Implementing risk management in software projects

Jakub Miler
Janusz Górski
Department of Applied Informatics
Technical University of Gdańsk
Gdańsk, Poland

Abstract

The paper presents a tool supporting risk management in software projects. It provides some details of the functionality, user interface, design and implementation of the tool. It also reports on an experiment that was carried out in order to provide an early feedback on the usability of the tool.

1. Introduction

Software projects are exposed to various risks where risk is understood as a possibility of loss, damage or disadvantage. Essential features of a risk are that it brings about negative consequences and that there is a degree of probability that these negative consequences come true. The risks related to the whole project are defined in relation to project goals. The goal of any project is to deliver, in time and within the budget constraints, a product that meets client's needs and expectations. The essential factors of the project success are the quality, the time and the budget [11]. In the essential project aspects, the lack of risk management results in schedule slippage, budget overrun, unsatisfactory quality of the product, failure to accomplish company business goals of the project and disappointment of the employees and breakdown of their careers. More information on a general approach to risks can be found in [2, 3, 9, 11].

Present software projects are often facing expanding and changing client demands and are put under schedule pressure. The systems are growing in size and become increasingly complex. To shorten the development time the systems are built out of reused (but often not reusable) components. The personnel turnover is high and the size and diversity of project groups are growing. Consequently, despite of the progress in technology, the software engineering project management still faces similar problems as twenty years ago [1]. The requirements are not defined precisely which results in constant expansion of the system scope or even in rejection of the final system. The involvement of people is relentlessly adding the factor of human mind and personality to the technical difficulties of the project. The resulting software is constantly error-prone, the co-operation among the project members is often poor and the expectations of the clients are not satisfied. Altogether, it calls for some significant improvements to the software engineering process. One of such innovative approaches is risk management.

Risk management means that we change our attitude towards risks. A project without risk management faces serious problems only after the risks came to the surface as a material fact (the deadline is not met, the budget is overrun, the quality is poor). Then, the only thing to do is to strive to minimise the negative influence of the consequences of those facts on the project. The reaction is always expensive and time-consuming. A project with risk management aims at early identification and recognition of risks and then actively changes the course of actions to mitigate and reduce the risk. This requires open communication, forward-looking view and team involvement in the management and developing and maintaining a knowledge base of typical problems. The lack of these exposes a project to a great risk of failure [4].

Much work on risk management has been already done at the Software Engineering Institute [12]. The generic risk management methodology tailored to the specific environment of a software engineering project was transformed into few recurring phases [4, 5, 9, 10, 11]. The subsequent phases cover the following risk management aspects:

- Risk assessment:
  - identification,
  - analysis
- Risk mitigation:
  - planning,
  - tracking,
  - controlling
- Communication.

The process starts with the identification of existing risks. Once the risks are identified, they are evaluated and prioritised and then appropriate corrective actions are planed and executed. To reach the acceptable level of success guarantee, the introduced actions must be controlled and the risks in mitigation must be continuously tracked for their status. This process continues throughout the project schedule and across all the phases of development.

The risk management paradigm defines a set of continually performed activities that are based on communication among the project participants including all the project members, the customer representatives and the upper management of both, the developer and the customer's organisation. The cost and effort of these activities must be smaller than the estimated profit from the successful risk mitigation. To reach the desired 'return of investment', the risk management efficiency may be increased by means of computer-based tools.

This paper describes a tool, named () Risk Guide, implementing risk management in a software project. The paper details the usage of the tool, provides some design and implementation data and suggests possible applications. This is followed by the presentation of a validation experiment. In conclusion we present plans of further extensions of the tool.

2. The system

() Risk Guide is a pilot implementation of an experimental risk management tool. The goal is to have an environment where we can experiment with various risk management strategies and techniques. Presently the tool offers support mainly in the risk identification phase. In general, it is built upon the concept of project reviews that are aimed at gathering risk-related information from project members. The result is the list of identified risks summarising the input collected from all the project members. Further on, the managers can use this list to elaborate effective mitigation plans. This way the tool supports communication of risk information within the project.

The present support by () Risk Guide comes from the automatic risk identification based on the answers to the checklist questions. The system offers a knowledge base of checklists that are related to the lists of common risks. The knowledge base is hierarchically structured that facilitates choosing checklists that are relevant to the viewpoint of the particular user. The tool is designed to support many members of a project and multiple projects at a time with independent risk identification. All those projects are sharing the same knowledge, however.

() Risk Guide records and maintains information on:

- projects developed at the time,
- system users (both members and non-members of the projects).

- reviews of risk at certain point of the development,
- risks identified during the review.

The users of () Risk Guide take different roles in the risk management process. They can be divided into the following categories:

• Administrator	–	The supervisor of the system, who manages user accounts.
• Risk Manager	–	The manager of the risk management process, who edits project and review data, assigns the other roles to people and evaluates identified risks.
• Project Member	–	A project member that, through checklists, supplies risk related information.
• Non-member User	–	Any user. He/she can browse the knowledge base of checklists and risks, read project descriptions and get the information on risks identified during reviews.

The system implements some access control mechanisms. The Administrator, Risk Manager and Project Member users are subjected to the authentication process (password based) whereas Non-member User identities are not controlled (note however that the latter are restricted to read-only access).

The general scenario of () Risk Guide usage includes:

- registration of system accounts for all the users by the Administrator,
- registration of a project by Risk Manager,
- opening a review by Risk Manager,
- answering the checklist questions by Project Members followed by automatic risk identification,
- closing the review by Risk Manager,
- evaluation of identified risks by Risk Manager,
- selection of the highest priority risks by Risk Manager.

The list of identified risks can be further analysed to provide some metrics, e.g.:

- the number of checklist answers pointing to a particular risk – it characterises the estimate of 'intensity' of the risk,
- the number of Project Members who identified a particular risk – it characterises the extend of the risk perception,
- the percentage of questions answered - it characterises the risk identification coverage.

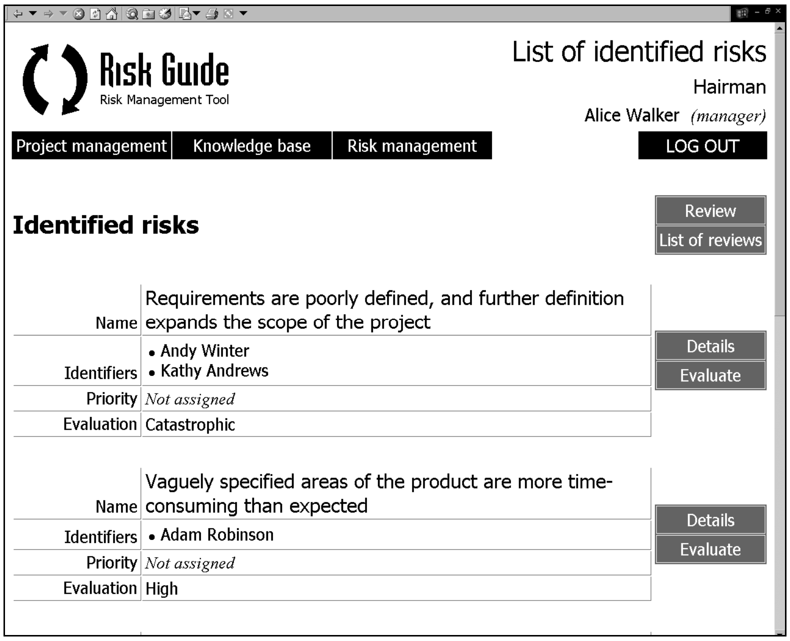
After preliminary analysis of identified risks, Risk Manager should thoroughly investigate the selected high-priority risks and decide on the resolution method. The tool helps the Risk Manager to focus on the most important risks and spend project resources efficiently. More details on the present version of Risk Guide can be found in [8].

3. User interface

The user interface of Risk Guide is a set of ASP web pages accessed with an Internet browser. Example pages are presented in figures 1 and 2.

The most complex data structure is a checklist. It is implemented as a hierarchic construct with the top item being the checklist itself. The checklist consists of many 'hierarchy classes' being the chapters, subchapters and points. The structure is recursive, so its depth is not limited by the design. Each 'hierarchy class' contains questions that form another recursive data structure of sub-questions with no design-defined limit. Each question has a set of answer alternatives assigned to it. The risks stored in the knowledge base are also structured using the concept of recursive 'hierarchy classes'. The contents of the checklists and the risk lists are fixed – presently no modification functions are available to the users.

As for now, Risk Guide knowledge base offers the SEI Taxonomy-Based Questionnaire [10] as a checklist and Steve McConnell's Complete List of Schedule Risks [6, 7] as a list of risks. Due to the flexible data structures, however, it is possible to extend the lists of checklists and risks as well as to modify the relationships between them.



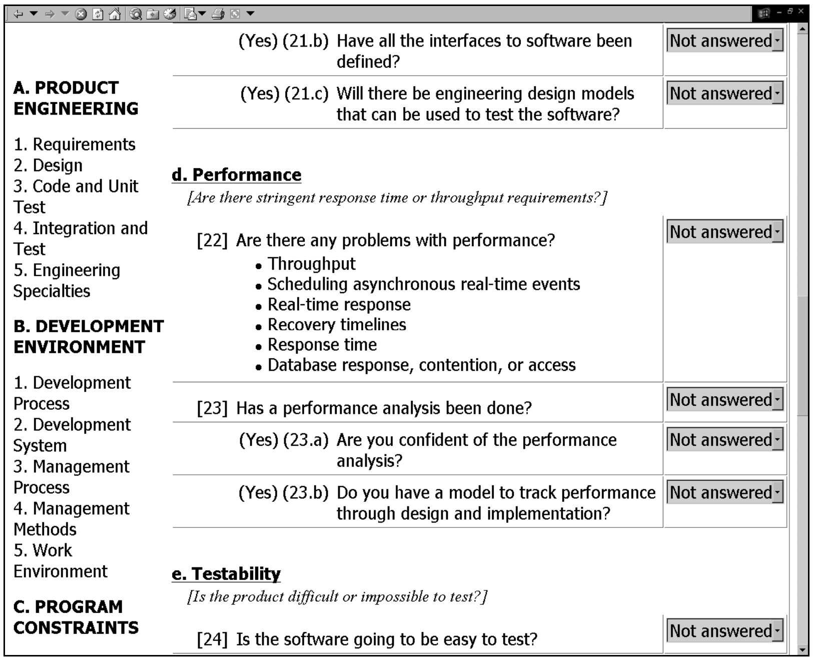
List of identified risks
Hairman
Alice Walker *(manager)*

Identified risks

Name	Requirements are poorly defined, and further definition expands the scope of the project
Identifiers	• Andy Winter • Kathy Andrews
Priority	*Not assigned*
Evaluation	Catastrophic

Name	Vaguely specified areas of the product are more time-consuming than expected
Identifiers	• Adam Robinson
Priority	*Not assigned*
Evaluation	High

Figure 1: List of risks identified in a review



A. PRODUCT ENGINEERING

1. Requirements
2. Design
3. Code and Unit Test
4. Integration and Test
5. Engineering Specialties

B. DEVELOPMENT ENVIRONMENT

1. Development Process
2. Development System
3. Management Process
4. Management Methods
5. Work Environment

C. PROGRAM CONSTRAINTS

(Yes) (21.b) Have all the interfaces to software been defined? Not answered

(Yes) (21.c) Will there be engineering design models that can be used to test the software? Not answered

d. Performance
[Are there stringent response time or throughput requirements?]

[22] Are there any problems with performance? Not answered
- Throughput
- Scheduling asynchronous real-time events
- Real-time response
- Recovery timelines
- Response time
- Database response, contention, or access

[23] Has a performance analysis been done? Not answered

(Yes) (23.a) Are you confident of the performance analysis? Not answered

(Yes) (23.b) Do you have a model to track performance through design and implementation? Not answered

e. Testability
[Is the product difficult or impossible to test?]

[24] Is the software going to be easy to test? Not answered

Figure 2: Fragment of a checklist chapter

4. System design

RiskGuide design is object oriented and is implemented on the top of a relational database. This required that the data transformation layer – a database wrapper must have been implemented. Due to the wrapper concept, the database data structures are manipulated like objects from the user interface. It implies that the data records have their identities assigned resulting in object-like data editing functions. RiskGuide provides object versioning and exclusion that ensures change traceability and modification rollbacks. The objects excluded from edition cannot be further modified and used in new references, but they keep the existing references intact. Further on, the excluded objects can be included back or deleted for good.

The system architecture conforms to the client-server model with one central server possessing and providing all the information and the user terminals connected via the network of any type. It is a universal approach suitable for local area networks and the Internet.

The RiskGuide software architecture uses the three-tier distributed application concept developed by Microsoft. According to this concept, the applications are built at three levels: database, server components and interface. This concept is helpful in implementation of Internet applications. The technology used in the construction of Risk Guide implies the following features of the system:

- encapsulation of data processing,
- limited software requirements for client workstations,
- multi-access,
- effective communication,
- open user interface,
- easy system updating and development.

5. Validation experiment

The experiment was planned and carried out in order to test the usefulness of the system.

In the academic year 2000/2001, Risk Guide was used as a supporting tool during the course on Software Project Management at the Technical University of Gdańsk. The course was given to the students of Informatics, during their fourth year of study. There is a student project associated with the course. The project aims at the familiarisation with the selected project management activities such as effort estimation, project planning and system analysis. The students work in groups and their task is to practice effort estimation and project planning with respect to, in most cases imaginary, projects (6 of those projects were real, related to other courses). As the first step a group (of 3-4 people) worked out a project vision and elaborated a preliminary project plan. Then, they used Risk Guide and practised risk identification. The output from the system (the identified risks) was then taken under consideration in the next step: the development of a detailed project plan. The experiment constraints are given in Table 1.

Table 1. Experiment constraints

No. of groups	38
Group size	3 to 4 students
Range of projects considered by groups	3 man-months (3 persons / 1 month) ÷ 40.5 man-years (27 persons / 1.5 years)
Statistics of projects	22 projects of size from 10 to 40 man-months, 10 projects of size from100 to 250 man-months, 6 other projects
Experiment duration	10 weeks
Phases covered	Planning, Design
Tasks	Prepare a preliminary project plan; Identify project risks (answer TBQ questions, evaluate risks, select top 5 risks, describe top 5 risks, prepare mitigation and contingency plans for top 5 risks); Elaborate detailed project plan.

Deliverables	Preliminary project plan, Risk management plan (list of identified risks generated by Risk Guide); Risk evaluation; Detailed description of top 5 risks; Mitigation and contingency plans for top 5 risks; Detailed project plan.

During the experiment, the students identified many different risks in their projects. While some risks had a generic nature, like incompleteness of the requirements or lack of developers' experience, many risks were highly project-dependent, like unfamiliarity with certain development tools, high system complexity or lack of management skills. However, because most of the projects were not actually carried out, these results could not be confronted with reality and just reflected the students' knowledge, experience and intuition.

To assess the experiment, a questionnaire was distributed among the students to gather their opinions about Risk Guide and the effectiveness of its support. In total, questionnaires from 25 groups were collected. The questionnaire answers, scaled from 1 (poor) to 5 (excellent), provided the following results:

Table 2. Tool assessment

overall system presentation	3.94
readability of the user interface	3.80
ease of access to system functions	3.06
overall ease of use	2.82
system performance	4.42
system reliability	4.34

Table 3. Support assessment

better understanding of vulnerable project areas	3.45
pointing out potentially omitted problems	3.77
help in elaboration of more realistic detailed plans	3.24
increase of awareness and interest in risk management	3.03
acquirement of general knowledge on software engineering projects	3.99

Finally, the participants were asked if, in their opinion, the system could be useful in the risk management process of a real-life software engineering project. The answer was 3.29.

6. Conclusions

In this paper we pointed out to the importance of risk management in the course of software projects and to the need of tool support in this area. We presented a risk management supporting tool, detailing its functionality and the user interface appearance as well as giving some design and implementation details. The tool offers in particular:

- automatic risk identification from interactively answered on-line checklists,

- qualitative risk evaluation,
- taking into consideration the opinion of every individual project member,
- great communication support for distributed project teams,
- effective information sharing and distribution,
- support for many projects and reviews at a time.

We have also presented the results of a validation experiment carried out on a number of student projects. Although it is not enough to derive any definitive conclusions, the results of the experiment are in our opinion quite encouraging.

The following features are considered important for future development of the system:

- mitigation functions of risk management process: planning, tracking and controlling,
- further communication support (informal messages, impromptu risk identification and tracking),
- checklists management including modifications and introduction of new checklists,
- knowledge base of common mitigation activities,
- risk evaluation dependent on project size registered in the knowledge base,
- risk analysis tools with great visualisation capability and probability calculus support.

It is expected that after further development of () Risk Guide, its application in a real-life project can bring in the following benefits:

- clarification and enforcement of the risk management process,
- encouraging active and constructive involvement of all project members,
- risk management becomes routine activity (like checking for new email),
- thoroughness of risk management,
- formalisation of risk management process,
- accumulation of project historical data,
- efficient constant access to risk information,
- easier decision-making due to visualisation of risk information,
- development of risk management standards.

References

[1]. F. P. Brooks, Jr., *The Mythical Man-Month, Essays on Software Engineering*, Anniversary Edition, Addison Wesley Longman Inc., 1995.

[2]. B. P. Galagher, *Software Acquisition Risk Management Key Process Area (KPA) – A Guidebook Version 1.02*, SEI report CMU/SEI-99-HB-001, Carnegie Mellon University, Pittsburgh PA, October 1999.

[3]. *Inżynieria oprogramowania [pol. Software engineering]*, edited by J. Górski, Wydawnictwo Mikom, 2nd edition, 2000, pp. 415-455, in polish.

[4]. R. P. Higuera, D. P. Gluch, A. J. Dorofee, R. L. Murphy, J. A. Walker, R. C. Williams, *An Introduction to Team Risk Management*, SEI report CMU/SEI-94-SR-01, Carnegie Mellon University, Pittsburgh PA, May 1994.

[5]. R. P. Higuera, Y. Y. Haimes, *Software Risk Management*, SEI report CMU/SEI--96-TR-012, Carnegie Mellon University, Pittsburgh PA, June 1996.

[6]. S. McConnell, *Code Complete*, Microsoft Press, 1993.

[7]. S. McConnell, *Rapid Development*, Microsoft Press, 1996.

[8]. J. Miler, *Computer system for supporting risk management in a software engineering project*, Master's dissertation, Technical University of Gdańsk, Poland, 2000.

[9]. L. H. Rosenberg, Dr., T. Hammer, A. Gallo, *Continuous Risk Management at NASA*, presented at the Applied Software Measurement / Software Management Conference, San Jose, CA, February 1999.

[10]. F. J. Sisti, S. Joseph, *Software Risk Evaluation Method*, SEI report CMU/SEI-94-TR-19, Carnegie Mellon University, Pittsburgh PA, December 1994.

[11]. R. L. Van Scoy, *Software Development Risk: Opportunity, Not Problem*, SEI report CMU/SEI-92-TR-30, Carnegie Mellon University, Pittsburgh PA, September 1992.

[12]. http://www.sei.cmu.edu/