Software Risk Management: a Process Model and a Tool

Tereza G. Kirner[1], Lourdes E. Gonçalves[1]

[1]Graduate Program in Computer Science
Methodist University of Piracicaba – SP, Brasil
tgkirner@unimep.br; lgoncalves@unasp.edu.br

Abstract. This paper is concerned with the risks associated with the software development process. A model (*GRisk-Model*) is proposed for the management of such risks and a software tool (*GRisk-Tool*), developed to support the model, is described. Both the method and the tool were created with the participation of senior managers and software engineers of software factories. The model and the tool serve as effective instruments for achieving the continuous improvement of software processes and products.

1 Introduction

Several approaches of software risk management have been proposed and used since Boehm [1], [2] and Charette [4], [5] introduced the topic and its importance in the software engineering context. However, despite of several studies and experiences published about risk management, the software industry, in a general way, does not seem to follow a model to analyze and control the risks through the development of their products.

This article comprises two objectives. The first one is to present a model of risk management process (*GRisk-Model*), that covers all the stages of the software development process. The second is to present a tool (*GRisk-Tool*) that supports this model. The *GRisk-Model* was proposed with basis on the literature and from the experience of managers and senior software engineers of Brazilian software factories. The *GRisk-Tool* implements the proposed risk management model and also was evaluated by professionals, with respect to its functional aspects and obtained benefits.

Section 2 points out the theoretical basis that has supported the proposal of the model and the construction of the tool. Section 3 details the *GRisk-Model* and section 4 presents the *GRisk-Tool*. Section 5 presents the conclusions, stressing the potentialities, limitations and future directions of the work.

2 Related Work

Risk has to do with any variable that can lead to the failure of the project. Generally, risk can include problems related to deadlines, requirements, budget and staff [9].

Please use the following format when citing this chapter:

Kirner, T.G., Conçalves, L.E., 2006, in IFIP International Federation for Information Processing, Volume 227, Software Engineering Techniques: Design for Quality, ed. K. Sacha, (Boston: Springer), pp. 149–154.

According to Pressman [10], there is a considerable debate regarding the accurate definition of software risk, but there is a consensus that risk always involves two characteristics: (a) Uncertainty, which means that an event that characterizes the risk can either happen or not, that is, there is not 100% of probability of the risk to occur. (b) Loss, which means that, if the risk becomes a reality, undesirable consequences will occur involving damages to the product in question.

Risk management comprises a systematic approach of evaluating the risks related to the software development process. A typical risk management model involves the identification and analysis of the potential risks of a project and, moreover, the adoption of monitoring strategies for reducing these risks.

One of the precursors of the area of risk management is Barry Boehm who, in 1988, proposed the Spiral Model that incorporates successive analyses of risks along the software development stages [1]. Later, this same author defined the risk management as a process composed of two phases: (a) Risk evaluation, that includes the identification of the risk, the analysis of the risk, and the prioritization of the risk. (b) Control of the risk, that includes a plan of risk management, the resolution of risks, and the monitoring of the risks [3]. Another well known model of risk management is the RISKIT [11], that incorporates a similar process to that proposed by Boehm [3], including the following stages: (a) definition of a risk management program; (b) review of the objectives of the project; (c) identification of the risks; (d) analysis of the risk; (e) planning of the risk control; (f) control over the risks; (g) monitoring of the risks.

The benefits propitiated by the tools that assist in the software development, as CASE tools, prototyping tools, etc., are unquestionable. Among these tools destined to support the risk management, discussed in the literature, ARMOR [8] and SERIM [7] tools are distinguished.

ARMOR (Analyzer for Reducing Module Operational Risk) aims to detect and evaluate software risks, based, mainly, on statistical models. The execution of the tool includes a series of functions, which make possible to: access the data that are pertinent to the characteristics of software; use and evaluate metrics applied to the software product; evaluate risks of performance; identify, validate, calculate and present the risks related to each software module, including indication of actions for the risk reduction. SERIM (Software Engineering of Risk Management) supports the identification of a reliable process for software development, based on the identification of the potential risks and the stages and activities of the project that need a more accurate attention. After identifying the risks, the tool assists the elaboration of plans for minimizing the latent risks, including since the identification of risks related to the system implementation until the involved costs and the defined deadlines.

3 Risk Management Model

The definition of the *GRisk-Model* counted on the participation of a team of professionals composed of 1 commercial manager, 1 manager of software factory, 3 coordinators of software factories, and 3 senior system analysts. For the stages, phases and activities of the software development, descriptions have been prepared and indicated

the classes and the risks associated to those. Periodically, meetings were held in which the group came up with an evaluation of the classes and the risks indicated for each phase and activity. At the end of the work, forms for analysis of the proposed structure were filled up through which the professionals of the work team informed their evaluation and contribution to the model.

Figure 1 illustrates the *G-Risk Model* phases, which occur in parallel to the software life cycle. In the model, the phases are subdivided in activities and both are defined. For each set of phase/activity, the risks, divided in their respective classses, are related.

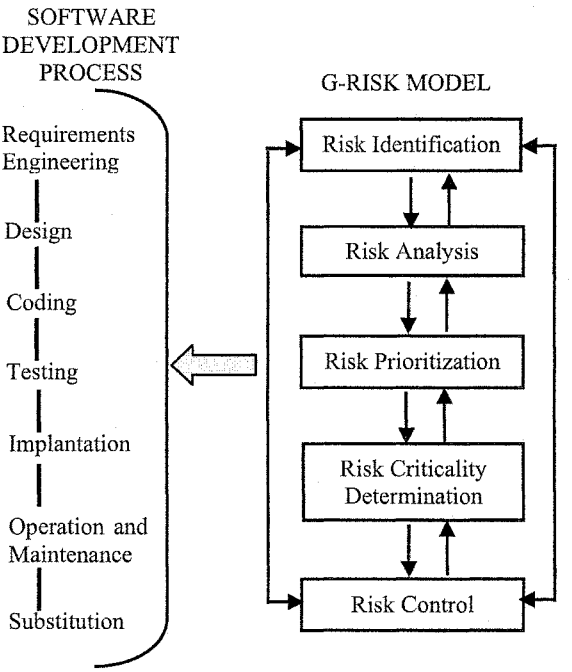

Fig. 1. Overview of the *GRisk-Model*

The classes of risk adopted in the model include:

- Relationship Risks (RR). They are risks that involve the interactions between developers and users, and between different types of users, concerning the definition of system functionality.
- Organizational Risks (OR). They are risks that involve organizational changes that affect the system under development as, for example, organization charts alterations, changes in the user's area, dismissal of professionals responsible for the system, etc.
- Management Risks (MR). They are risks that involve the management of the system development, such as: definition of development methodologies, definition of professionals who will compose the work team; definition of the necessary development environment; definition of resources for the development; etc.

- Financial Risks (FR). They are risks that cause financial expenses beyond the planned one, including high values of proposals, cost of equipments, etc.
- Technical Risks (TR). It is a broad class of risks, which can be caused by the professionals' lack of experience, use of inadequate methodologies and techniques, etc.
- Legal Risks (LR). They are risks related to laws, such as fiscal requirements, licenses for software, changes of tax laws during or after the system development, etc.

As part of the *GRisk-Model*, a list of probable risks, for each phase of the software development was defined [6]. These risks were identified with basis on bibliographical studies and also considering the experience and suggestions of the professional team who participated in the work.

So that the risks can be controlled and monitored, the impact that these risks will be able to cause in the project development and to the expected product must be determined. The degree of risk impact, may it be high, medium or low, will have to be analyzed, considering the probability of occurrence of the risk. The higher the probability of occurrence of the risk and its degree of impact, the greater is the control and monitoring it will have to receive [3].

4 Risk Management Tool

The Risk Management Tool (*GRisk-Tool*) has two objectives. The first objective is the creation of a knowledge base, with information obtained from the *GRisk-Model*, that will be used in the management of risks of future projects. The second objective is the compiling, follow-up and control of occurrence of risks, identified along the development of new projects. The compiling of the risks makes possible to keep the knowledge base updated, as well as to generate information to define metrics concerning to significant impacts for the identified risks.

Figure 2 gives an overview of the tool, which includes the following modules:

- Creation of knowledge base. In this module, the information, already classified in phases and activities of the software development, is loaded in the files.
- Control of risk management. In this module, the information of occurrences of risk identified in the software development process is registered.
- Monitoring and control of risks. This module makes available to the user the register of monitoring carried through a determined risk occurrence.
- Maintenance of knowledge base. In this module, the knowledge base is updated through the registering of risk occurrences.
- Reports and consultations. This module makes available to the user a series of reports and consultations related to the knowledge base contents, risk management, and occurrences about the risk monitoring.

The *GRisk-Tool* was evaluated by six software engineering professionals, including 1 manager of software factory, 3 project managers and 2 senior system analysts. Two of them had participated of the GRisk-Model definition and the other ones did not

know the model and the tool. These professionals were invited to participate of the evaluation, in function of their experience on software project management, specifically on risk management.

The tool works on personal computer environment, under Windows operational system (see [6], for a complete description of implementation issues). It was set free for use by the software development team of the software factories that participated of its development. It is being used together with the software development methodology, aiming at the optimization of the software production, in terms of deadlines, costs, and quality.

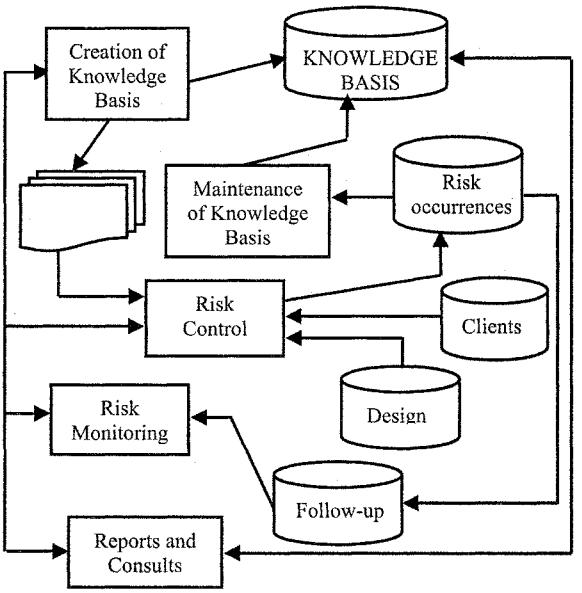

Fig. 2. Overview of the GRisk-Tool

5 Conclusion

This article presented a process model of risk management (*GRisk-Model*) and a tool (*GRisk-Tool*). The *GRisk-Model* was developed with basis on the literature, also getting the experience and effective participation of directors, managers and senior software engineers of Brazilian software factories. The *GRisk-Tool* implements the risk management model and the professional team involved in the definition of this model also evaluated it.

The *GRisk-Model* is characterized by incorporating a knowledge base concerning the software development process. The *GRisk-Tool* supports this model, as well as it offers conditions so that the knowledge base created can be continuously updated and extended with information of new risks related to projects monitored by the tool.

A very important result from the use of the *GRisk-Tool* is to detect potential problems in the software development phases, as well as the impacts and the costs related to such problems. It is possible to get the mapping of these problems from a script of actions to be taken, containing a set of information on different classes of risks related to all stages and activities of the software development, as is shown in the *GRisk-Model.*

As the impact of the risks is determined for each project, it is expected that, with the descriptions included in the tool, metrics can be obtained for determining standards to be applied to the risk impacts. Such metrics and standards have been gradually incorporated in the model and the tool.

It is also expected that the *GRisk-Tool* work as an additional mechanism to assist the software development in the company. The tool offers to the risk management a list of actions to be taken in all stages of software development, thus preventing that the responsible professionals need more accurate knowledge on the subject. With the dynamic updating of the knowledge base, this characteristic becomes an essential factor for the risk management and the success of the project being developed.

Now, some experiments are being conducted, focusing on the use of the model and of tool, in the development of new software projects, in the software factories. It is expected these experiments will provide important information for the improvement and extension of the *GRisk-Model* and the *GRisk-Tool.*

References

1. Boehm, B.W. "A Spiral Model of Software Development and Enhancement", *IEEE Computer*, Volume 21, Number 5, May 1988, pp. 61-72.
2. Boehm, B.W. *Tutorial: Software Risk Management*, IEEE Computer Society Press, New York, 1989.
3. Boehm, B.W. "Software Risk Management: Principles and Practices", *IEEE Software*, Volume 8, Number 1, January 1991, pp. 32-41.
4. Charette, R.N. *Software Engineering Risk Analysis and Management*, McGraw-Hill New York, 1989.
5. Charette, R.N. "Large-Scale Project Management is Risk Management", *IEEE Software*, Volume 13, Number 4, July 1996, pp. 110-117.
6. Gonçalves, E.L. *Risk Management in the Software Development Process,* Master Dissertation, Methodist University of Piracicaba, 2006 (in Portuguese).
7. Karolak, D.W. *Software Engineering Risk Manag*ement, Wiley-IEEE Computer Society Press, Los Alamitos, CA, 2002.
8. Lyu, M.R, Yu, J.S, Keromidas, E., Dalal, S. "ARMOR: Analyser for Reducing Module Operational Risk*", 25th Symposium on Fault-Tolerant Computing*, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 137-142.
9. Padayachee, K. "An Interpretative Study of Software Risk Management Perspectives", SAICSIT 2002, South Africa, 2002, pp. 118-127.
10. Pressman, R.S. *Software Engineering – A Practitioner's Approach*, 4th edition, McGraw-Hill, New York, 1997.