STRATEGIES FOR SUCCESSFUL SOFTWARE DEVELOPMENT RISK MANAGEMENT

Marija Boban[*]*, Željka Požgaj*[**]*, Hrvoje Sertić*[***]

Received: 16. 05. 2003 Preliminary communication
Accepted: 02. 11. 2003 UDC: 004.42 : 65.012

Nowadays, software is becoming a major part of enterprise business. Software development is activity connected with advanced technology and high level of knowledge. Risks on software development projects must be successfully mitigated to produce successful software systems. Lack of a defined approach to risk management is one of the common causes for project failures. To improve project chances for success, this work investigates common risk impact areas to perceive a foundation that can be used to define a common approach to software risk management. Based on typical risk impact areas on software development projects, we propose three risk management strategies suitable for a broad area of enterprises and software development projects with different amounts of connected risks. Proposed strategies define activities that should be performed for successful risk management, the one that will enable software development projects to perceive risks as soon as possible and to solve problems connected with risk materialization. We also propose a risk-based approach to software development planning and risk management as attempts to address and retire the highest impact risks as early as possible in the development process. Proposed strategies should improve risk management on software development projects and help create a successful software solution.

1. INTRODUCTION

Nowadays, software is becoming a major part of enterprise business. The significance of software is growing everyday, along with the progress of

[*] Marija Boban, MSc, University of Split, Faculty of Law, Domovinskog rata 8, 21000 Split Croatia, Phone: +385 98 955 0805, Fax: +385 21 393597, E-mail: marija.boban@pravst.hr
[**] Željka Požgaj, PhD, University of Zagreb, Faculty of Economics, J.F. Kennedy Square 6, 10000 Zagreb, Croatia, Phone: +385 1 238 32 77, Fax: +385 1 233 56 33, E-mail: pozgaj@efzg.hr
[***] Hrvoje Sertić, MSc, Research & Development Center, Ericsson Nikola Tesla, Krapinska 45, 10000 Zagreb, Croatia, Phone: +385 98 416 777, Fax: +385 1 365 361, E-mail: hrvoje.sertic@etk.ericsson.se

technology. Software development is activity connected with advanced technology and high level of knowledge. Every software development project faces a significant amount of uncertainty that is usually manifested as possible risk materialization The success of a software development project is directly connected with the involved risk, i.e. project risks should be successfully mitigated in order to finish a software development project. The conditions on today's global software market demand the most advanced software solutions from enterprises in order to be comparable and competitive. Development of an advanced software solution in the shortest possible time is a process associated with an extremely high number of risk impacts.

Every aspect of a software development project could be influenced by risks that could cause project failure. It is common to say that risk is the price of opportunity, i.e. a project with a high number of risks has an opportunity on the global software market if the project is completed on time and within planned expenses. Many development projects are trying to advance current software capabilities and achieve something that has not been done before. The opportunity for advancement cannot be achieved without taking risks. The use of advanced and, in most cases, unproven technology on software development projects leads to a large number of risks. In order to complete a complex software development project within planned boundaries, risks on the project should be well understood and mitigated.

Definition of an approach to risk management in the form of strategies suitable for different software development projects is the subject of this work. With this work, we will show that risk management should be a part of every software development project and that only successful risk mitigation leads to successful software development. Furthermore, we will show that a well-implemented risk management process, defined with the risk management strategies we propose, increases an enterprise's opportunities on the global software market. Today's common software development processes, such as Unified Process (Booch, Rambaugh & Jacobson, 2001), define risk management as activity on software development projects, but they do not define strategies suitable for risk management implementation on different development projects.

This paper is organized as follows: After this introduction, an overview of the risks in software development is given. The third section deals with the risk management on software development projects. The fourth section presents strategies for risk management proposed in this work. Conclusions are given at the end.

2. RISKS IN SOFTWARE DEVELOPMENT

As a result of the conditions on the global software market, the software industry is becoming more ruthless everyday. In recent years, software solutions have become extremely complex, and the complexity of software solutions is growing everyday (Sertić, 2002). The global market requires software with new features and capabilities developed in small time frames. Conditions on the global software market demand new levels of usability, quality, reliability, and performance from software solutions. The only chance for enterprises to survive on such a market is to build software that is better than the competition, in the shortest possible time. Building quality software in the terms defined by the conditions from the global software market is fundamentally hard and connected with a high level of risk (Hall, 1998). Risks are present in every aspect of software development. Software development is a special kind of engineering activity because of the involved knowledge and technology, so it is common to say that risks have an extremely high influence on the success of software development projects (Jones, 1994). There are numerous kinds of risks that can cause software development project failure.

A risk can be defined as a consideration that has some degree of probability of compromising the success of a software development project (Karolak, 1995). Formally, risk can be described as project variables that designate project success (Capers, 1994). Risk defines the probability that the software development project will experience unwanted and inadmissible events, such as termination, delays in project schedule and overrun of project resources. Risk is the possibility of suffering loss that describes the impact on the project which could be in the form of poor quality of the software solution, increased costs, failure, or delayed completion. Software development is specific because risks in software development can be classified in many categories. It is difficult to clearly define risk categories that could apply to a wide area of software development projects. The essence of risk classification is not in precisely defining risk categories, but in identifying and describing as many risks as possible on the project. Due to that, it is suggested that risks on software development projects should be classified according to the main impact areas (Ould, 1996).

For most software development projects, we can define five main risk impact areas: (1) The use of new and unproven technologies, (2) software system requirements, (3) software system architecture, (4) software system performance and (5) organizational and non-functional area.

The most important risk impact area is connected to the problems related to new technologies. The majority of software development projects is connected with the application of new technologies. The improper use of new technologies usually leads to project failure. Knowledge is the main problem connected with the use of new technologies (Sertić, 2002). In order to implement new technology, a development team should have sufficient knowledge about it. The importance of knowledge about technologies involved in a software development project is often disregarded. Software solution is built upon technological features, which are often new to a project development team. These technological features are often predicted at the beginning of a software development project, and only a software prototype can prove these predictions (Larman, 2002). The knowledge about these technological features can refine predictions about features and minimize the risk connected with the use of these technologies. A software development team should have sufficient knowledge in order to efficiently exploit technological features. The knowledge about the involved technologies is a precondition for a successful software development project (Ould, 1998).

The second important risk impact area is connected with software requirements. Software requirements are a common term for all users' needs regarding software system functionality and quality of service. It is often very hard to develop the right software solution that absolutely meets users' expectations. In order to develop a software solution according to user expectations, a software development team should discover a whole set of user requirements (Larman, 2002). These requirements, that must guide the entire development process, can be divided into functional and non-functional (Larman, 2002). Functional requirements define behavior of the software solution, while non-functional requirements define qualities and constraints to which the software solution must conform.

The process of the identification and definition of requirements is long and complicated, so it is common that requirements change during the realization of a software development project (Booch, Rambaugh & Jacobson, 2001). The change of requirements is a major problem in software development because the change of one or few requirements could impact the complete software solution and lead to the failure of a software development project. As a result, it is absolutely necessary to find the right requirements and to manage the change of requirements during a software development project.

The third important risk impact area is the architecture of a software solution. Software architecture could be described as a set of significant

decisions about the organization and components of a software solution (Booch, Rambaugh & Jacobson, 2001). Software architecture must be defined in the early development phases in order to build a quality software solution. It is possible that software architecture defined in the early development phases does not satisfy all of the requirements set on a software solution. The software architecture can be verified only with a software prototype, which can be realized only in later development phases. Due to the importance of software architecture, many development processes focus directly on software architecture in order to build a quality software solution according to the defined requirements (Royce, 1998).

The fourth risk impact area is connected with software system performance. In order to satisfy non-functional requirements, a software solution should have satisfactory performance. The performance of a software solution could be tested only on a real and realized software solution. Thus, it is necessary to make predictions about software system performance in the early development phases. These predictions are very important because it is possible to develop a software solution that satisfies functional requirements, but it is too slow to fulfil performance requirements (Karolak, 1995). As a result, development team members, with experience in given technologies, should do performance predictions and assumptions.

The fifth risk impact area could be considered as the organizational and non-functional area. The risks connected with this area could be described as organizational problems and problems related to the project resources and schedule. Organizational problems may affect the realization of a software solution since only the efficient organization of software development leads to a successful software development project (Sertić, 2002). A defined project schedule could become a risk because there are many unwanted events that could cause a delay in software realization. It is a management problem to define a project schedule in order to satisfy both customers and developers (Ould, 1996). In order to fulfil defined project deadlines, resources given to the project should be sufficient. This means that every software development project should have enough project members with the right competencies and all the required resources for the planned completion.

Besides risk identification and specification, risks in software development should be addressed and managed. In software development, there are many choices of dealing with risks (Ould, 1998). Based on the risk impact area and type, risks can be avoided, restricted, mitigated and monitored (Jones, 1994). The best possible way of risk addressing is to completely avoid risks. There are

a number of risks that can completely be avoided by the use of different technologies, changing requirements or project plans. The next best way to address risks on the project is to confine them. A risk can be confined in order to restrict the risk impact area to affect only a small part of a software development project. In order to reduce the impact area, risks must be identified and some changes on the project should be made regarding the technology and resources used. This is a proper way of addressing risks that cannot be avoided.

There are many risks that cannot be avoided or confined. These risks should be mitigated or monitored. Some risks on a project 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 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 these risks, contingency plans should be done 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 decisions about the project future should be made. 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 could be enormous (Larman, 2002).

Addressing risks is time-consuming. It takes time to change project plans in order to avoid some risks and it takes time to build prototypes needed for risk mitigation. Thus, it is difficult to address all risks at the same time. In order to successfully address risks that arise on a software development project, it is necessary to divide risks into categories by priority and to plan software development according to risk priorities. For successful project planning and realization according to risks categorized by priority, it is required to have risk management as a continuous activity on a software development project.

3. RISK MANAGEMENT

The key to risk management is the identification and mitigation of all true risks or the development of a contingency plan in case the potential risk becomes a reality (Charette, 1989). The process of risk management and risk

mitigation is connected with preventing huge losses in software development. Risk management should focus on risk reduction and prevention. Software risk management is defined as practice for managing risks that occur in a software development project (Hall, 1998). Risk management should continuously assess possible problems on a software development project and define potential risks, determine what risks are important to deal with and implement strategies to deal with those risks. This means that the global project picture is required for successful risk identification, but every project member should assess and identify risks in project areas defined by his role in the project. There are four basic steps in risk definition: identification, assessment, mitigation and conclusion (Capers, 1994).

The first step in risk definition is risk identification, which is responsible for the recognition of potential losses and their causes. In order to implement successful risk management, project team members should have a global perspective on the software development project. Risk assessment should determine the level of exposure to potential loss caused by risk materialization (Jones, 1994). The mitigation step is responsible for the creation of a risk avoidance plan, while the conclusion step describes the execution of risk avoidance and mitigation plans. These steps will lead to a complete description of all risks, which should be captured in a formal document called the *Risk List*. This document should contain all risks with the description of definition, consequence, likelihood, risk ranking, indicators, risk mitigation strategy and contingency plan for every possible risk on a software development project.

The process of risk management should start with risk identification. The purpose of risk identification is to discover all factors that could lead to project failure (Hall, 1998). These factors are connected with the technology used on the project, software development process and organizational factors. These areas should be observed and assessed in order to capture all of the potential risks. It is necessary to capture details connected with the discovered risk, like risk description, probability of risk occurrence, costs connected with materialized risk and possible risk solutions and avoidance strategies.

The second step of the risk management process is to assess the level of exposure for each risk. In this step, discovered risks should be ranked in levels according to risk impact (Capers, 1994). Risks should be classified according to the degree of impact in order to choose important risks to be solved first. Risks with a devastating impact should be assessed before risks with a low impact. This is important because risks with a huge impact should be considered in the early development phases, when the costs connected with risk materialization

and project failure is smaller than in later development phases (Booch, Rambaugh & Jacobson, 2001).

After risk assessment, risk mitigation is the next step in the risk management process. Risk mitigation is an attempt to avoid or prevent the consequences of risk materialization (Ould, 1998). There are three main strategies of risk mitigation: risk avoidance, risk reduction and risk transfer (Hall, 1998). Risk avoidance is the best possible answer to risk materialization, but there are times when it is very difficult to avoid risks on a software development project. The best way to avoid risks is to completely reorganize the software development project, which is sometimes impossible. If risks cannot be avoided, they can be reduced or transferred. Risk reduction is connected with re-planning the software development project in order to reduce the probability of risk occurrence. Risk transfer could be described as a project reorganization strategy in order to forward the risk to areas where it would cause less damage. Risk mitigation plans should be defined as soon as possible for every identified risk.

The final step in the risk management process is risk conclusion. This step is taken after the definition of a risk mitigation plan and includes all the actions needed for risk avoidance or actions, which are required after the risk materializes. The actions taken in the conclusion step should be defined in the contingency plan. The contingency plan should describe actions, which are taken once a risk becomes a reality.

4. STRATEGIES FOR RISK MANAGEMENT

In this chapter, we present three strategies for risk management. These strategies are defined in order to support various types of software development projects according to the amount of risk influence. Based on the amount of risk on a software development project, we define three risk management strategies: (1) careful, (2) typical and (3) flexible.

Careful risk management strategy is intended for fresh and inexperienced organizations whose software development projects are connected with new and unproven technology. Typical risk management strategy is defined as a support for mature organizations with experience in software development projects and used technologies, but whose projects carry a decent number of risks. Flexible risk management strategy is engaged in experienced software development organizations whose software development projects are formally defined and based on proven technologies.

Careful risk management strategy should be used on state of the art software development projects, which are completely unknown to the organization. This risk management strategy should be used on a software development project with high acceptable risks, new technology and inexperienced people. Careful risk management strategy could be described as high priority risk management. Risks on software development projects, with an implemented careful risk management strategy, should be taken with utmost importance. This requires rigorous risk analysis on multiple levels. Risks should be analyzed and traced on individual, team and organizational levels. In order to identify important risks as soon as possible and on a wide problem area, every project team member should be involved in risk management. This means that every project member should identify and define risks connected with his problem area, but risks should also be identified and defined on the team and organizational area. Risk mitigation strategies must be defined for each organizational level.

In order to successfully implement the risk management strategy, an organization should define risk management roles on the software development project, i.e. there should be project members whose primary activities are connected with risk management. Since careful risk management strategy is connected with an extremely high number of risks on a software development project, an organization should formally define risk interpretation and ranking policy. This should help to separate the important from the unimportant risks in order to address important risks as soon as possible. Project team members should continuously trace risks and actions connected with risks. Work on software development must be ordered according to project risks, i.e. risks with high importance should be solved first.

This requires constant planning and project supervision, but with this approach, risks will be mitigated in the early phases of software development, when the cost of a software development project is still small. Along with constant risk identification and project planning, the risk list should be reordered every time a new risk is identified because the most important risks should be first on the list. Each risk ought to be formally well-described and risk mitigation and contingency strategies should be defined. Careful risk management strategy is dedicated to the software development project under high-risk influence and, with this approach, risks can be well-understood and addressed on time. Activities of careful risk management strategy are presented in Fig. 1 using UML (Unified Modeling Language) Activity Diagram (Larman, 2002).

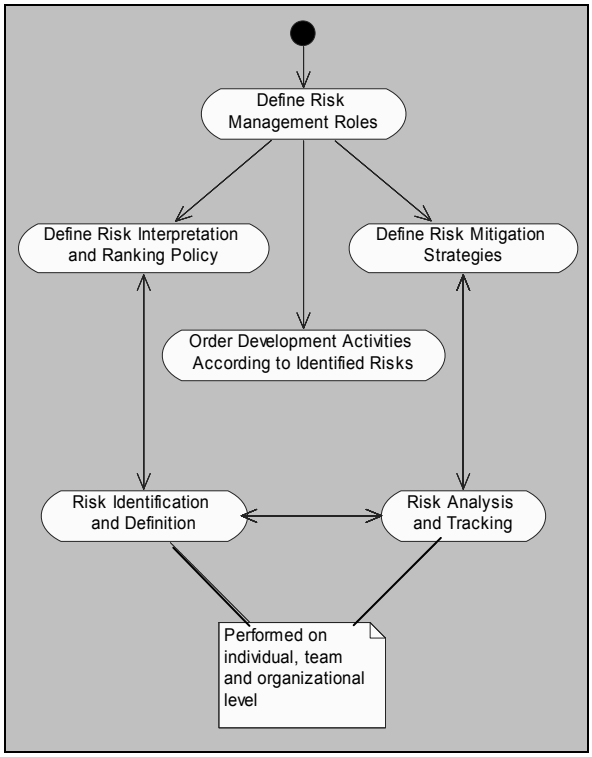

Figure 1. Activities of careful risk management strategy

Typical risk management strategy is intended for mature software development organizations with experience in software development. This risk management strategy should be used on software development projects with a certain level of known technology and mostly experienced people. The level of acceptable risks for this strategy is medium, i.e. there are risks, which could be considered as too harmful, and projects with those risks would be cancelled. With this strategy, risks should be identified in steps according to project progress.

The main difference between careful and typical strategy is in risk importance because the typical risk management strategy assumes that most of the risks on the project are easily addressed, while the careful risk management strategy assumes a high number of dangerous risks. Risk management should mostly be practiced on the team and organizational level because risks on a well-defined project are connected with the entire project area. This risk strategy does not require project roles responsible for risk management because

risks should be identified by the project management team and occasionally by project team members. As with the careful risk management strategy, risks should also guide software development in order to early address important risks, but because a relatively small amount of risks have an influence, project plans should be stable and only sporadically changed. In this strategy, risk interpretation and ranking policy is less formal because risks should be defined and ranked by the project management team. This strategy is intended for software development projects connected with mostly known technologies because this strategy should handle sporadically materialized risks. Activities of the typical risk management strategy are presented in Fig. 2.

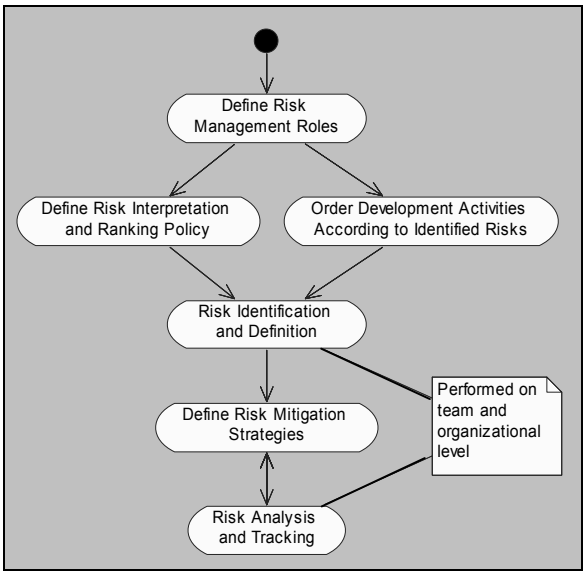

Figure 2. Activities of typical risk management strategy

Flexible risk management strategy is defined for a mature organization with formally defined software development projects, which use known and proven technology with efficient organization. This strategy assumes a small number of acceptable risks, which are mostly transferred to other organizations. Flexible strategy requires a relatively informal risk definition, with risk mitigation and contingency plans defined only for a few important risks. This is defined in order to minimize the amount of work required for risk management because there is a small level of risk influence in mature and experienced software development organizations. This strategy is based on comparing currently identified risks to previously encountered ones and risk management should be practiced mostly on the organizational level. There is no risk

interpretation and ranking policy connected with the flexible risk management strategy, mostly because of the small risk influence and importance on formally defined software development projects. Risks should be identified at the beginning of the software development project and occasionally during the project since there is a small influence of change on these software development projects. Initial project plans should be made according to identified risks, but eventual risk materialization should not change project plans because of the small risk influence. Flexible risk management strategy is intended for mature and experienced organizations in order to achieve efficient risk management and to improve project conditions. Activities of the flexible risk management strategy are presented in Fig. 3.

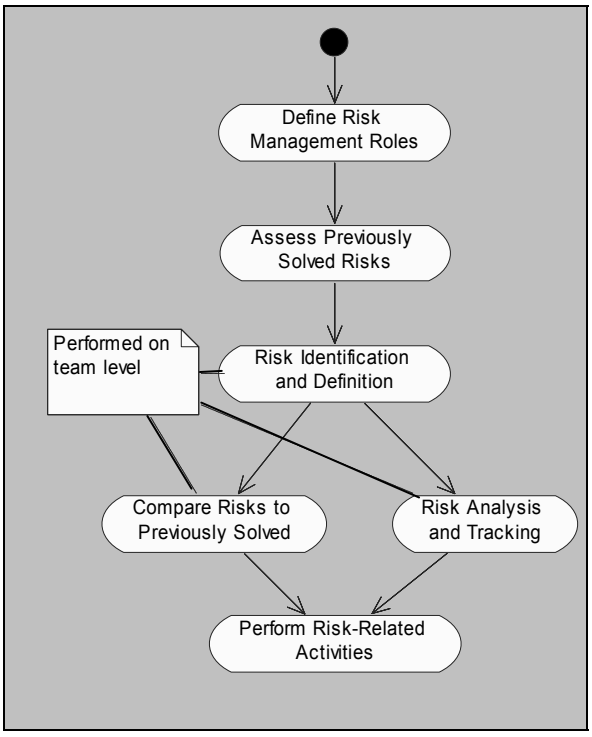

Figure 3. Activities of flexible risk management strategy

The selection of a suitable risk management strategy should be primarily based on organizational experience and the quality of project organization, but software development project complexity, use of new technologies, stability of requirements and competencies of project members should also be considered. We believe that the careful risk management strategy should be used in the

initial project phase and that the risk management strategy should be reconsidered and possibly changed to typical or flexible on each project milestone. Proposed strategies are the result of a theoretical study based on real experiences from software development projects at Ericsson Nikola Tesla. Activities defined in the careful risk management strategy are a result of experiences from the IP Telex development project, whose project goal was to build a telecom exchange solution based on a commonly used computer platform. The two other strategies are a theoretical proposition for more mature development projects than the current ones at Ericsson Nikola Tesla Company and should be tested in the future.

5. CONCLUSION

This work describes risks in software development. Risks are present in every software development project because software development is based on knowledge and new technologies, and the chances for success of a software development project are closely connected with successful risk addressing. As a result of that, we have closely investigated risks and risk impact areas in software development projects. With this paper, we propose a key element of modern software development practices to be software risk management. In order to achieve efficient risk management, we have proposed three risk management strategies suitable for different software development projects according to the amount of risk impact.

We have also proposed a risk-based approach to development planning and risk management as an attempt to address and retire the highest impact risks as early as possible in the development process. The risk-based approach to software development should enable early risk addressing and conclusion when the expenses connected with risk materialization and project failure are small and insignificant. Strategies proposed in this work should enhance risk management on software development projects and increase project chances for success.

From the global business perspective, the success of many enterprises is becoming increasingly dependent on the success or failure of the software they build. It is not important if software is intended to be sold, internally used or to drive business transactions, the future of many enterprises is connected with the software they develop. Thus, risk management is not only a crucial development practice, but also a vital business practice.

6. REFERENCES

1. Hall, E. (1998): *Managing Risk: Methods for Software System Development,* Addison-Wesley, New York
2. Jones, C. (1994): *Assessment and Control of Software Risk,* Prentice-Hall, New York
3. Sertić, H. (2002): *Applying Unified process on complex software system development,* Master Thesis, Economic Faculty, University of Zagreb, Zagreb
4. Booch, G.; Rambaugh, J.; Jacobson, I. (2001): *The Unified Software Development Process,* Addison-Wesley, New York
5. Karolak, W. (1995): *Software Engineering Risk Management,* Wiley-IEEE Press, San Francisco
6. Royce, W. (1998): *Software Project Management: A unified framework,* Addison-Wesley, New York
7. Larman, C. (2002): *Applying UML and Patterns,* Prentice Hall, Upper Saddle River
8. Robertson, S.; Robertson J. (2001): *Mastering the Requirements Process,* Addison-Wesley, New York
9. Capers, J. (1994): *Assessment and Control of Software Risks,* Prentice Hall PTR, Upper Saddle River NJ
10. Ould, M. (1996): *Strategies for Software Engineering: The Management of Risk and Quality,* John Wiley & Sons, San Francisco
11. Charette, R. (1989): *Software Engineering Risk Analysis and Management,* McGraw Hill, New York
12. Ould, M. (1998): *Managing Software Quality and Business Risk,* John Wiley & Sons, San Francisco

STRATEGIJE ZA USPJEŠNO UPRAVLJANJE RIZIKOM U RAZVOJU SOFTVERA

Sažetak

Softver je danas postao jednim od ključnih čimbenika poslovanja. Razvoj softvera je aktivnost povezana s razvijenom tehnologijom i visokom razinom znanja. Proizvodnja uspješnog softverskog sustava traži uspješno smanjivanje rizika u projektu softverskog razvoja. Jedan od čestih razlika za neuspjeh projekta leži u nedostatku planiranog pristupa upravljanju rizikom. Ovaj rad istražuje temeljna područja djelovanja rizika, kako bi se stvorio opći teorijski pristup upravljanju rizika u razvoju softvera, a s ciljem poboljšanja uspješnosti softverskih projekata. Na temelju tipičnih područja djelovanja rizika na proces softverskog razvoja, predlažu se tri strategije upravljanja rizikom, pogodne za veliki broj poduzeća i projekata razvoja softvera, koje obilježava različita razina rizika. Predloženim se strategijama definiraju aktivnosti koje bi trebalo obavljati u okviru uspješnog upravljanja rizikom, kao i aktivnosti vezane uz poboljšanje percepcije rizika i uklanjanje problema nastalih uslijed negativnih posljedica rizika. Autori također predlažu pristup utemeljen na upravljanju rizikom kao pokušaj identificiranja i uklanjanja najznačajnijih oblika rizika projekta softverskog razvoja, i to u što ranijoj fazi projekta. Predložene bi strategije trebale poboljšati upravljanje rizikom u projektima razvoja softvera, te pomoći u stvaranju uspješnih softverskih rješenja.