysis of risk components and performance on software projects”. *Journal of Systems and Software*, Vol. 74, No.1, pp.42-50

[7] R. Costaa, Marcio Barros (2007). “Evaluating software project portfolio risks”. *Journal of Systems and Software*, Vol. 80, No.1, pp.16-31.