Evaluating Software Project Portfolio Risks

Hélio R. Costa¹,²*, Marcio de O. Barros²,³, Guilherme H. Travassos²


Abstract

Like any other business, software development organizations try to maximize their profits and minimize their risks. Such risks represent the uncertain events and conditions that may prevent enterprises from reaching their goals. This possibility has turned 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 to evaluate a software project risk level by means of analogies with economical concepts. Such technique allows a manager to estimate the probability distribution of earnings and losses incurred by an organization according to its software project portfolio. This approach has been calibrated by data collected in an empirical study planned and executed to provide information about the relative importance of risk factors in software projects. A usage example of such 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 as many other investment activities: there must be a justification for using the available resources in the project and such justification usually depends on its expected returns. In software projects, resources are represented by the development team, 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 [1] present a roadmap for researchers emphasizing the need for a strategic investment approach to Software Engineering. Favaro [2] presents a real option-based approach addressing the value of software reuse. Boehm [3] states the concept of Software Economics as the field that seeks improvements in Software Design and Engineering through economic reasoning about 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 evaluation technique for software projects based on an economical view of the elements that constitute risk factors for software projects. In order to support this technique, an empirical study was conducted to quantify the relative importance of different risk factors for software development projects. As an application of the obtained results, we present an approach to calculate the probability distribution of losses and earnings that can be attained by a software development organization due to its software project portfolio. This methodology is inspired on credit risk theory for loan operations, which is extensively used by financial institutions.

We organize this paper in five sections and two appendixes. Section 1 comprises this introduction. In section 2, a technique to evaluate a project risk level is presented. In section 3, we describe an empirical study conducted during this research, aiming to raise 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, as well as an example, are presented. Finally, in section 5 we draw some conclusions and future perspectives for this research. Appendix I describes the questions used to support the risk level evaluation. Appendix II details the results of the t-test application in our empirical study.
2. Calculating a project risk level

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 the project risk level as the probability of a project failure in achieving its proposed goals. The risk level summarizes, in a single number, how risky is a project. For instance, if a project has a risk level of 30% it has 70% of chance to be successful. The risk level helps managers to compare two or more projects based on their risk-and-return ratios. Also, by quantifying the risks and highlighting which project aspects may be more risk-prone, managers can better identify where they can 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 economical analyses classify the risks faced by organizations into two categories: systemic and specific. Systemic risks influence a large number of organizations within an economic sector, a segment, or a 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 economical environment [4]. Examples of systemic risks are governmental policies upon markets, international economical scenario, and wars. Specific risks are such as local market conditions, organizational policies/practices, business objectives, and so on. While systemic risks are useful to analyze how external issues affect an organization, specific risks look for internal factors that can affect its performance.

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, among others). 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 during 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 to implement a given algorithm or the lack of appropriate hardware to conduct tests may not be relevant issues to every software project of 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 risk level. The technique is 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 regarding it. The questions help managers to evaluate the risk factors, relating them to practical elements that can be observed through the project characteristics. This structure allows a manager to address the project specific risks by answering the questions. Afterwards, the manager can aggregate project exposure to major factors that represent the systemic risks.

To develop this questionnaire, we started from the 194 questions proposed by Carr et al [5], which are classified according to 13 factors. Furthermore, we added questions to address issues highlighted by other approaches presented in the literature [6; 7; 8; 9; 10; 11] but not present in the original questionnaire. Finally, we reduced the number of factors (by redistributing questions and merging factors) to facilitate the paired comparison that was used in our empirical study (see Section 3). The number of questions currently associated to each factor is presented on Table 1.

The resulting questionnaire is shown in Appendix I. A brief description about each risk factor is presented in the following, 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, development process definition, adaptation and usage;
- **Control**: project execution, approval of intermediate artifacts, conflict resolution, team

---

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 the benefits from the project.
A project manager shall answer the questionnaire by assigning a number from 0 to 5 for each question. This coding operation is based on Likert’s research [12], where numbers are used to express values of an ordinal scale. The higher the manager’s perception about the impact of the issue described by the question in a given project, the higher shall be the answer. For instance, the question “Are the requirements stable?” might be answered with zero if requirements are not changing 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 at an industrial context, the questionnaire is distributed as a sequential electronic form (within the RISICARE tool – described in subsection 4.2) that allows the manager to select one of the above answers for each question.

After filling in 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 are not taken into account while calculating the median.

Next, the manager 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 exposure to risks related to a given risk factor, that is, the project specific risks for the factor.

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

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

In the last step, the project 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. For the sake of the example, only two factors are taken into account. The first factor has 3 questions and a 70% weight. The second factor comprises only 2 questions and a 30% weight.

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 concerning their risk exposure. In Section 4, we describe an application of the present technique to support the selection of a limited set of projects from a greater sample of available ones. In such application, the project risk level is not required to be an absolute measure of project failure probability, but a relative measure allowing a manager to compare the chances of two or more projects to fail.
Regarding factor weights, it could be argued that they may vary from project to project. Currently, we cannot refute this argument, but we assume that within a system category this difference can be attributed to specific risks. According to this logic, a high weighted factor that is not relevant to a project will have its weighted normalized median reduced if its questions are answered with low values.

3. An empirical study for evaluating factor weights

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

With the purpose of reducing this research’s scope and increase its precision, the first execution of the study was limited to evaluate the factor weights for Information Systems projects. In this section, we present the study summary and suggest its application in other domains and system categories. The study was planned according to the methodology proposed by Wohlin et al. [13].

Objective: To determine the weights that should be assigned to each risk factor to represent its contribution to the project risk level measurement (presented in Section 2). The weights reflect 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 to the failure of a project.

Subjects: The study was performed with the assistance of subjects from the industry and academia, represented by professionals, professors and graduate students, all of them with experience in software project development in the industry. The methods adopted to choose the subjects were Quota Sampling and Convenience Sampling. Thus, subjects were selected from distinct groups of the target population (software developers), but not randomly.

The subjects have agreed to participate and signed in 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>
<thead>
<tr>
<th>Academic Formation</th>
<th>Ph.D. 07</th>
<th>M.Sc. 15</th>
<th>MBA 19</th>
<th>Graduate 09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Managers</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysts</td>
<td></td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Years working with Software Development (Average)</td>
<td>12</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Projects Developed (Average)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current Occupation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Academic Researcher / Professor</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Industry Professional</td>
<td></td>
<td>39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Organizations Visited (Rio de Janeiro – Brazil)</td>
<td></td>
<td>26</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Project Size: While analyzing the factors, subjects were asked to keep in mind a specific project size, choosing the project size that closely represents 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 in a subject characterization form. The information provided by the subjects was used to group the projects by size (small, medium or large).

To obtain the project groups, the values of the project sizes (man-months) given by the subjects have been plotted in bar charts, so that could be observed a starting cutting point (visual observation) in order to create the three initial groups. Then, the standard deviation for each group was calculated. Finally, we interactively divided the groups around the starting cutting points, including or removing projects closer to these points in each group, so that we could minimize the standard deviation of each group. Doing so, we have defined small projects as those with 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 distinct factor weights could be identified for different project sizes.
Our goal was to observe whether there were any differences between the weights for the factors in distinct project sizes. For instance, subjects could say that the weight for the risk factor Testing in a small project could be different from a large project, and thus, there should exist different values to be applied in the risk quantification technique, depending on its size.

**Grouping:** It was expected that subjects’ experience and project size could influence the results of this research. Thus, we decided to block subjects and projects before data analysis. Subjects were classified according to their experience in three groups: low, medium, and high experience. The grouping process was based on a characterization form that was filled by each participant before evaluating the risk factors. The form captured academic and industrial experience information about each participant, such as the number of years working with software development, the number of developed projects, academic formation, and degree of experience in risk management. These information allowed us to summarize the participant’s expertise in a single number and group them using the approach proposed by Farias et al. [14] adapted for our purposes. The formula used to summarize the experience of a participant was:

\[
W(i) = \frac{Y(i)}{\text{Median}Y} + \frac{P(i)}{\text{Median}P} + A(i) + R(i);
\]

where \( W \) means the weight of the participant, \( Y \) is the number of years working with software development, and \( P \) is the number of projects developed. The values used to describe the Academic Formation (\( A \)) in Computer Science (CS) for the subjects were:

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

The values used to describe the experience in risk management (\( R \)) for the subjects were:

- \( R(i) = 0 \), for no experience;
- \( R(i) = 1 \), for low experience;
- \( R(i) = 2 \), for medium experience;
- \( R(i) = 3 \), for high experience, and;
- \( R(i) = 4 \), for expert.

After calculating the participant weights, they were grouped according to their experience. The grouping procedure allowed us to have a common weight for the opinions given by the subjects of the same group. To determine the more suitable cutting points for classifying participant experience as low, medium or high, the same procedure applied to group projects by size was applied to the participant experience data. It was calculated the average weight of the three groups and assigned a weight of 1.0 for the low experienced subjects. The weights of the following groups were determined by dividing their average weight value by the original average weight value calculated for the low experience subjects group. Table 4 presents the number of subjects, the average and the final weights calculated for each group. For instance, opinions given by individuals with medium experience have weight 1.61 times greater than opinions from individuals pertaining to the low experienced group.

<table>
<thead>
<tr>
<th># Participants</th>
<th>Low Exp</th>
<th>Medium Exp</th>
<th>High Exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Weight</td>
<td>4.09</td>
<td>6.62</td>
<td>10.78</td>
</tr>
<tr>
<td>Weights</td>
<td>1.0</td>
<td>1.61</td>
<td>2.63</td>
</tr>
</tbody>
</table>

**Instrumentation:** A subjective evaluation method was adopted to capture the experience of each participant. Among the available methods, the one chosen for this study was the 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 analyses each pair of objects and determines which one shall receive the preference. In our study, the subjects were asked to subjectively determine which risk factor was more important than the other one, i.e., which one had more contribution to project failure. After all comparisons made by the subjects, it was possible to determine which factor had more relevance for him or her and successively until the least relevant factor.

The main advantage of comparing the objects in pairs is to reduce the evaluation complexity and increase the precision of the results. Another point to be highlighted is that the human mind can more easily establish differences than estimate absolute values. Finally, through the comparison of an object with the others, the participant is forced to make a decision about the relation between two entities [15]. The disadvantage of this method is the great number of comparisons required in case of many objects: this value is in the \( n^2 \) order, being \( n \) the number of objects.

The instrument designed to support the paired comparisons was based on the MACBETH tool in its
demonstration version\textsuperscript{2} [16]. This tool allowed the subjects to formalize their preferences in a semantic way. It mapped the votes given by them into an interval scale representing the relative importance among risk factors. According to this mapping, each risk factor was given a percentile weight, so that the sum of all factor weights would be 100%. This mapping complies with the two specified properties for factor weights (P1 and P2) that were stated in Section 2. The MACBETH tool also checks for inconsistencies in the votes and helps in conflict resolution among comparisons, enforcing a coherent judgment without influencing or restricting the freedom of the subjects in expressing their opinion.

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

### Table 5

<table>
<thead>
<tr>
<th>Risk Factor</th>
<th>#</th>
<th>Risk Factor</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clients</td>
<td>14.36</td>
<td>Testing</td>
<td>10.81</td>
</tr>
<tr>
<td>Control</td>
<td>13.86</td>
<td>Policies/Structure</td>
<td>9.01</td>
</tr>
<tr>
<td>Planning</td>
<td>13.51</td>
<td>Design</td>
<td>8.40</td>
</tr>
<tr>
<td>Analysis</td>
<td>12.61</td>
<td>Coding</td>
<td>6.00</td>
</tr>
<tr>
<td>Team</td>
<td>11.11</td>
<td>Contracts</td>
<td>0.33</td>
</tr>
</tbody>
</table>

**Outlier Elimination:** After collecting the judgment of all subjects, we have 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 10 risk factors), only 16 were rejected, according to Table 6.

### Table 6

<table>
<thead>
<tr>
<th>Risk Factor</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>04</td>
</tr>
<tr>
<td>Design</td>
<td>02</td>
</tr>
<tr>
<td>Coding</td>
<td>01</td>
</tr>
<tr>
<td>Testing</td>
<td>00</td>
</tr>
<tr>
<td>Planning</td>
<td>03</td>
</tr>
</tbody>
</table>

**Data Analysis:** We have submitted the factor weights 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 factor weights were independent of project size, that is, if the same factor weights could be used to determine a project risk level, regardless its size. Therefore, the t-test was applied for the results of 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 wise comparison, the t-test is a special case of ANOVA.

It was not observed any significant difference for the Analysis, Design, Coding, Testing, Planning, and Contracts factors for different project sizes. However, the remaining factors – Team, Control, Policies & Structure, and Clients – have presented a small, though 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.

Therefore, we assumed that the project risk level measure presented in Section 2 should be calculated with distinct factor 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 about the t-test results can be found in Appendix II.

Besides the contribution to 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 in the software process to deal or reduce such issues depending on the project size. They can also support the decisions regarding risks mitigation for the project.

### Table 7

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

\textsuperscript{2} Even being a demonstration version, the tool has shown enough reliability and facilities to support the study.
others, may also constrain the generalization of the identified weights. Such detailed segmentation was out of the scope in 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 for specific contexts.

**Internal Validity:** Subjects were selected by Quota Sampling and Convenience Sampling, based on their experience in software projects development. They agreed to take part in the study. The number of subjects that voted for a certain project size was random and no dropout was observed, due to the nature of the study. The usage of MACBETH tool supporting the trial to produce the results was considered positive according to a follow up (qualitative) questionnaire filled by the subjects after using the tool in the study.

**External Validity:** Due to subjects’ characteristics and their selection process, it can be said they are representative of the developers’ population. The quantification of the risk factors was based uniquely in the experience of the subjects that had the opportunity to operate the tool in the time and the place they judged adequate. The interviewer helped subjects to use the tool, but was not allowed to influence their evaluation. The weights for risk factors can be considered valid as a starting point to evaluate the risks related to Information System projects and for the project sizes defined in this study. The number of subjects in the present study restricts our ability to generalize the observed results, but their reliability is not diminished. It could be interesting to replicate the study considering a different and large population and aggregate results to observe how they can evolve.

**Construction Validity:** As much as the factors as the questions used in the study were based on instruments presented in the technical literature. We relied on these information sources to consider them valid for the study. A pilot experiment was conducted prior to the execution of the main trial study to test the plan, the usefulness of the MACBETH tool, and to improve the study. The subjects were explained about the difference of the risk factors and characteristic of the questions pertaining to each factor. The possibility of guessing the result was eliminated by the use of paired comparison and the MACBETH tool, where the votes were checked for consistency. The percentile values for the risk factor weights were computed automatically, without the interference of the participant, neither the researchers. The software supported conflict resolution, by pointing inconsistent votes to the participants, requiring their correction and thus leading to a more reliable result.

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

4. An application of the proposed risk level measure

Managers can use the risk level measure approach presented in Section 2 (calibrated by their specific software project categories\(^1\)) 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 these projects may fail due to problems derived from risks incurred during the development process.

To address such a problem, we take advantage of a financial related 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 portfolio of loans. Such information is used to limit an organization’s losses due to default events. A default event occurs 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 the specification of the expected return over investment for each loan, since profits from borrowers that pay back their loans may compensate the losses generated by unpaid debts.

Software projects, as well as 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 its causes. Software projects also have an amount of money involved in their development and negotiation process.

\(^1\) Section 3 described the results regarding Information Systems risk factor weights
To some extent, money will be lost if a software project fails and profits will be attained if a project reaches its goals. Therefore, software development organizations may analyze their project portfolio in order to determine the probability of losses and earnings and maintain a worthy business even after a (limited) number of software project failures. As well as the loans domain, each project is expected to generate profit for the development organization. Such profit may compensate, to some extent, the losses caused by failing projects.

Based on this analogy, in the next section we present an application of loan transaction related credit risk concepts to software project evaluation context.

### 4.1 - Credit Risk Modeling

Credit risk can be defined as the probability that a borrower organization or counterpart will fail to meet its obligations in accordance with agreed terms in a given deal. Its measurement allows an enterprise to maintain its risk exposure to each business partner at an acceptable level. It also allows estimating the capital that an organization has to hold in order to sustain losses derived from unsuccessful dealings.

Main approaches for credit risk analysis presented in financial literature are the KMV, Credit Metrics, and CreditRisk+ models [17]. 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 not suitable.

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 [17]. Again, since there is no tradable asset to protect software projects from failure due to development related aspects (for instance, contract liabilities), the Credit Metrics approach is not directly applicable too.

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 one built upon the preceding and, thus, adding details and complexity to its calculation process. The first CreditRisk+ model is based on an analytical formula that calculates the loan portfolio losses and earnings probability distribution, each loan described by an amount of money lent to the borrower, the expected profits to be attained if the borrower pays the loan, and the inherent risks of that negotiation. The model assumes that all money given in a loan transaction will be lost in case of a default event. Borrowers are usually grouped in accordance to their ability to accomplish the payment and each group is assigned with a default probability.

The classification in groups is possible due to the great amount of borrowers and historical data available on the financial market. The second and third CreditRisk+ models also take into account the variability of the 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 [18] to describe the number of default events that may happen in loans given the borrowers of each group. According to the average number of defaults for each group, the expected profits and the amount of money invested in each loan, the model estimates the expected return for the whole portfolio, its variability, and sensibility to concentration of borrowers from each group.

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 concurrent projects, inhibiting the creation of projects groups. Therefore, instead of determining a default probability for each project group (to be used in the Poisson Distribution), we calculate a default probability for each project using a Bernoulli Distribution [19] 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, one or zero, being the first as much frequent as stated by \( p \).

By changing the probability distribution function, the CreditRisk+ analytical formula (based on the

---

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.
Poisson Distribution) does not hold anymore. Therefore, we propose the use of the Monte Carlo Simulation to estimate the losses and earnings probability distribution for a software project portfolio based on the approach proposed by [19].

In the CreditRisk+ model, some assumptions are made and they must be considered to determine the limits of our approach. For instance, the model presumes that all costs are spent when the loan is given and 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) occurring throughout the project life-cycle (intermediary cost along project execution, partial payments, and so on). Nevertheless, we assumed the premises proposed by the model, in order to apply it in the software development context. Table 8 presents a comparison between the financial and the software project assumptions.

Table 8
Financial and software assumptions

<table>
<thead>
<tr>
<th>Financial</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each loan has its own risks.</td>
<td>Each software project has its own risks.</td>
</tr>
<tr>
<td>States of the debtor: defaulter or not.</td>
<td>States of the project: canceled or not.</td>
</tr>
<tr>
<td>The amount of money involved in the loan is agreed before the operation is dealt.</td>
<td>The projects’ costs are estimated previously to its beginning.</td>
</tr>
<tr>
<td>The payment of the loan is made at the end of the term.</td>
<td>The payment of the project is made after its conclusion.</td>
</tr>
<tr>
<td>In case of a default event, the loss has a fixed value (the money agreed for the loan) not considering the recovering rate.</td>
<td>In case of project failure, the loss has a fixed value (project cost) regardless the expected return.</td>
</tr>
<tr>
<td>There is no correlation between default events, except for external factors that can affect the debtors in a similar manner.</td>
<td>There is no correlation between the failures of the projects, except for external risks that can affect them in a similar manner.</td>
</tr>
</tbody>
</table>

Assuming these premises and by means of simulations, it is possible to create the losses and earnings probability distribution for any number of projects of an organization’s portfolio, based only on their probability of fail (risk level), their costs, and the returns planned for the projects. The calculus of the probability distribution is performed as follows:

The simulator generates a Bernoulli random value for each project comprising the portfolio, which is compared with the project risk level. Such number indicates whether the project succeeds or fails for the current 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 value received from the client;
- At the end of each simulation, the resulting values (cost or profit) for all projects are summed. This represents the amount of money lost or earned by the organization from the development of such projects;
- The simulation process is repeated several times to provide a better estimation of the expected return of the projects portfolio, along with its variance. It is suggested at least 10,000 cycles;5
- After all simulations, the losses or earnings are grouped and mapped to the frequency domain. The grouping process depends on a predefined criteria, which is usually as simple as dividing the space between the better earning and the worst loss into a fixed number of bars. The frequency mapping is performed by counting the number of losses and earnings generated by the simulation process which fits in each group;
- Finally, the percentage of losses and earnings in each group is calculated from the total number of simulation cycles and such values are plotted in a chart, usually in a cumulative fashion.

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

Table 9
Project with risk levels and costs

<table>
<thead>
<tr>
<th>Project</th>
<th>Risk Level</th>
<th>Cost</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>10%</td>
<td>$10 K</td>
<td>$25 K</td>
</tr>
<tr>
<td>#2</td>
<td>20%</td>
<td>$20 K</td>
<td>$35 K</td>
</tr>
<tr>
<td>#3</td>
<td>15%</td>
<td>$30 K</td>
<td>$45 K</td>
</tr>
<tr>
<td>#4</td>
<td>25%</td>
<td>$40 K</td>
<td>$50 K</td>
</tr>
<tr>
<td>#5</td>
<td>20%</td>
<td>$50 K</td>
<td>$75 K</td>
</tr>
</tbody>
</table>

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 Figure 1.

5 In general, our simulations were stable after running about ten thousand cycles.
The results presented in Figure 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 observation. As it can be observed, 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%, that is the cumulative probability of obtaining values lowers than 80K). If such scenario is not affordable for the organization, some possibilities can be considered:

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

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

Table 10
<table>
<thead>
<tr>
<th>Project</th>
<th>Risk Level</th>
<th>Cost</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>10%</td>
<td>$10K</td>
<td>$30K</td>
</tr>
<tr>
<td>#2</td>
<td>18%</td>
<td>$22K</td>
<td>$35K</td>
</tr>
<tr>
<td>#3</td>
<td>15%</td>
<td>$25K</td>
<td>$45K</td>
</tr>
<tr>
<td>#4</td>
<td>20%</td>
<td>$45K</td>
<td>$60K</td>
</tr>
<tr>
<td>#5</td>
<td>15%</td>
<td>$55K</td>
<td>$80K</td>
</tr>
</tbody>
</table>

Performing the simulation again, we could obtain a distribution like the one presented in Figure 2.

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

Many distinct strategies can be tested by managers, in order to observe what is the best composition for the project portfolio and better application for enterprise’s resources. It must be noted that the approach should be used, preferably, by enterprises with many projects in its portfolio, because it may permit a wide variation of the parameters used in the simulation and a better analysis of the scenario.

It can also be said that, if an enterprise has in its portfolio a considerable large amount of projects in such a way that is possible to create groups with enough number of 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.

4.2. The RISICARE Tool

Due to the extension of the risk level evaluation questionnaire and the different weights that can be attributed to each project (according to its 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. Therefore, the RISICARE Tool was conceived and implemented to support these activities. The tool 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 fulfill the risk level evaluation 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.

Figure 3 and 4 respectively show the questionnaire and the simulation screens of the RISICARE tool, which is available for public download at the URL http://www.uniriotec.br/~marcio.barros/risks.

5. Conclusions and Future perspectives

In this paper, we have presented a technique to evaluate a software project risk level based on its systemic and specific risks. By means of analogies with the economical 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 through an empirical study performed to better understand the relative importance of risk factors regarding Information System projects. Besides, a CASE tool that supports the application of the proposed techniques has been developed and made available to the software engineering community. Therefore, we believe that a first cycle to develop such technology has been accomplished, what includes a proof of concept discussed in Section 4.

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

However, based on the results we have 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 claim, the technique has been made available into the context of a Software Development Environment that is being used by several Brazilian industrial organizations [20]. Our intention is to evaluate its feasibility into this organizational context to improve software process maturity and management capabilities making the results available in a nearest future.

As future perspectives, 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 extend some of the CreditRisk+ concepts in order to capture the cash-flow structure of a software project.

Acknowledgements

The authors would like to thank all the subjects that took part at the empirical study. We acknowledge the support from the Brazilian Air Force and CAPES. Prof.
Travassos is granted by the CNPq – Brazilian Research Council.

References

APPENDIX I – RISK QUESTIONNAIRE

The following items represent the questions grouped by risk factors that compose the risk identification questionnaire used to calculate a project risk level.

I. 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?
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 Factors requirements? Do you see any difficulty in meeting them?
18. Is the software requirements specification adequate to design the system?
19. Are there any problems with performance (Real-time response, Response time, Database response, Contention, or Access)?
20. Are the external interfaces changing?
21. Are the external interfaces completely defined?
22. Is there a requirements traceability mechanism that tracks requirements from the source specification through the test cases?
23. Is 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?

II. 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?

III. 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. Is the design changing while coding 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. Are the hardware specifications changing while the source code is being written?
11. Are the source codes reusable?

IV. TESTING
1. Is there a formal testing plan?
2. Are the test specifications adequate to fully test the system?
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 units testing that you think should be done?
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. Do all contractors take 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?

V. 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 development process well defined?
29. Can you measure whether the development process is meeting your productivity and quality goals?
30. Are there 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?
VI. 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?

VII. TEAM

1. Do people get trained in skills required for this project?
2. Do people get assigned to the project who does 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 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 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?

VIII. 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. Are the external interfaces changing 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 amount of external vendors a problem for the project?

IX. 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. Do the goals of the project are well defined?

X. CLIENTS

1. Are clients and users involved with the project?
2. Are users avert to changes?
3. Is the amount of changes in users’ life caused by the system a concern?
4. Is the amount of users affected by the system a concern?
5. Does the system will change the organizational structure of the client?
As stated in section 3, we have compared the observed factor weights in a pair-wise fashion 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 risk factor weights 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 risk factor 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 reason 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 evaluation technique.

### APPENDIX II – T-TEST RESULTS

#### I. Analysis

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>-0.812</td>
<td>2.045</td>
</tr>
<tr>
<td>Small x large</td>
<td>-1.506</td>
<td>2.052</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-0.385</td>
<td>2.042</td>
</tr>
</tbody>
</table>

#### II. Design

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>-0.002</td>
<td>2.040</td>
</tr>
<tr>
<td>Small x large</td>
<td>-0.127</td>
<td>2.042</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-0.116</td>
<td>2.045</td>
</tr>
</tbody>
</table>

#### III. Coding

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>0.097</td>
<td>2.037</td>
</tr>
<tr>
<td>Small x large</td>
<td>-0.574</td>
<td>2.042</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-0.638</td>
<td>2.042</td>
</tr>
</tbody>
</table>

#### IV. Testing

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>0.211</td>
<td>2.035</td>
</tr>
<tr>
<td>Small x large</td>
<td>-0.487</td>
<td>2.042</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-0.649</td>
<td>2.040</td>
</tr>
</tbody>
</table>

#### V. Planning

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>0.174</td>
<td>2.040</td>
</tr>
<tr>
<td>Small x large</td>
<td>-1.561</td>
<td>2.052</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-1.693</td>
<td>2.042</td>
</tr>
</tbody>
</table>

#### VI. Control

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>-1.339</td>
<td>2.037</td>
</tr>
<tr>
<td>Small x large</td>
<td>-2.553</td>
<td>2.045</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-1.642</td>
<td>2.040</td>
</tr>
</tbody>
</table>

#### VII. Team

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>-0.902</td>
<td>2.040</td>
</tr>
<tr>
<td>Small x large</td>
<td>-2.317</td>
<td>2.048</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-1.761</td>
<td>2.040</td>
</tr>
</tbody>
</table>

#### VIII. Contracts

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>-1.146</td>
<td>2.037</td>
</tr>
<tr>
<td>Small x large</td>
<td>-1.724</td>
<td>2.042</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-0.130</td>
<td>2.042</td>
</tr>
</tbody>
</table>

#### IX. Policies and Structure

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>-0.981</td>
<td>2.035</td>
</tr>
<tr>
<td>Small x large</td>
<td>-2.513</td>
<td>2.042</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-1.780</td>
<td>2.040</td>
</tr>
</tbody>
</table>

#### X. Clients

<table>
<thead>
<tr>
<th>Project Size</th>
<th>T0</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small x medium</td>
<td>-1.101</td>
<td>2.040</td>
</tr>
<tr>
<td>Small x large</td>
<td>-2.459</td>
<td>2.045</td>
</tr>
<tr>
<td>Medium x large</td>
<td>-1.512</td>
<td>2.042</td>
</tr>
</tbody>
</table>