Roadmap for the Application of Risk Management Process in Software Projects in Accordance with ISO 31000:2009

Marcelo Nogueira*, Ricardo J. Machado+

* Software Engineering Research Group, University Paulista, UNIP, Campus of Tatuapé, São Paulo, Brasil
+ ALGORITMI Center, School of Engineering, University of Minho, Campus of Azurém, Guimarães, Portugal

Email: marcelo@noginfo.com.br, rmac@dsi.uminho.pt

Abstract

In an increasingly competitive global market, software development companies, under the pressure for conquering new market shares, subject themselves to business demands where the risks inherent to these operations are diversified and of exposure not always calculated. Given that a minority of such companies adopt risk management into their business processes, such exposure may affect the participation and success of these projects. To assure the quality of the software risk analyses and risk assessments are required. Among the uncertainties of software design, some risk factors should be treated: timeline, estimated costs and compliance to business requirements, among others, can be mentioned. Through a bibliographical review it was possible to produce a risk roadmap to provide to the professional in the field the treatment of risks in a friendly way. To contribute to these software projects, this work presents how the application of a risk management process, in order to insert the culture and capacity of professionals who work in such projects, can objectively target to the mitigation of the risks to which such projects are exposed. In addition, the adopted approach is in accordance to ISO 31000 standard.

Keywords: software engineering; risk management; software crisis; quality software, information systems.

1 Introduction

In a competitive environment of increasingly complex change, the appropriate management of information is crucial in the process of decision making in organizations (Nogueira, 2009).

Being this subject both comprehensive and specialized, the adoption of the practices of software engineering as a baseline of information management enables the development and consolidation of knowledge in the production of software.

These practices also prepare professionals to confidently face new challenges in the business world, strengthening their skills and abilities and keeping them up to date on the potential of information systems and new technologies in a globally competitive business perspective.

2 Software Engineering

According to Pressman (2006), software engineering is: The establishment and use of sound engineering principles to be able to obtain in economical way software that is reliable and works efficiently on real machines. It descends from systems and hardware engineering and encompasses a set of three basic elements (methods, tools and procedures) which allows the manager to control the software development process and offers the professional a basis for the construction of high-quality software.

For Martin (1991), is the study of principles and their application in the development and maintenance of software systems. Both software engineering and structured techniques are collections of methodologies of software and tools.

Software enginnering is a set of practices to the development of software solutions or, in other words, a guideline that may use several techniques. The sequence of prestablished steps allows the option for or the variation of tools and techniques along its diverse phases.

And it also can be defined as a set of methods, procedures and tools aimed at the production of software with quality, in other words, in accordance with customer requirements (Nogueira, 2009).

2.1 Objectives of Software Engineering

Software engineering has as primary objective the quality improvement of software products and the increase of the productivity of software engineers, in addition to meeting the requirements of efficiency and effectiveness (Maffeo, 1992).

Software engineering aims to systematize the production, maintenance, evolution and recovery of software products, so that the process occurs within time and cost estimates, with monitored progress and using principles, methods, technologies and processes in continuous improvement.

The adoption of an effective software development process in accordance with principles of software engineering can assure, by construction, products developed and maintained with satisfactory quality, adequate support for their users in performing their duties and satisfactory operation in real environments which may continuously evolve, adapting to a continuously evolving world (Fiorini, 1998).

Combining these goals, the term engineering is intended to indicate that software development should be subject to rules which are similar to those that govern the manufacture of industrial products in traditional engineering, as both are methodological (Maffeo, 1992).

Based on the goals of software engineering, it is possible to demonstrate the necessity of adopting a systematic model to standardize and manage the processes of software development, with the aim of pursuing quality in software processes and products.

2.2 Software Crisis

In the study of software engineering, author Roger S. Pressman (2006) mentions the "Software Crisis", where numbers are given that express the problem with non-completion of software projects. The same author points out that one of the main factors that cause such "Software Crisis" is the lack of adoption of methods, procedures and tools in building software.

The term "Software Crisis", which began to be used in the 60s, has historically alluded to a set of problems recurrently faced in the process of software development (construction, deployment and maintenance) (Maffeo, 1992). In general terms, the "Software Crisis" occurs when the software does not meet the customers, users, developers or enterprise needs and exceeds cost and time estimates (Nogueira, 2009).

Despite the enormous variety of problems that characterize the software crisis, in computer systems development field, engineers and project managers tend to focus their concerns on the following point: "There is huge uncertainty of estimates of timelines and development costs" (Nogueira, 2009).

Many of these errors could be avoided if organizations could have a software engineering process defined, controlled, measured and improved. However, it is clear that for many IT professionals these concepts are not very clear, which certainly hampers the action of managers in the improvement of their production processes (Blaschek, 2003).

There are several techniques, methodologies and quality standards to contribute to the development of software, including risk management. Professionals who do not embrace them find difficulties in performing software projects which are free of maintenance and re-work, so directly condemning the product quality.

Adoption of software engineering leads the individual to perform the activities related to their professional role through systematic methods throughout the software life cycle, allowing the developed product to represent the company actual processes and to meet in fact the company needs.

2.3 Software Life Cycle

Software life cycle is defined as the process that includes ideas from conception to the discontinuity of the software product (ISO / IEC 12207, 2008).

During the life cycle of software, multiple processes are performed, each of which contributes to achieving the goals of a stage of the cycle (Machado, 2002).

The life cycle models are categorized by the definition of a sequence of predefined activities, which aim to develop or maintain software (Pressman, 2006).

Several models were created and tested in the production of software, including the spiral model of Boehm (1988).

According to Sommerville (2007), although there is no "ideal" software process, there are many opportunities of working to improve it, in many organizations. The processes may include outdated techniques or may not take advantage of best practices of software engineering.

The improvement of software processes can be deployed in different ways. It can occur through the standardization of processes, as it is the first essential step in introducing new methods and techniques of software engineering (Sommerville, 2007).

Taxonomy for the life cycle of software is defined by the standard ISO / IEC 12207: 2008 - Systems and software engineering – software life cycle processes. This standard aims to define all the necessary guidelines to achieve quality in the process.

2.4 Software Quality

Achieving a high quality product or service is the goal of most organizations. It is no longer acceptable to deliver products with low quality and fix the problems and deficiencies after the products were delivered to the customer (Sommerville, 2007).

According to Machado (2001), for many software engineers, quality software process is as important as product quality. So in the 90s there was great concern in improving the software process. Important approaches to the standards ISO 9000 and ISO / IEC 12207, CMMI model (Capability Maturity Model Integration) and SPICE (Software Process Improvement and Capability dEtermination) suggests that the improvement of the software process can improve the quality of products.

Quality is a result of processes, people and technology. The relationship between product, quality and each of these factors is complex. Therefore, it is much harder to control the degree of product quality than to control the requirements (Paula Filho, 2009).

When producing software with quality, the real possibility of extracting relevant information from a system is created. This may not only contribute to the decision, but to be a factor of business excellence, enabling new business, retention and survival in an active market. Thus, it is of paramount importance to identify and analyze risks that threaten the success of the project and manage them so that the business objectives may be achieved.

Professionals who adjust their processes to produce quality software within reliable deadlines and budgets and have the ability to integrate, harmonize and accelerate the process of software development, will have the primacy of the market, when subjected to competition that forces the substantial reduction of time for the product being delivered (Machado, 2001).

Aiming at quality in the process of software production, risk management has the focus to address the uncertainties inherent to software projects, because many factors that involve technology, people and processes are in conflict and can determine whether the development of the software product will be successful or not.

3 Risks and Software Engineering

Risk, such as science, was born in the sixteenth century, during the Renaissance. In an attempt to understand the games of chance, Blaise Pascal, in 1654, discovered the "Theory of Probability" and created the "Pascal Triangle", which determines the likelihood of possible outcomes, given a certain number of attempts (Bernstein, 1997).

In the twentieth century, risk management was disseminated, studied and used mainly in the areas of health, finance, life insurance and so on. For these companies, risk management is not a bad thing, on the contrary, risk management is the business. All projects in these areas deal with risks, because profits depend on attractive opportunities, balanced by well calculated risks (Bernstein, 1997).

Risk in the software area was represented in a systematic manner by Barry Boehm in the 80s through the Spiral Model, which has as its principle be iterative and directed to risks, because for each iteration it is performed an analysis of risk (Boehm, 1988).

Each iteration of the spiral model is divided into four sectors:

- Definition of objectives;
- Risk assessment and risk reduction;
- Development and validation;
- Planning;

The choice of the spiral model to support this research, among various models of software life cycle, refers to its emphasis and its direct relationship with the risk analysis.

- The important distinction between the spiral model and other models of software process is the explicit consideration of risks (Sommerville, 2007).

Risks in software can not be mere agenda items. They should be the "heart" of the business, as in other areas (Chadbourne, 1999).

Currently, the area that addresses risks in software engineering has evolved from an analysis within the model of development, as proposed by spiral model, to become a management technique that should permeate all the processes of software life cycle.

Risk management is understood as a general procedure for resolution of risk, ie when it is applied in any instance, the possible consequences are all acceptable, and policies to cope with the worst outcome must be defined in the process.

Risk is presented in some way and to some degree in most human activities and is characterized by being partially known, changing over time and being manageable in the sense that a human action can be applied to change its shape and the extent of its effect. The process of risk management starts with uncertainties, worries, doubts and unknowns that become acceptable risks (Machado, 2002).

Risk management, in software design domain, is a defined and systematic process with the purpose of treating risk factors in order to mitigate or minimize its effects, producing a quality software product that meets customer needs, within estimated time and costs (Nogueira, 2009).

3.1 Work Motivation

According to Standish Group (2009), through a study called "Chaos Report", for projects in the area of information technology, the following conclusions were drawn (Table 1):

- 32% of projects finish on time and on budget;
- 44% of projects are challenged;
- 24% are canceled before its deployment.

Table 1: Chaos Report

Projects / Year	1994	1996	1998	2000	2002	2004	2006	2009
Successful	16	27	26	28	34	29	35	32
Contested	53	33	46	49	51	53	46	44
Cancelled	31	40	28	23	15	18	19	24
Failed	84	73	74	72	66	71	65	68

Source: Standish Group, 2009.

As for cost and schedule, the following information was obtained:

- Surplus in original estimated cost in 45% of the projects.
- Surplus in original schedule in 63% of the projects.

Other collected data are:

- 94% of the projects have at least one restart (Standish, 2009);
- 9% of projects in large companies come into operation within initially estimated cost and time.
- In software projects only 67% of originally proposed requirements are delivered in the end.

Standish Group has identified that, in 1995, United States invested U.S. $ 81 billion in software projects that were canceled before they even come into operation and $ 59 billion in projects that exceeded their initially estimated schedule.

Such a waste of time and money cannot be accepted in a world of increasingly scarce financial resources. Numbers of this magnitude can seriously compromise the competitive aspect of an organization and lead to its exit from the market.

A large volume of published data points to the risks that occur in software projects performed without the use of appropriate processes (Paula Filho, 2009).

A published survey from a database of 4,000 projects noted the frequent occurrence of the following problems:

- 70% of major applications projects suffer from instability of requirements. The requirements typically grow about 1% per month, reaching levels of over 25% of swelling at the end of the project.
- At least 50% of the projects are executed with productivity levels below normal.
- At least 25% of the shelf software and 50% of custom product have defects levels higher than reasonable.
- Products made under pressure of deadlines can quadruple the number of defects.
- At least 50% of large software projects exceeded its budget and its deadline.
- 2/3 of very large software projects are canceled before the end.
- Users are not satisfied with 25% of commercial products for personal computers, 30% of commercial products for mainframe computers and 40% of products made to order.
- Typically, 50% of the software assets of the companies are not used.
- Dissension between the area of information technology and top management occur in more than 30% of organizations.
- Dissension with customers in application development occurs in 50% of service contracts and in 65% of performance contracts.

Despite the "Software Crisis" is not a new problem, even nowadays its impact and its negative effects are faced. The scarce use of methodologies and models of quality in Brazil indicates that this reality has to be modified.

According to the Ministry of Science and Technology (2002), only 11.8% of companies in Brazil have adopted risk management in software projects.

Due to the relevance of the theme and its direct impact on the success in producing software, the number presented by the ministry is alarming because the sample used for the research included both the major software companies and the small and medium enterprises in the country (Nogueira, 2009).

The concerning fact is that small and medium enterprises, which hold 65.1% of the software market in Brazil (MCT, 2002), lack a culture of risk management. Besides contributing to the possibility of failure in current projects, this situation undermines the still promising future opportunities that this sector needs to explore in both domestic and foreign markets.

New research in 2005 and 2008 were made by the Ministry of Science and Technology, but the item risk management was not added to the survey.

3.2 Risk Engineering

According to Robert Charette (1989), the definition of risk is:

First, risk affects future events. Present and past are irrelevant, because what is reaped today was planted by our previous actions. The issue is changing our actions today. Can opportunity be created for a different and possibly better situation tomorrow?

Secondly, this means that risk involves change, such as change of thought, opinion, action or places. Thirdly, risk involves choice and the uncertainty that choice entails itself.

Thus, paradoxically, the risk, like death and taxes, is one of the few certainties of life.

Peter Drucker (1975) once said “Since it is futile to try to eliminate risks and questionable to try to minimize them, it is essential that the considered risks be the right ones”. Before the "right risks" which will take place during a software project may be identified, it is important to identify all the others that are obvious, both for managers and for professionals (Pressman, 2006).

According Higuera (1995), a 100% likely risk is a restriction on the software project.

In a simplified way, a risk can be thought as a probability that some adverse circumstance will really occur. The risks may threaten the project, the software being developed or the organization.

Risk categories can be defined as follows (Sommerville, 2007):

- Risks related to the project: risks that affect the schedule or the resources of the project;
- Risks related to product: risks that affect the quality or performance of software that is under construction;
- Risks for business: risks affecting the organization that is building or buying the software.

Risk engineering involves two key areas in the process: risk analysis and risk management (Peters, 2001).

According to this author, risk analysis consists of three processes:

- Identification of risks
- Risk estimate
- Risk assessment

And risk management is composed of five processes:

- Risk planning;
- Risk control;
- Risk monitoring;
- Risk targeting;
- Recruitment (Answers to risks).

When the risk is considered in the context of software engineering, three conceptual foundations are always in evidence (Pressman, 2006):

- The future is our concern: Which risks may cause the failure of the software project?
- Change is our concern: How changes in customer requirements affect the timeliness and overall success?
- We take care of choices: Which methods and tools shall we use, how many people should be involved, how much emphasis on quality is enough?

Project risk can be estimated quantitatively or qualitatively. The main objective of risk analysis is to develop a set of strategies for risk prevention (IEEE, 1995). Risks should be assessed and monitored (Paula Filho, 2009).

Risk estimate is a very important and rarely practiced task. Good planning is not just a prediction of what should happen if everything goes well, but also what can go wrong, what the consequences of the problems and what can be done to address them. Among the risk factors that should be considered, it may be included:

- Legal risks;
- Technological risks;
- Risks due to the size and complexity of the product;
- Risks relating to personnel involved in the project;
- Risks relating to the acceptance by users.

Sommerville (2007) has described the types of risks that may affect the project and the organizational environment in which software is being built. However, many risks are considered universal and they include the following areas: Technology, personnel, organizational, tools, requirements and estimation.

The estimation of risks involves the following tasks:

- Identification of possible risks to the project;
- Analysis of these risks, evaluating their probability and likely impact;
- Prediction of corrective or preventive countermeasures;
- Prioritization of risks, organizing them according to likelihood and impact.

Risks do not remain constant during the execution of a project. Some disappear, new ones arise, and others suffer changes of probability and impact, therefore changing the priority. Therefore a monitoring report of the project along with an updated table shall be used for monitoring the risks. The estimation table should be reviewed and updated to reflect the modifications until the risks are realized or completely eliminated (Paula Filho, 2009).

The following questions were derived from risk data obtained by a survey of experienced software project managers in different parts of the world (Keil, 1998). The questions are sorted by their relative importance in relation to project success:

1. Have the top management of the software company and of the customer formally committed themselves to support the project?
2. Are the end users enthusiastically engaged with the project?
3. Are the requirements fully understood?
4. Have the customers been fully involved in specifying requirements?
5. Do end users have realistic expectations?
6. Is the project scope stable?
7. Does the project team have the appropriate mix of skills?
8. Are the project requirements stable?
9. Does the project team have experience with the technology to be built?
10. Is the number of staff appropriate to the project?
11. Do all team members and users involved in the project agree on the importance of the project and the system requirements?

If any of these questions is answered negatively, the steps of mitigation, monitoring and management should be instituted immediately. The degree to which the project is at risk is directly proportional to the number of negative answers to these questions.

The adoption of risk engineering is part of the critical success factors in software projects. The management of risks throughout the life cycle of development is critical to project success (Nogueira, 2009).

3.3 Risk Management

Nowadays there are great expectations about the applications of information technology in companies, thanks to the new business alternatives they provide and the improvements that they bring to existing processes. However, there is a questioning of the real gains arising from the massive investments in technology, because often the return is less than expected (Nogueira, 2008).

In this context, IT governance and risk management emerge as key issues for companies and managers, as the biggest risks to which organizations are exposed today are related to information technology (Nogueira, 2008).

IT Governance is defined as a set of processes and controls that direct the information technology strategy, ensuring that it can support the business strategies and business objectives (Nogueira, 2008).

For most organizations, information and technology that support business represent its most valuable assets. In an environment increasingly competitive and dynamic, management requires quality, functionality and ease of use of IT resources, as well as high availability at lower costs.

There is no doubt about the benefits of using technology. However, to be successful, an organization must adopt a management model that enables the effectiveness and efficiency of IT. In this context, risk management on software projects comes as one of the key processes, enabling the representation of best practice, and adhering to IT governance as it relates to control, transparency and monitoring (Nogueira, 2008).

Risk management is particularly important for software projects, due to the inherent uncertainties that most projects face (Sommerville, 2007).

Risk management consists of coordinated activities to direct an organization in relation to risk. Risk management generally includes assessment, treatment, acceptance and communication of risks (MCT, 2002).

The risk management process involves several stages or activities (Sommerville, 2007):

1. Risk identification: It identifies potential project risks, product and business;
2. Risk analysis: the possibilities and the consequences of such risks are evaluated;
3. Risk planning: plans are outlined to address the risks, either by avoiding them, or minimizing its effects on the project;
4. Risk monitoring: The risk is constantly evaluated and plans for reducing the risk are revised as more information about them becomes available.

Project managers of information systems should regularly assess the risks during the development process to minimize the chances of failure. In particular, the problems of schedule, budget and functionality of the software can not be totally eliminated but they can be controlled through the implementation of preventive actions (Higuera, 1996).

Risk management has six well-defined activities that are: Risk identification, risk analysis, risk planning, risk monitoring, risk control and risk communication (Higuera, 1996).

The activities of risk identification and risk analysis, critical risk assessment, risk mitigation and contingency plans should be made. The methods of risk assessment should be used to demonstrate and

evaluate the risks. Constraint policies of the project must also be determined at the time when discussions with all others involved take place. Aspects inherent to risks of software, such as the tendency of professionals to add features that are difficult to measure or even the risks of intangible nature of software, should influence the risk management of project (SWEBOK, 2004).

The Orange Book (2004), originally developed by the British government, now an international reference handbook, details the guidelines for good risk management, involving the following activities: Identifying risks, assessing risks, risk appetite, addressing risks, reviewing and reporting risks, communication and learning.

The ISO 31000 (2009) standard directs the policy for risk management with the following activities: establishing the context, risk identification, risk analysis, risk assessment, risk treatment, communication and consultation and monitoring and review.

4 Risk Management Process

After the literature review, it was possible to identify critical areas in the process of software development. However, to support risk management in software projects, it is necessary to use a workflow with activities where the decision maker can use it as an auxiliary instrument in the process of risk management.

Therefore, the following activities make up this workflow: Communication and consultation; establishing the context; risk identification; risk analysis; risk evaluation; risk treatment and monitoring and review.

These activities are described below according to the complexity of application in accordance with ISO 31000.

4.1 Communication and Consultation

Communication and consultation with external and internal stakeholders should take place during all stages of the risk management process. It should take place in the beginning, with the first meeting of sensitization, during activities and in the end, with the presentation of results.

4.2 Establishing the Context

By establishing the context, the organization articulates its objectives and defines the external and internal parameters to be taken into account when managing risk, and sets the scope and risk criteria for the remaining process.

4.3 Risk Identification

The organization should identify sources of risk, areas of impacts, events (including changes in circumstances) and their causes and their potential consequences. The aim of this step is to generate a comprehensive list of risks based on those events that might create, enhance, prevent, degrade, accelerate or delay the achievement of objectives.

It is important to identify the risks associated to not pursuing an opportunity. Comprehensive identification is critical, because a risk that is not identified at this stage will not be included in further analysis. It is recommended to use a universal framework with risks common to different designs when it is the first iteration.

4.4 Risk Analysis

Risk analysis involves developing an understanding of the risk. Risk analysis provides an input to risk evaluation and to decisions on whether risks need to be treated, and on the most appropriate risk treatment strategies and methods.

Risk analysis can also provide an input into making decisions where choices must be made and the options involve different types and levels of risk. A framework can be used with the universal risk weights established from expert opinion, especially when you do not have a knowledge base.

4.5 Risk Evaluation

The purpose of risk evaluation is to assist in making decisions, based on the outcomes of risk analysis. It defines which risks need treatment and the priority for treatment implementation.

4.6 Risk Treatment

Risk treatment involves selecting one or more options for modifying risks, and implementing those options. Once implemented, the provision of treatments or modification of controls must be performed.

4.7 Monitoring and Review

Both monitoring and review should be a planned part of the risk management process and involve regular checking or surveillance. It can be periodic or ad hoc.

5 Conclusion

In this bibliographical review, it was found that the authors recognize the difficulties in the process of software development. Despite the proposal of forms and procedures in order to facilitate understanding as well as to disseminate the use of these methods, their application does not always happen, as it is acknowledged that no method is foolproof, because of the complexity of the problems to be solved in building an information system. However, given the many uncertainties inherent to software projects, it was identified the need to prepare a roadmap to the activities of risk management with the goal of creating culture and spreading it, since the formal applicability of them in business is still very low. With ISO 31000, the scope can be defined and validated. As future work, the application of the roadmap here depicted on projects that never used the risk management is intended. From the positive and negative results so gathered, a proposition for adoption of risk management as a common practice will be presented.

References

ISO 31000. (2009). ISO 31000: *Risk management – Principles and guidelines*: ISO.
ISO 12207. (2008). NBR ISO/IEC 12207: *TI: Processo de ciclo de vida de software*. Rio de Janeiro: ABNT.
Bernstein, Peter. (1997). *Desafio aos deuses: a fascinante história do risco*. Rio de janeiro: Campus.
Blaschek, J. R. (2003). *O principal problema dos projetos de software*. Rio de Janeiro.
Boehm, Barry. (1988). *A spiral model of software development and enhancement*: IEEE Computer.
Chadbourne, B. C. (1999). *To the heart of risk management: teaching project teams to combat risk*: Pennsylvania.
Charette, R. N. (1989). *Software Engineering risk analysis and management*: McGraw Hill.
Drucker, P. (1975). apud Pressman, Roger S. (2002) *Engenharia de Software*. Rio de Janeiro: McGraw Hill.
Fiorini, Soeli T. (1998). *Engenharia de Software com CMM*. Rio de Janeiro: Brasport.
Higuera, R. P.; Haimes, Y. Y. (1996) *Software risk management technical report*: CMU/SEI 96 TR 012. SEI.
IEEE 1044.1 (1995). *IEEE Standard Glossary of Software Engineering Terminology*: IEEE.
Keil, M., et al. (1998). *A Framework for identifying software project risks*: CACM, vol.41 nº 11.
Machado, C. A. F.(2002). *A-Risk*. Pontifícia Universidade Católica do Paraná, Curitiba: PUC-PR
Machado, C. A. F. (2001) *NBR ISO/IEC 12207: Processos de ciclo de vida do software*: Prentice Hall.
Maffeo, Bruno. (1992). *Engenharia de Software e Especificação de Sistemas*. Rio de Janeiro: Campus.
Martin, James. (1991). *Engenharia da Informação*. Rio de Janeiro: Campus.
MCT. (2002) *Qualidade e Produtividade do Software Brasileiro*. Brasília: MCT - Secretaria de Política de Informática.
Nogueira, M. (2008). *Normas e modelos de qualidade como política de produção de software*: XV SIMPEP - UNESP.
Nogueira, M. (2009). *Engenharia de Software. Um Framework para a Gestão de Riscos*, Rio de Janeiro: Ciência Moderna.

Orange Book (2004). *Management of Risk – Principles and Concepts*, HM Treasury, Crown, London.
Paula Filho. (2009). *Engenharia de Software: fundamentos, métodos e padrões*. Rio de Janeiro: LTC.
Peters, James F.(2001). *Engenharia de Software: Teoria e prática*. Rio de Janeiro: Elsevier.
Pressman, R. S. (2006). *Engenharia de Software*. 6. ed. São Paulo: McGraw-Hill.
Sommerville, I. (2007). *Engenharia de Software*. 8. Ed. São Paulo: Pearson Addison Wesley.
Standish. (2009). *CHAOS Summary 1995..2009*, Boston: Standish Group.
SWEBOK. (2004). *Guide to the software engineering body of knowledge*. USA: IEEE Computer Society.