Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 80 (2007) 16–31
The Journal of Systems and Software
www.elsevier.com/locate/jss

Evaluating software project portfolio risks

Hélio R. Costa [a,b], Marcio de O. Barros [b,c], Guilherme H. Travassos [b,*]

[a] CCA-RJ, Brazilian Air Force, Ponta do Galeão s/nº, Ilha do Governador, CEP: 21941-510 Rio de Janeiro, Brazil
[b] COPPE/UFRJ, System Engineering and Computer Science Department, Caixa Postal: 68511, CEP: 21945-970 Rio de Janeiro, Brazil
[c] UNIRIOTEC, Applied Computer Science Department, Av. Pasteur 458, Urca, CEP: 22290-240 Rio de Janeiro, Brazil

Received 1 October 2005; received in revised form 9 March 2006; accepted 26 March 2006
Available online 22 May 2006

Abstract

As in any other business, software development organizations try to maximize their profits and minimize their risks. The risks represent uncertain events and conditions that may prevent enterprises from attaining their goals, turning risk management into a major concern, not only for project managers but also for executive officers involved with strategic objectives. In this sense, economical concepts can greatly support Software Engineers in the effort to better quantify the uncertainties of either a single project or even a project portfolio.

In this paper, we present a technique for evaluating risk levels in software projects through analogies with economic concepts. This technique allows a manager to estimate the probability distribution of earnings and losses incurred by an organization in relation to its software project portfolio. This approach has been calibrated by data collected in an empirical study, which has been planned and accomplished to provide information about the relative importance of risk factors in software projects. A usage example of such an approach is presented. Finally, we introduce a case tool specially built to support the application of the proposed techniques.

Keywords: Software project management; Software risk management; Software Engineering Economics; Empirical studies

1. Introduction

The development of software systems can be analyzed in the same way as many other investment activities: there must be justification for using the available resources in the project and such justification usually depends on expected returns. In software projects, resources are represented by the development team and hardware and software facilities that are required throughout the project, while returns are the profits expected to be earned by the development organization or the benefits provided by the system.

Many authors have introduced the notion that software development is a value-driven activity. Sullivan et al. (1999) present a roadmap for researchers emphasizing the need for a strategic investment approach to Software Engineering. Favaro et al. (1998) presents a real option-based approach addressing the value of software reuse. Boehm et al. (2000) defines Software Economics as the field that seeks improvements in Software Design and Engineering through economic reasoning on product, process, program, portfolio, and policy issues.

Basically, Software Engineering Economics approximates concepts from economic sciences or corporate finance theories to the software development context, supporting managers and investors who work in the software industry to make better decisions about their projects, increasing profits or minimizing losses.

In this paper, we present a risk evaluating technique for software projects based on an economic view of the elements that constitute risk factors for software projects. In order to support this technique, an empirical study was conducted to assess the relative importance of different risk factors for software development projects. As an application of the results obtained, we present an approach to

* Corresponding author. Tel.: +55 21 2562 8712; fax: +55 21 2562 8676.
E-mail address: ght@cos.ufrj.br (G.H. Travassos).

0164-1212/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2006.03.038

calculate the probability distribution of losses and earnings that can be attained by a software development organization from its software project portfolio. This methodology is inspired on credit risk theory for loan operations, which is extensively used by financial institutions.

The paper is organized into five sections and two appendixes. Section 1 comprises this introduction. In Section 2, we present a technique for evaluating project risk levels. In Section 3, we describe an empirical study conducted during this research, aiming to highlight the relative importance of risk factors in software projects. In Section 4, an approach to establish the probability distribution of losses and earnings generated by a software project portfolio, together with an example, are set out. Finally, in Section 5, we draw some conclusions and trace future prospects for this research. Appendix A describes the questions used to support risk level assessment. Appendix B details the results of the *t*-test application in our empirical study.

2. Calculating project risk levels

Risk management has gained importance in software project management in the last few years. It is well accepted that the uncertainties faced by software projects should be taken into account when planning and controlling the development of software systems.

We define *project risk level* as the probability of a project's failure in achieving its proposed goals. The risk level summarizes, in a single number, how risky a project is. For instance, if a project has a risk level of 30%, it has 70% of chance of being successful. The risk level helps managers to compare two or more projects based on their risk-and-return[1] ratios. Furthermore, by quantifying the risks and highlighting which project aspects may be more risk-prone, managers can better identify where to apply their limited resources in an attempt to reach projects goals.

To estimate a software project risk level, we took advantage of an analogy with Economics. Many economic analyses classify the risks organizations face into two categories: systemic and specific. Systemic risks influence a large number of organizations within a certain economic sector, segment or country. On the other hand, specific risks are inherent to an organization, focusing the aspects that make it distinct from other organizations in the same economic environment (Ross et al., 1996). Examples of systemic risks are governmental policies on markets, the international economic scenario and wars. Specific risks include local market conditions, company policies and practices, business objectives and so on. While systemic risks are useful for analyzing how external issues may affect an organization, specific risks look for internal factors that can affect its performance.

[1] The risk-and-return ratio is often used instead of ROI (return on investment) to evaluate an organization's willingness to develop a project. Unlike ROI, which only considers how much the organization will earn from the project, the risk-and-return ratio measures how much risk the organization will incur while attempting to earn profits from the project.

We applied this classification to software projects by defining *systemic software risks* as issues that may affect the performance of software projects pertaining to a given category (information systems, military systems, off-the-shelf components, etc.). Also, we define *specific software risks* as particular characteristics of a given project that may decrease its chance to be successful. Thus, any software project in a given category should observe the systemic risks that affect its category in the course of planning and controlling activities. For instance, Information Systems projects may be affected by requirements volatility, planning inadequacies and testing coverage (systemic risks). However, the difficulty in implementing a given algorithm or the lack of appropriate hardware to conduct tests may not be relevant issues to every software project in a given category, but may be relevant for a particular one (specific risks).

Based on this classification, we created a technique to evaluate a software project's risk level based on a risk identification questionnaire composed of two abstraction levels (questions and risk factors). The questionnaire has 211 questions classified according to 10 groups (risk factors). Each question pertains to a single risk factor and addresses a software project characteristic related to it. The questions help managers evaluate risk factors, relating them to practical elements that can be observed in the project's characteristics. This structure enables them to address the project's specific risks by answering the questions. The manager can subsequently aggregate project exposure to the major factors that represent its systemic risks.

To develop this questionnaire, we started from the 194 questions proposed by Carr et al. (1993), which are classified in accordance with 13 factors. Furthermore, we added questions to address issues highlighted in other approaches presented in the literature (Jones, 1994; Karolak, 1996; Barki et al., 1995; Boehm, 1991; Moynihan, 1997; Wallace et al., 1999) but absent from the original questionnaire. Finally, we reduced the number of factors (by merging them and redistributing questions) to facilitate the paired comparison used in our empirical study (see Section 3). The number of questions currently associated to each factor is presented in Table 1.

Table 1
Risk factors and their questions

Factor	# Questions
Analysis	28
Design	17
Coding	11
Testing	25
Planning	36
Control	17
Team	32
Policies and structure	08
Contracts	21
Clients	16

The resulting questionnaire is shown in Appendix A. A brief description of each risk factor is presented below, highlighting the issues addressed by their specific questions:

- *Analysis*: requirement elicitation, volatility, implementation, complexity and validation.
- *Design*: software architecture, interfaces, algorithms and coding related mechanisms.
- *Coding*: system implementation, programming languages and code reuse.
- *Testing*: planning, execution and testing coverage.
- *Planning*: manager experience, project estimation, planning, definition of development process, adaptation and usage.
- *Control*: project execution, approval of intermediate artifacts, conflict resolution, team support, planning adjustment and process evaluation.
- *Team*: team stability, training, capability, maturity, as well as the development environment and the degree to which the team follows the organization's development process.
- *Policies and structure*: level of support from stakeholders, their goals and conflicts.
- *Contracts*: outsourcing, suppliers, and external dependencies.
- *Clients*: client involvement, number of users, and the impact of the project on clients.

A project manager should answer the questionnaire by assigning a number from 0 to 5 to each question. This coding operation is based on Likert's research (1932), where numbers are used to express values on an ordinal scale. The higher the manager's perception of the impact of the issue described in the question on a given project, the higher the answer should be. For instance, the question "Are the requirements stable?" might be answered with zero if requirements do not change frequently. On the other hand, the same question might be answered with five if requirements are highly volatile. Finally, some questions may not be relevant to the project under consideration. Such questions should be marked by the manager as not relevant (NR). When applied in an industrial context, the questionnaire is distributed in a sequential electronic form (within the *RISICARE* tool—described in Section 4.2) that allows the manager to select one of the possible answers for each question.

After completing the questionnaire, the manager calculates the median value for the answers given to the questions pertaining to each risk factor. The median is used instead of an average value because the answers are given in ordinal scale. Questions marked as not relevant are treated as missing values and not taken into account when calculating the median.

The manager then maps the median calculated for each risk factor to the [0–1] interval. This operation requires a mapping function that converts the ordinal scale value (median) to the interval scale (normalized median). The mapping function might be monotonic, that is, the greater the median, the greater the scaled median. A simple mapping function divides the original median calculated for each factor by 5. Since the maximum value for the median is 5, the division is a monotonic operation that maps the median to the required interval. The normalized median represents a project's exposure to risks related to a given risk factor, that is, the project's specific risks for that factor.

In the following step, the normalized median is multiplied by a "weight". The weight of the risk factor is a number in the [0–1] interval scale that represents the systemic risk, i.e., the degree to which a factor contributes to project failure in a given category.

In the last step, the project's specific and systemic risks are combined into a single number by summing the weighted normalized medians for all factors. Table 2 presents an example of the project risk level calculation process, where only two factors have been taken into account. The first factor includes three questions and has a weight of 70%. The second one includes only two questions and a weight of 30%.

A process to determine the weights of the risk factor is described in Section 3. However, some properties for these numbers should be stated: ($P1$) the sum of all weights of the risk factors must be 1 (100%), and ($P2$) the higher a factor's relevance, the higher its weight should be. The first property normalizes the project risk level, allowing it to assume any value between 0% and 100%. This is granted since both the weight of the risk factor and normalized median pertain to the unitary interval. The second property adjusts the specific risk assessment (question answers) to the systemic risks (factors).

We recognize that the sum of the weighted normalized medians is not a precise value for the probability of project failure, but it can be used to compare different projects as far as their risk exposure is concerned. In Section 4, we describe an application of the present technique to support the selection of a limited set of projects from a larger range of available ones. In such an application, project risk level is not required to be an absolute measure of project failure

Table 2
Calculating a project's risk level

Answer the questionnaire	Factor 1			Factor 2	
	Q1	Q2	Q3	Q1	Q2
	1	4	3	2	4
Calculate the median of the answers	Median (1, 4, 3) = 3			Median (2, 4) = 3	
Weighted normalized median	$\frac{3}{5} * 70\% = 42\%$			$\frac{3}{5} * 30\% = 18\%$	
Sum of weighted values	Risk level = 60%				

probability, but a relative measure allowing a manager to compare the chances two or more projects might have of failing.

Regarding the weights of the risk factors, it could be argued that they may vary from project to project. We cannot refute this argument, but assume that within a system category, this difference can be attributed to specific risks. By this logic, a highly weighted factor that is not relevant to a project will have its weighted normalized median reduced if the answers to questions relevant to it produce low values.

3. An empirical study for evaluating the weights of the risk factors

Due to the many different types of software projects that can be undertaken in an increasing number of domains, it is supposed that the weights of the risk factors required by the risk level assessment technique presented in Section 2 can vary dramatically across different system categories. In order to determine these values for a particular category, we planned and executed an empirical study.

In order to reduce this research's scope and increase its precision, the first study was limited to evaluating the weights of the risk factors for Information Systems projects. In this section, we present the study summary and suggest its application to other domains and system categories. The study was planned in accordance with the methodology proposed by Wohlin et al. (2000).

3.1. Objective

To determine the weights that should be assigned to each risk factor in order to reflect their contribution to the measurement of a project's risk level (presented in Section 2). The weights portray how critical a risk factor is for a specific system category. In this study, criticality is defined as the degree to which a risk factor contributes towards the failure of a project.

3.2. Subjects

The study was performed with the assistance of subjects taken from industry and academia, represented by professionals, professors and graduates, all of whom have experience in software project development. The methods adopted to select them were *Quota Sampling* and *Convenience Sampling*. They were thus drawn from different groups of the target population (software developers), but not randomly.

They duly agreed to participate and signed a consent form regarding the study. They were asked about the relative importance of the risk factors concerning the selected system category (in the present study, Information Systems). Table 3 describes the characterization of all subjects.

Table 3
Subjects' characterization

Academic background			
Ph.D.	M.Sc.	MBA	Graduate
07	15	19	09
Project function			
Managers		Analysts	
34		16	
Years working in software development (average)		*Number of projects developed (average)*	
16		12	
Current occupation			
Academic researcher/ Professor		Industry professional	
11		39	
Number of organizations visited (Rio de Janeiro, Brazil)			
26			

3.3. Project size

While analyzing the factors, subjects were asked to keep in mind a specific project size, choosing one that closely reflects their own experience. The project size chosen by each participant was characterized by the effort required for its development (measured in man–months) and registered on a subject characterization form. The information provided by them was used to group the projects by size (small, medium or large).

To form the project groups, the values of the project sizes (man–months) given by the subjects have been plotted in bar charts, so that a cut-off point (visual observation) could be established in order to create the three initial groups. The standard deviation for each group was then calculated. Finally, we interactively divided the groups around the initial cut-off points, including or removing the projects closest to these points in each group, so that we could minimize the standard deviation of each group. In doing so, we have defined small projects as those requiring less than 80 man–months. Medium projects were those between 80 and 250 man–months, and large projects were those greater than 250 man–months.

These project sizes were used in data analysis for grouping participant opinions, so that different weights of the risk factors could be identified for different project sizes. Our goal was to observe whether there were any differences between the weights for the risk factors in relation to project sizes. For instance, subjects could say that the weight for the *Testing* risk factor in a small project could be different from a large project, and that different values should thus be applied to the risk quantification technique, depending on size.

3.4. Grouping

It was expected that subjects' experience and project size could influence the research results. We, therefore, decided to block subjects and projects before analyzing the data.

Subjects were classified into three groups according to their experience: low, moderate or high. The grouping process was based on a characterization form filled by each participant before evaluating the risk factors. The form ascertained the academic and industrial background information about each participant, such as the number of years working in software development, number of projects developed, academic qualifications and degree of experience in risk management. This information allowed us to summarize the participants' expertise in a single figure and group them using the approach proposed by Farias et al. (2003) adapted for our purposes. The formula used to summarize a participant's experience was

$$W(i) = \frac{Y(i)}{\text{Median } Y} + \frac{P(i)}{\text{Median } P} + A(i) + R(i),$$

where W is the weight of the participant, Y the number of years working with software development and P the number of projects developed. The values used to describe their academic background (A) in Computer Science (CS) were:

- $A(\text{i}) = 0$, for no degree;
- $A(\text{i}) = 1$, for CS degree;
- $A(\text{i}) = 2$, for CS MBA;
- $A(\text{i}) = 3$, for CS M.Sc. and
- $A(\text{i}) = 4$, for CS Ph.D.;

The values used to describe their experience in risk management (R) were

- $R(\text{i}) = 0$, for no experience;
- $R(\text{i}) = 1$, for little experience;
- $R(\text{i}) = 2$, for moderate experience;
- $R(\text{i}) = 3$, for highly experienced, and;
- $R(\text{i}) = 4$, for experts.

After calculating the participants' weights, they were grouped according to their experience. The grouping procedure allowed us to have a common weight for opinions given by subjects of the same group. To determine the most suitable cut-off points for classifying participant experience as little, moderate or high, the same procedure initially applied to group projects by size was also applied to participant experience data. The average weight of the three groups was calculated and a weight of 1.0 was assigned for those with little experience. The weights of the other groups were determined by dividing their average weight value by the original average weight value calculated for the group with little experience subjects. Table 4 presents the number of subjects and the average and final weights calculated for each group. For instance, opinions given by individuals with moderate experience have a weight 1.61 times greater than opinions of individuals pertaining to the group with little experienced.

Table 4
Participants' count per group and weights

	Little exp	Moderate exp	Highly exp
# Participants	13	28	9
Average weight	4.09	6.62	10.78
Weights	1.0	1.61	2.63

3.5. Instrumentation

A subjective evaluation method was adopted to ascertain each participant's experience. Of the methods available, the one chosen for this study was paired comparison. In this method, the objects of interest (in our case the risk factors) are placed in a matrix, so that each cell represents a comparison between a pair of objects. The participant analysis each pair of objects and determines which one should receive preference. In our study, the subjects were asked to subjectively determine which risk factor was more important than the other, i.e., which one most contributed to project failure. With all comparisons completed, it was possible to determine which factor was considered most relevant for each participant and so on, successively, until the least relevant factor.

The main advantage of pair comparison is to minimize evaluation complexity and increase the precision of the results. Another point worth highlighting is that the human mind can more easily establish differences than assess absolute values. Finally, by comparing one object with another, the participant is forced to make a decision about the relation between the two (Miranda et al., 1999). The disadvantage of this method is that a greater number of comparisons is required in the case of many objects: this value is to the order of n^2, n being the number of objects.

The instrument designed to support the paired comparisons was based on the MACBETH tool in its demonstration version[2] (Bana e Costa and Vansnick, 1995). This tool allowed the subjects to formalize their preferences in a semantic way. It mapped the votes they gave onto an interval scale representing the relative importance among risk factors. According to this mapping, each risk factor was given a weight percentage, so that the sum of all the weights of the factors would be 100%. This mapping complies with the two specified properties for the weights of the factors ($P1$ and $P2$) set out in Section 2. The MACBETH tool also checks for inconsistencies in the votes and helps resolve comparison conflicts, generating coherence without influencing or restricting the participants' freedom to express their opinion.

A typical result given by the tool after comparing the ten risk factors is described in Table 5.

[2] Even being a demonstration version, the tool has proven reliable and flexible enough to support the study.

Table 5
A participant's opinion on the weights of the risk factors

Risk factor	#
Clients	14.36
Control	13.86
Planning	3.51
Analysis	12.61
Team	11.11
Testing	10.81
Policies/structure	9.01
Design	8.40
Coding	6.00
Contracts	0.33

Table 6
Number of rejected values per risk factor

Risk factor	#
Analysis	04
Design	02
Coding	01
Testing	00
Planning	03
Control	01
Team	02
Contracts	01
Policies/structure	00
Clients	02

3.6. Outlier elimination

Having collected all the participants' opinions, we analyzed the data generated by the tool. All the votes that differed from the average value by two or more standard deviations were marked as outliers. From the original 500 votes (50 subjects voting for each of the 10 risk factors), only 16 were rejected, as seen in Table 6.

3.7. Data analysis

We have submitted the weights of the risk factors to a *t*-test with a significance level of 0.05 (5%) in order to evaluate the differences between the averages of these values for each project group (according to project size). Such analysis was intended to verify if the weights of the risk factors were independent of project size, that is, if the same weight could be used to determine a project's risk level, regardless of its size. The *t*-test was, therefore, applied to the weights of each risk factor for all project sizes (small, medium, and large). We used the *t*-test instead of ANOVA because we wanted to compare each project group with the remaining ones. In such pair comparison, the *t*-test is a special case of ANOVA.

No significant difference was observed for the analysis, design, coding, testing, planning, and contracts factors for different project sizes. However, the remaining factors—team, control, policies and structure and clients—have shown small, but significant difference when small projects were compared to large ones. Even for these factors, no significant differences were found when small projects were compared to medium ones or when medium projects were compared to large ones.

Table 7
Weights of the risk factors

Factors	Small (%)	Medium (%)	Large (%)
Analysis	12.36	12.57	10.78
Design	8.59	7.52	6.23
Coding	4.84	4.13	4.00
Testing	7.36	6.17	5.82
Planning	15.26	13.04	13.85
Control	10.39	11.64	12.19
Team	11.28	11.53	12.63
Contracts	3.40	5.37	4.60
Policies/structure	11.41	12.47	14.11
Clients	15.11	15.57	15.79

Therefore, we assumed that the project risk level measure presented in Section 2 should be calculated with different weights according to project size. The weights obtained in the study after all the statistical analysis and normalization processes are presented in Table 7. Details of the *t*-test results can be found in Appendix B.

Besides contributing to assess project failure, these risk factors can be used to improve the software project management plan. For instance, managers could use this information to make adjustments to the software process or reduce the influence of such issues depending on the project size. They can also support decisions regarding the project's risk mitigation.

We acknowledge that all risk factors are important, but some may be more important than others. Table 7 highlights these differences, ranking factors according to the general opinion of developers on their relative importance to Information System projects depending on their size. The values presented in this table can be used to calculate the project risk level, weighting the answers of the questionnaire completed by the project manager. However, it should be noted that current experimental constraints limit the generalization of such weights for other system categories and developer populations. Moreover, organizational factors, like the maturity of the development process, expertise in a specific domain, and availability of reusable components, among others, may also restrain the generalization of the identified weights. Such detailed segmentation was outside the scope of our current empirical study. However, the identified weights can be used as a starting point for IS projects and further studies can follow the proposed plan and be executed to provide more precise values in specific contexts.

3.8. Internal validity

Subjects were selected by Quota Sampling and Convenience Sampling, based on experience in software project development. The subjects agreed to take part in the study. The number who voted for a certain project size was

random and no dropout was observed, due to the nature of the study. The usage of the MACBETH tool supporting the trial to produce the results was considered positive according to a follow-up (qualitative) questionnaire completed by the subjects after using the tool in the study.

3.9. External validity

Due to the subjects' characteristics and their selection process, it can be said they are representative of developers' population. The quantification of the risk factors was based uniquely on the experience of subjects who had the opportunity to operate the tool in the time and place they judged adequate. The interviewer helped them use the tool, but was not allowed to influence their assessment. The weights for risk factors can be considered valid as a starting point for evaluating the risks related to Information System projects and for project sizes as defined in this study. The number of subjects in the present study restricts our ability to generalize upon the results observed, but does not diminish their reliability. It could be interesting to replicate the study considering a different, large population and aggregate the results to see how they might evolve.

3.10. Construction validity

The factors, as much as the questions used in the study, were based on instruments presented in technical literature. We relied on these information sources to consider them valid for the study. A pilot experiment was conducted prior to the main trial study in order to test the plan, the effectiveness of the MACBETH tool and to improve the study. The difference between the risk factors and the characteristics of the questions pertaining to each one was explained to the subjects. The possibility of guessing at the result was eliminated by the use of paired comparisons and the MACBETH tool, where the votes were checked for consistency. The percentages of the weights of the risk factors were computed automatically, without interference from neither participant nor researchers. The software resolved conflicting answers by indicating inconsistent votes to the participants, requiring their correction and thus leading to a more reliable result.

3.11. Conclusion validity

The *t*-test performed with a significant level of 0.05 (5%) led to reliable observation of the weights of the risk factors obtained in the study. The same instruments were presented to all subjects and the implementation of the study can, therefore, be considered reliable. Elimination of outliers reduced the possibility of data misinterpretation and blocking the subjects and projects in different groups minimized the heterogeneity of the elements. The population size restricts the use of the results usage in other types of software projects portfolios.

4. Application of the proposed risk level measure

Managers can use the risk level measure approach presented in Section 2 (calibrated for their specific software project categories[3]) in a variety of decision-making activities. In this section, we propose an application of such information to measure how much money a software development organization can lose or earn from several ongoing software projects, assuming that some of them may fail due to risk-related problems incurred during the development process.

To address this problem, we took advantage of a financial analogy: a set of software projects is analyzed as a portfolio of long-term loans. Credit-giving financial institutions have particular interest in determining the probability of losing or earning a certain amount of money while maintaining a loan portfolio. Such information is used to limit an organization's losses due to *default* events, which occur when a borrower fails to pay back a loan acquired from a financial institution.

Credit analysis is based on assigning a default probability for each borrower and determining the amount of money expected to be lost from a loan portfolio. This analysis also requires specifying expected returns on investment for each loan, since profits from borrowers that pay back their loans may compensate losses generated by unpaid debts.

Software projects, like loans, have uncertain probabilities of success, represented by their risk level. A *default* event for a software project may be associated to its failure, regardless of the causes. Software projects also involve an amount of money in their development and negotiation processes. To some extent, money will be lost if a software project fails and profits will be attained if it reaches its goals. Software development organizations should, therefore, analyze their project portfolio in order to determine the losses and earnings probabilities and maintain a profitable business even after a (limited) number of project failures. As in the loans domain, each project is expected to generate profit for the development organization. Such profit should, to some extent, compensate for losses incurred by projects that fail.

Based on this analogy, in the section below we present a technique of loan transaction related credit risk concepts that can be applied in the context of software project assessment.

4.1. Credit risk modeling

Credit risk can be defined as the probability of a borrower organization or counterpart failing to meet its obligations in accordance with the terms agreed in a given deal. Its measurement allows an enterprise to maintain its risk exposure to each business partner at an acceptable

[3] Section 3 described the results regarding Information Systems weights of risk factors.

level. It also allows us to estimate the capital that an organization has to set aside in order to sustain losses derived from unsuccessful dealings.

The main approaches for credit risk analysis presented in financial literature are the KMV, Credit Metrics, and CreditRisk+ models (Barbanson et al., 2004). Each model requires a distinct set of parameters and is based on particular assumptions. The KMV approach depends on the existence of a liquid market for trading equities for the counterpart under analysis. Since there is no counterpart in our software project analogy (project risk is due to uncertain aspects of the development process or product), the KMV method is unsuitable.

The Credit Metrics technique requires the existence of a secondary market for trading loans and protection assets based on these loans, such as credit swaps and default options (Barbanson et al., 2004). Again, since there is no tradable asset to protect software projects from failure due to development-related aspects (such as contract liabilities), the Credit Metrics approach is also not directly applicable.

The CreditRisk+ approach does not need a secondary trading market and the only information required from the underlying counterpart is its default probability for each loan contract. So, its application to the analysis of a software project portfolio is more straightforward.

The CreditRisk+ model is a suite of three distinct models, each built on the previous one, thus, adding details and complexity to its calculation process. The first CreditRisk+ model is based on an analytical formula that calculates the distribution probability of the loan portfolio's losses and earnings, each loan described by an amount lent to the borrower, the expected profits to be gained if the borrower pays back the loan and the inherent risks of the whole negotiation. The model assumes that all money given in a loan transaction will be lost in case of a *default* event.[4] Borrowers are usually grouped in accordance with their ability to make the payments and each group is assigned with a *default* probability.

Classification into groups is possible due to the large number of borrowers and historical data available on the financial market. The second and third CreditRisk+ models also take into account the variability of *default* probabilities and the distribution of debtors among economic sectors. In order to establish the capital required to maintain a loan portfolio, the CreditRisk+ model uses a Poisson distribution (Johnson and Bhattacharyya, 1977) to describe the number of *default* events that may occur with loans made to the borrowers of each group. In accordance with the average number of *defaults* for each group, the expected profits and the sum invested in each loan, the model estimates the expected return on the whole portfolio, its variability and its sensitivity to the concentration of borrowers from each group.

[4] Some models can work with recovery rates, which represent a percentage of the loan that is expected to be paid back by the borrower even in case of a default event.

The approach for software projects presented in this paper is based on the first CreditRisk+ model, which deals with fixed *default* probabilities, without taking into account the uncertainty of such probabilities. Mapping this model to software projects, the proposed technique assumes that each project has a fixed probability to be canceled, represented by the project risk level and calculated as presented in Section 2.

Mapping credit-related concepts to software project portfolios; we observe that is not usual for an organization to maintain such a large number of projects concurrently, inhibiting the creation of project groups. So, instead of determining a *default* probability for each project group (to be used in the Poisson distribution), we calculated a *default* probability for each project using a Bernoulli distribution (Aragão et al., 2003) to describe its behavior.

The Bernoulli distribution is the simplest probability distribution function, receiving a single parameter, a probability p and yielding two possible results, either one or zero, the first being as frequent as stated by p.

By changing the probability distribution function, the CreditRisk+ analytical formula (based on the Poisson distribution) no longer holds. We, therefore, propose using the Monte Carlo simulation to estimate the losses and earnings probability distribution for a software project portfolio based on the approach proposed by Aragão et al. (2003).

In the CreditRisk+ model, some assumptions are made that must be considered in order to determine the limits of our approach. For instance, the model presumes that all costs are spent when the loan is given and that all returns are received when the loan is paid back. This is not true for all software projects, because some of them may have a cash-flow forecast (expenses and payments) programmed to last throughout the project's life-cycle (intermediary cost during project execution, partial payments and so on). All the same, we assumed the premises proposed by the model in order to apply it in a software development context. Table 8 presents a comparison between the financial and software project assumptions.

Assuming these premises and by means of simulations, it is possible to create the probability distribution of losses and earnings for any number of projects in an organization's portfolio, based only on their probability of failure (risk level), their costs and their projected returns. The probability distribution is calculated as follows.

The simulator generates a Bernoulli random value for each project in the portfolio, and this is compared to the project risk level. Such a number indicates whether the project might succeed or fail in that simulation;

- If the project fails, the money invested in its development is lost.
- If the project succeeds, profits attained from it are calculated as the difference between its cost and the sum received from the client.

Table 8
Financial and software assumptions

Financial	Software
Inherent risks to each loan	Inherent risks to each software project
State of debtor: defaulter or not	State of project: canceled or not
Sum of loan is agreed upon before operation is dealt	Project's costs are estimated prior to starting
The payment of the loan is made at the end of the term	The payment of the project is made upon its completion
In case of a default event, the loss has a fixed value (the money agreed upon for the loan) not considering the recovery rate	In case of project failure, the loss has a fixed value (project cost) regardless of the expected return
There is no correlation between default events, except for external factors that can affect debtors in a similar manner	There is no correlation between the project failures, except for external risks that can affect them in a similar manner

- At the end of each simulation, the resulting values (cost or profit) for all projects are added up. This figure represents the total amount of money lost or earned by the organization from the development of such projects.
- The simulation process is repeated several times to obtain a better assessment of the expected returns of the projects portfolio, along with its variations. At least 10,000 cycles[5] are recommended.
- After completing all simulations, losses or earnings are grouped and mapped on the frequency domain. The grouping process depends on predefined criteria, usually as simple as dividing the space between the better earnings and the worst losses into a fixed number of bars. The frequency mapping is performed by counting the number of losses and earnings generated by the simulation process that fits each group.
- Finally, the percentage of losses and earnings in each group is calculated from the total number of simulation cycles and these values are plotted onto a chart, usually in a cumulative manner.

Let us consider the projects in Table 9. Each one has its own risk level, calculated by using the technique proposed in Section 2, the costs planned for its development and its expected return. All values are fictitious and presented here only as an example.

After applying the Monte Carlo simulation ten thousand times to our example, the cumulative probability distribution of the losses and earnings that can be attained by the organization is represented in Fig. 1.

These results set out in Fig. 1 are expressed as cumulative probability distributions, that is, the vertical bars represent the sum of the probability of the previous bars plus the probability of the value under scrutiny. As can be seen, the probability of losing money with this portfolio is almost 20%, represented by the cumulative probabilities of negative values. We can also observe that the probability of earning \$28K or more is about 67%, while the chance of attaining \$80K (maximum possible profit) is 36% (100–67%, the cumulative probability of obtaining values lower than 80K). If such scenario is not affordable for the organization, several options can be considered:

[5] In general, our simulations were stable after running about ten thousand cycles.

Table 9
Project with risk levels and costs

Project	Risk level (%)	Cost	Return
#1	10	\$10K	\$25K
#2	20	\$20K	\$35K
#3	15	\$30K	\$45K
#4	25	\$40K	\$50K
#5	20	\$50K	\$75K

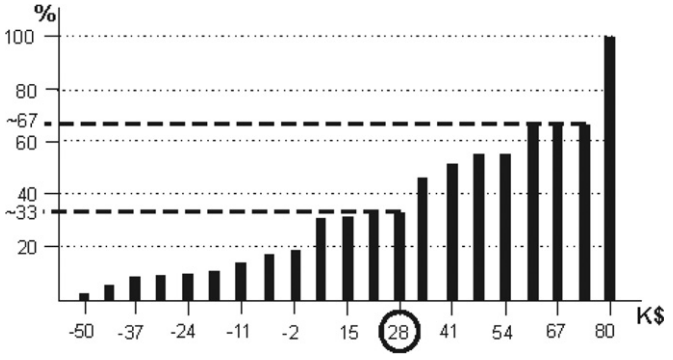

Fig. 1. Probability distribution.

- Minimizing the costs of one or more projects.
- Improving the return on one or more projects.
- Canceling one or more projects or
- Performing a contention or contingency plan to reduce risk levels.

Let us suppose the enterprise has decided to carry out a combination of these alternatives, changing the projects parameters to the values shown in Table 10.

Redoing the simulation, we obtain a distribution such as presented in Fig. 2.

Table 10
Project with risk levels, costs and returns changed

Project	Risk level (%)	Cost	Return
#1	10	\$10K	\$30K
#2	18	\$22K	\$35K
#3	15	\$25K	\$45K
#4	20	\$45K	\$60K
#5	15	\$55K	\$80K

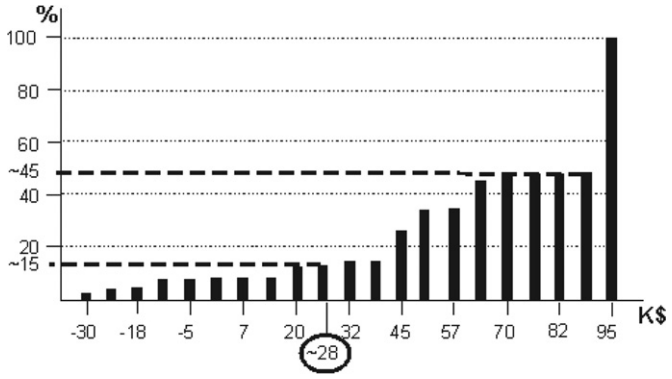

Fig. 2. Probability distribution changed.

In this scenario, the probability of losing money (negative values) in this portfolio is around 6%, while in the previous scenario it had been around 20%. The chance of earning around \$28K or more has improved to 85%. Furthermore, the maximum possible profit has increased to \$95K, whilst the chance of attaining this value is about 55%.

Many different strategies can be tested by managers in order to ascertain what the best composition for the project portfolio is, and subsequently better application of the enterprise's resources. It should be noted that the approach is best used by enterprises with many projects in their portfolio, because it permits a wide variation of the parameters used in the simulation and a better scenario analysis.

It can also be said that if an enterprise has, in its portfolio, a large number of projects to the extent that it is possible to create groups with enough projects to generate a probability distribution, the original formula of the CreditRisk+ model and the Poisson distribution could be used, as proposed by the model, and similar results would be obtained.

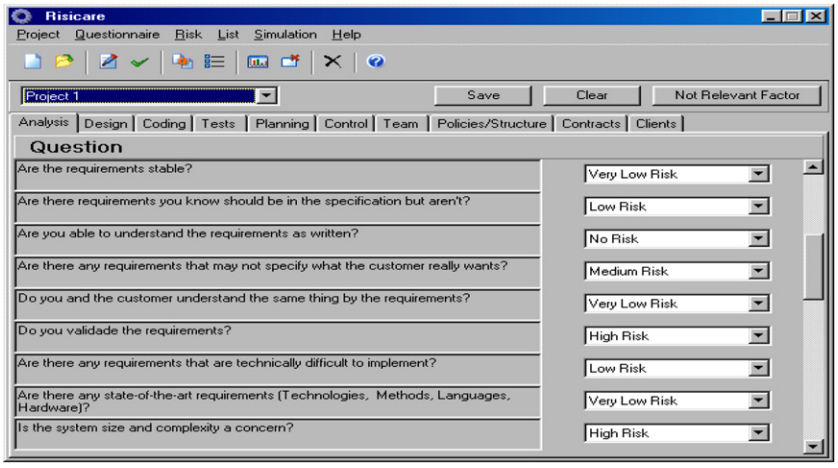

Fig. 3. *RISICARE* questionnaire screen.

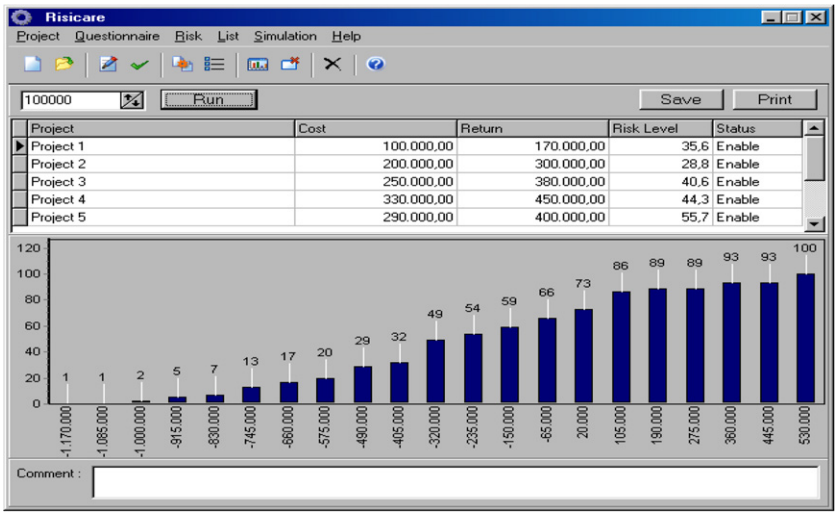

Fig. 4. *RISICARE* simulation screen.

4.2. The RISICARE tool

Due to the scope of the risk level assessment questionnaire and the different weights that can be attributed to each project (according to each one's specific characteristics), the calculation of a project risk level and the simulations required to create the losses and earnings probability distribution for a project portfolio cannot be performed manually. Thus, the *RISICARE* tool was conceived and implemented to undertake these activities. Figs. 3 and 4, respectively, show the questionnaire and the simulation screens of the *RISICARE* tool. It is composed of five major modules:

- *Project characterization*: acquires information such as project name, team size, project duration, expected costs and profits.
- *Questionnaire*: allows the manager to complete the risk level assessment questionnaire for a given project (see Section 2 and Appendix I).
- *Risk level*: used to calculate the risk level of any project previously characterized.
- *Projects list*: allows the user to transfer project information to the simulation module.
- *Simulation*: performs simulations with selected projects and generates the losses and earnings' probability distribution.

5. Conclusions and future prospects

In this paper, we have presented a technique for assessing the risk level of a software project based on its systemic and specific risks. By means of analogies with economic concepts, we introduced an approach to estimate the probability distribution of earnings and losses incurred by an organization according to its software project portfolio. This approach is supported by data collected in an empirical study performed to obtain a better understanding of the relative importance of risk factors regarding Information System projects. Furthermore, a CASE tool that supports the application of the proposed techniques has been developed and made available to the software engineering community.[6] We, therefore, believe that a first cycle to develop such technology has been accomplished, and includes a proof of concept discussed in Section 4.

We acknowledge that there are some limitations in this proposal. The first one is that, to enhance its reliability, the empirical study was restricted to collecting data regarding Information Systems. The second issue concerns the inability of the CreditRisk+ model to account for expenses and earnings (cash flow) incurred by the organization during project execution.

[6] Available at http://www.uniriotec.br/~marcio.barros/risks.

However, based on the results obtained so far, we suggest that the proposed technique (supported by the *RISICARE* tool) could be used by software development organizations to evaluate their software project portfolio. To support such a claim, the technique has been made available in the context of the Software Development Environment currently used by several Brazilian industrial organizations (Santos et al., 2005). Our intention is to evaluate its feasibility in this organizational context to improve software process maturity and management capabilities, making the results available in the near future.

As for future prospects, besides the feasibility study previously mentioned, we would include the replication of the empirical study to other types of systems using a larger population. Such studies would expand the applicability of our approach and verify the feasibility of its generalization to other system categories. We also expect to expand some of the CreditRisk+ concepts to encompass a software project's cash-flow structure.

Acknowledgements

The authors would like to thank all the subjects that took part in the empirical study and Margi Moss for helping us in the text reviewing. We acknowledge the support of the Brazilian Air Force and CAPES. Prof. Travassos is supported by the CNPq—Brazilian Research Council.

Appendix A. Risk questionnaire

Listed below are the questions, grouped into risk factors that compose the risk identification questionnaire used to calculate a project's risk level.

A.1. Analysis

1. Are the requirements stable?
2. Are there requirements you know that should be in the specification but are not there?
3. Are you able to understand the requirements as they have been written?
4. Are there any requirements that may not specify what the customer really wants?
5. Do you and the customer understand the same thing by reading the requirements?
6. Do you validate the requirements?
7. Are there any requirements that are technically difficult to implement?
8. Are there any state-of-the-art requirements (Technologies, Methods, Languages, Hardware)?
9. Are the system size and complexity concerns for the project?
10. Does the size require a larger organization than usual for your company?
11. Are reliability requirements allocated to the software?

12. Are availability requirements allocated to the software?
13. Are safety requirements allocated to the software?
14. Will it be difficult to verify satisfaction of safety requirements?
15. Are there unprecedented or state-of-the-art security requirements?
16. Have you implemented this level of security before?
17. Are there human factor requirements? Do you see any difficulty in meeting them?
18. Is the software requirements specification adequate for designing the system?
19. Are there any problems with performance (Real-time response, Response time, Database response, Contention or Access)?
20. Do the external interfaces change?
21. Are the external interfaces completely defined?
22. Is there a requirements traceability mechanism that tracks requirements from source specification through to test cases?
23. Does the traceability mechanism used in evaluating requirements change impact analyses?
24. Is there a formal change control process?
25. Are changes at any level mapped up to the system level and down through the testing level?
26. Is there adequate analysis when new requirements are added to the system?
27. Are there automated tools used for modeling?
28. Are there automated tool for requirements traceability?

A.2. Design

1. Are there any specified algorithms that may not satisfy the requirements?
2. Are there mechanisms for determining the feasibility of algorithms and design (Prototyping, Modeling, Analysis and Simulation)?
3. Does any of the design depend on unrealistic or optimistic assumptions?
4. Are there any requirements or functions that are difficult to design?
5. Are the internal interfaces well defined?
6. Is there a process for defining internal interfaces?
7. Is hardware being developed in parallel with software?
8. Has a performance analysis been done?
9. Does the design include features to aid testing?
10. Does the hardware limit your ability to meet any requirements?
11. Are you reusing or re-engineering software not developed on the project?
12. Are there any problems with using COTS (commercial off-the-shelf) software?
13. Do you foresee any problem with integrating COTS software updates or revisions?
14. Are there design standards?
15. Is it possible to reuse models or standards?
16. Is it possible to reuse interfaces?
17. Are there automated tools used for designing?

A.3. Coding

1. Are any parts of the product implementation not completely defined by the design specification?
2. Are the selected algorithms and designs easy to implement?
3. Are the design specifications in sufficient detail to write the code?
4. Does the design change whilst coding is being done?
5. Are there system constraints that make the code difficult to write?
6. Is the language suitable for producing the software on this project?
7. Are there multiple languages used on the project?
8. Is the development computer the same as the target computer?
9. Are the hardware specifications adequate to code the software?
10. Do the hardware specifications change whilst the source code is being written?
11. Are the source codes reusable?

A.4. Testing

1. Is there a formal testing plan?
2. Are the test specifications adequate to test the system fully?
3. Will compromises be made regarding testing if there are schedule problems?
4. Have acceptance criteria been agreed to for all requirements?
5. Are there any requirements that will be difficult to test?
6. Do the testers get involved in analyzing requirements?
7. Are the testing plans and procedures updated as part of the change process?
8. Do you begin unit testing before you verify source code with respect to the design?
9. Has sufficient unit testing been specified?
10. Is there sufficient time to perform all unit testing that you think necessary?
11. Has sufficient product integration been specified?
12. Has adequate time been allocated for product integration and testing?
13. Has sufficient system integration been specified?
14. Has adequate time been allocated for system integration and testing?
15. Will the product be integrated into an existing system?
16. Will sufficient hardware be available to accomplish adequate integration and testing?
17. Is there any problem with developing realistic scenarios and test data to demonstrate any requirements?

18. Are you able to verify performance in your facility?
19. Does hardware and software instrumentation facilitate testing?
20. Will the target hardware be available when needed?
21. Are all contractors part of the integration team?
22. Does the hardware limit your ability to meet any requirements?
23. Is regression testing performed?
24. Are there automatic tools for regression testing?
25. Are the tests performed by different people from the development team?

A.5. Planning

1. Is there a formal management methodology?
2. Is the project managed according to the plan?
3. Is re-planning done when disruptions occur?
4. Are the necessary plans being done?
5. Is there a detailed work breakdown structure for all activities?
6. Are there defined metrics for the project?
7. Are there tools for collecting and analyzing metrics from the project?
8. Are the project milestones well established?
9. Are there tools for project management?
10. Is there an effective risk management process?
11. Are there contingency plans for known risks?
12. Does the project have experienced managers?
13. Is the software quality assurance function adequately staffed on this project?
14. Do you have defined mechanisms for assuring quality?
15. Does schedule get in the way of quality?
16. Do you have an adequate configuration management system?
17. Is the configuration management function adequately staffed?
18. Has the schedule been stable?
19. Is the schedule realistic?
20. Is the estimation method based on historical data?
21. Has the method worked well in the past?
22. Is there anything for which adequate schedule was not planned?
23. Is the budget stable?
24. Is the budget based on a realistic estimate?
25. Is the estimation method based on historical data?
26. Has the method worked well in the past?
27. Is there anything for which adequate budget was not allocated?
28. Is there a well-defined development process?
29. Can you measure whether the development process is meeting your productivity and quality goals?
30. Is more than one development model being used?
31. Are there formal, controlled plans for all development activities?
32. Is the development process adequate for this product?
33. Is the development process supported by a compatible set of procedures, methods, and tools?
34. Does the development system support all aspects of the project?
35. Is there good documentation of the development system?
36. Is the development system considered reliable?

A.6. Control

1. Are there periodic structured status reports?
2. Does appropriate information get reported to the right organizational levels?
3. Do you track progress versus plan?
4. Are project members at all levels aware of their status versus plan?
5. Have features or functions been deleted as part of a design-to-cost effort?
6. Do budget changes accompany requirement changes?
7. Do schedule changes accompany requirement changes?
8. Are there external dependencies which are likely to impact the schedule?
9. Are there dependencies on external products or services that may affect the product, budget or schedule?
10. Does management communicate problems up and down the line?
11. Are conflicts with the customer documented and resolved in a timely manner?
12. Does management involve appropriate project members in meetings with the customer?
13. Does management work to ensure that all customer factions are represented in decisions regarding functionality and operation?
14. Are there good policies to present an optimistic picture to the customer or senior management?
15. Do you have a way to track interfaces?
16. Are the deviations to original plans corrected?
17. Are defect data collected during software development?

A.7. Team

1. Do people get trained in skills required for this project?
2. Do people get assigned to the project who do not match the experience profile for your work area?
3. Is there any problem keeping the people you need?
4. Are there any areas in which the required technical skills are lacking?
5. Is the staffing stable?
6. Do you have access to the right people when you need them?
7. Have the project members implemented systems of this type?
8. Is the project reliant on a few key people?
9. Is there sufficient capacity for overlapping phases, such as coding, integration and test?

10. Are the people trained in the use of the development tools?
11. Do people work cooperatively across functional boundaries?
12. Do people work effectively towards common goals?
13. Is it easy for project members to get management action?
14. Do people feel it's important to follow the plan?
15. Does management consult with people before making decisions that affect their work?
16. Is management intervention sometimes required to get people working together?
17. Is there good communication among the members of the project?
18. Are the managers receptive to communication from project staff?
19. Does the team feel free to ask for the managers' help?
20. Do the project members get timely notification of events that may affect their work?
21. Is the morale on the project high?
22. Are the development facilities adequate?
23. Are there enough workstations and processing capacity for all staff?
24. Are all staff levels oriented toward quality procedures?
25. Is risk management mentality part of the team culture?
26. Do people understand their own and others' roles in the project?
27. Do people know who has authority for what?
28. Are developers familiar with the plans?
29. Does everyone follow the development process?
30. Are people comfortable with the development process?
31. Is the team size a risk for the project?
32. Does the team know software engineering?

A.8. Contracts

1. Does the type of contract represent a risk for the project?
2. Is the contract burdensome in any aspect of the project?
3. Is the required documentation burdensome?
4. Are there problems with data rights?
5. Is the customer approval cycle timely (Documentation, Project reviews, Formal reviews)?
6. Do the external interfaces change without adequate notification, coordination or formal change procedures?
7. Is there an adequate transition plan in case of associate contractors?
8. Is it supported by all contractors and site personnel?
9. Is there any problem with getting schedules or interface data from associate contractors?
10. Are there any ambiguities in the subcontractor task definitions?
11. Is the subcontractor reporting and monitoring procedure different from the project's reporting requirements?
12. Is subcontractor administration and technical management done by a separate organization?
13. Are you highly dependent on subcontractor expertise in any areas?
14. Is subcontractor knowledge being transferred to the company?
15. Is there any problem with getting schedules or interface data from subcontractors?
16. Are your task definitions from the prime contractor ambiguous (If project is a subcontract)?
17. Is there any problem with getting schedules or interface data from the prime contractor?
18. Do you interface with two separate prime organizations for administration and technical management?
19. Are you highly dependent on the prime contractor for expertise in any areas?
20. Are you relying on vendors for deliveries of critical components (Compilers, Hardware, COTS)?
21. Is the number of external vendors a problem for the project?

A.9. Policies and structure

1. Are policies affecting the project?
2. Are policies affecting technical decisions?
3. Does project management communicate problems to senior management?
4. Does corporate management give you timely support in solving your problems?
5. Is the organizational structure stable?
6. Is the top management involved with the project?
7. Are there internal conflicts that affect the project?
8. Are the project goals well defined?

A.10. Clients

1. Are clients and users involved with the project?
2. Are users averse to changes?
3. Is the number of changes in the users' life caused by the system a concern?
4. Is the number of users affected by the system a concern?
5. Will the system change the organizational structure of the client?
6. Does the team know the client's organizational culture?
7. Does the team know the client's business?
8. Do you ever proceed before receiving customer approval?
9. Does the customer understand the technical aspects of the system?
10. Does the customer understand software?
11. Does the customer interfere with process or people?

12. Does management work with the customer to reach mutually agreeable decisions in a timely manner?
13. How effective are your mechanisms for reaching agreements with the customer?
14. Are all customer factions involved in reaching agreements?
15. Is it a formally defined process?
16. Does management present a realistic or optimistic picture to the customer?

Appendix B. *t*-Test results

As stated in Section 3, we have compared the observed weights in a pair-wise manner according to project size. So, weights estimated for small projects were compared to weights for medium projects. Weights for small projects were compared to large ones and, finally, weights for medium projects were compared to large projects. For each comparison, we have used a *t*-test with 5% significance level to verify if the observed values were statistically different.

The null hypothesis states that there are no significant differences in the weights of risk factors regarding project size. In the following tables, we present the values resulting from the *t*-test (T0) that were compared to the *t*-distribution (TD) for each risk factor and project size, aiming to accept or refute this hypothesis. As it can be observed, the null hypothesis was accepted in any comparison for six factors: analysis, design, coding, testing, planning, and contracts. Thus, these weights are statistically identical regardless the size of the project.

However, the null hypothesis was not observed in the four remaining factors (control, team, policies/structure, and clients) when large projects were compared to small ones. This implies that the weights of these risk factors are statistically distinct for small and large software projects. We intend to address the reasons why medium projects do not differ from small and large ones in further studies.

Since the weights for some risk factors were proven statistically different for distinct project sizes, it was decided to adopt a conservative posture and use different weights for all risk factors in all project sizes in the proposed risk level assessment technique.

Project size	T0	TD
I. *Analysis*		
Small × medium	−0.812	2.045
Small × large	−1.506	2.052
Medium × large	−0.385	2.042
II. *Design*		
Small × medium	−0.002	2.040
Small × large	−0.127	2.042
Medium × large	−0.116	2.045
III. *Coding*		
Small × medium	0.097	2.037
Small × large	−0.574	2.042
Medium × large	−0.638	2.042
IV. *Testing*		
Small × medium	0.211	2.035
Small × large	−0.487	2.042
Medium × large	−0.649	2.040
V. *Planning*		
Small × medium	0.174	2.040
Small × large	−1.561	2.052
Medium × large	−1.693	2.042
VI. *Control*		
Small × medium	−1.339	2.037
Small × large	**−2.553**	**2.045**
Medium × large	−1.642	2.040
VII. *Team*		
Small × medium	−0.902	2.040
Small × large	**−2.317**	**2.048**
Medium × large	−1.761	2.040
VIII. *Contracts*		
Small × medium	−1.146	2.037
Small × large	−1.724	2.042
Medium × large	−0.130	2.042
IX. *Policies and structure*		
Small × medium	−0.981	2.035
Small × large	**−2.513**	**2.042**
Medium × large	−1.780	2.040
X. *Clients*		
Small × medium	−1.101	2.040
Small × large	**−2.459**	**2.045**
Medium × large	−1.512	2.042

References

Aragão, C.S.L., Carvalho, L.E.Z.L., Barros, M.O., 2003. A Monte Carlo-based credit risk analysis technique. Resenha BM&F (152) (in Portuguese).

Bana e Costa, C.A., Vansnick, J.C., 1995. A theoretical framework for measuring attractiveness by a categorical based evaluation technique (MACBETH). In: Clímaco, J. (Ed.), Multicriteria Analysis. Springer, Berlin.

Barbanson, R., et al., 2004. An Introduction to Credit Risk with a Link to Insurance, AFIR, Working Group.

Barki, H., Rivard, S., Talbot, J., 1995. Toward an assessment of software development risk. Journal of Management Information Systems 10 (2), 203–225.

Boehm, B.W., 1991. Software risk management: principles and practices. IEEE Software 8 (1), 32–41.

Boehm, B.W., Sullivan, K., 2000. Software Economics: A Roadmap, in the Future of Software Engineering, 22nd International Conference on Software Engineering, June, 2000.

Carr, M.J., Konda, S.L., Monarch, I., Ulrich, F.C., Walker, C.F., 1993. Taxonomy-based risk identification, Technical Report CMU/SEI–93-

TR-6, Software Engineering Institute, Carnegie Mellon University, EUA, July, 1993.

Farias, L.L., Travassos, G.H., Rocha, A.R.C., 2003. Managing organizational risk knowledge. Journal of Universal Computer Science 9 (7), 670–681.

Favaro, J.M., Favaro, K.R., Favaro, P.F., 1998. Value based software reuse investment. Annals of Software Engineering 5, 5–52.

Johnson, R., Bhattacharyya, G., 1977. Statistical Concepts and Methods. John Wiley & Sons, New York.

Jones, C., 1994. Assessment and Control of Software Risks. Yourdon Press Computing Series, New Jersey.

Karolak, D.W., 1996. Software Engineering Risk Management. IEEE, Computer Society Press, Los Alamitos, CA.

Likert, R., 1932. A technique for the measurement of attitudes. Arquives of Psycology.

Miranda, E., 1999. Establishing Software Size Using the Paired Comparisons Method. Ericsson Research, Canada.

Moynihan, T., 1997. How experienced project managers assess risk. IEEE Software 14 (3), 35–41.

Ross, S., Westerfield, R., Jordan, B., 1996. Fundamentals of Corporate Finance. Times Mirror Professional Publishing.

Santos, G., Villela, K., Montoni, M., Rocha, A.R., Travassos, G.H., Figueiredo, S., Mafra, S., Albuquerque, A., Paret, B.D., Amaral, M., 2005. Knowledge Management in a Software Development Environment to support Software Processes Deployment, Springer LNAI v. 3782, Professional Knowledge Management Conference, Kaiserslautern, Germany, April, 2005.

Sullivan, K., Chalasani, P., Jha, S., Sazawal, V., 1999. Software design as an investment activity: a real options perspective, in real options and business strategy: application to decision making, L. Trigeorgis, Consulting Editor, Risk Books.

Wallace, L., 1999. The development of an instrument to measure software project risk. PhD thesis. College of Business Administration, Georgia State University, Georgia.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A., 2000. Experimentation in Software Engineering—An Introduction. Kluwer Academic Publishers.