

A Risk Based Economical Approach for Evaluating Software Project Portfolios

Hélio R. Costa
CCA-RJ / COPPE-UFRJ

Ponta do Galeão s/nº
Ilha do Governador – RJ, Brazil
55-21 2468-9352

heliorcosta@infolink.com.br

Marcio de O. Barros
DIA-UNIRIO

Av. Pasteur 458, Urca
Rio de Janeiro, Brazil
55-21 2244-5613

marcio.barros@uniriotec.br

Guilherme H. Travassos
COPPE-UFRJ

Caixa Postal: 68511
Rio de Janeiro, Brazil
55-21 2562-8712

ght@cos.ufrj.br

ABSTRACT

Software engineers have been applying economical concepts to shed light upon the value-related aspects of software development processes. Based on credit risk analysis concepts, we present an approach to estimate the probability distribution of losses and earnings that can be incurred by a software development organization according to its software project portfolio. Such approach is built upon an analogy that compares software projects to unhedged loans issued to unreliable borrowers. As loans may not be paid back, software projects may fail, leading their development organizations to losses. By applying this approach, an organization may estimate the variability of its expected profits related to a set of software projects. Initial calibrating data were acquired by accomplishing an experimental study.

Keywords

Software Engineering Economics, Risk Management, Empirical Studies on Software Engineering.

1. INTRODUCTION

The development of software systems can be analyzed as many other investment activities: since it consumes the organization's limited resources, there must be a reasonable justification for using the available resources in the project. Such justification usually depends on the expected returns over the required resources' costs and the risks involved in such business activity.

The notion that software development is a value-driven activity has been introduced by many authors [4] [8]. Boehm [2] defines Software Economics as the knowledge area that seeks improvements in software design and engineering through economic reasoning on product, process, program, portfolio, and policy issues.

According to [7], US\$ 55 million were spent in failing software projects in the United States just in 2004. When these software systems are developed by small companies, the development organization may not survive a conjunction of project failures. When this issue is addressed, three questions arise: (i) how much money can an organization lose or earn due to the development of a set of software projects?; (ii) what is the variability (or risk) of such profits or losses?; and (iii) what are the driving forces of such variability? If these questions could be answered, the organization may be able to better control its software development project portfolio in order to balance the expected profits related to the incurred risks.

In this paper, 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. The proposed methodology is based on credit risk theory for loan operations, which is extensively used by financial institutions. Given some assumptions, the proposed approach allows an organization to measure how volatile (or risky) the expected profits provided by a given set of projects can be. Moreover, the organization may address whether the development of a new project increases or reduces the overall portfolio risk and its expected returns.

To support this approach, we suggest a technique that allows the evaluation of an overall measure of a software project risk. Such evaluation is based on an economical view of the elements that constitute risk factors for software projects. An empirical study was planned and conducted to provide information to calibrate the proposed technique for a specific category of software project. The obtained results are summarized in this paper.

We organize this paper in four sections. Section 1 comprises this introduction. In section 2, we present our approach to establish the losses and earnings probability distribution for a software project portfolio. In section 3, we present the technique that evaluates a project overall risk level. Finally, in section 4 we draw some limitations and future perspectives for this research.

2. CREDIT RISK ANALYSIS

To address the questions described in the previous section, we take advantage of a financial analogy: we analyze a set of software projects as a portfolio of long-term bullet¹ loans issued to unreliable borrowers. Based on this analogy, we apply credit risk evaluation models to address how profitable a software project portfolio might be. Credit risk can be defined as the probability that a borrower will fail to meet its obligations in accordance to agreed terms in a given deal. Its measurement allows an enterprise to maintain its risk exposure to each borrower at an acceptable level. The main approaches for credit risk analysis presented in financial literature are the KMV, Credit Metrics, and CreditRisk+ models [1]. As in every economical framework, these models are based on underlying assumptions.

The KMV approach depends on the existence of a liquid market for trading equities for the loan borrowers. However, there is no

¹ Bullet loans are contracts where the notional value and accrued interests are paid in the contract maturity date, that is, without interest payments between the issue date and the maturity.

organizational borrower whose equities can be traded in an open market in our software project analogy: project risk is due to uncertain aspects of the development process or product. Thus, the KMV method is not directly applicable. The Credit Metrics model requires the existence of a secondary market for trading loans and protection assets based on these loans, such as credit swaps and default options. Again, since there is no tradable asset to protect software projects from failure due to development related aspects, the Credit Metrics approach is not directly applicable too.

Thus, we have adopted the CreditRisk+ model in our approach, which is a suite of three distinct models. The first one is based on an analytical formula that calculates the loan portfolio losses and earnings probability distribution according to n loans, each one described by an amount of money lent to the borrower, the expected profits to be attained, and a *default* probability, that is, the chance that the borrower will not pay back the loan. The second model takes the variability of the *default* probabilities into account, while the third considers the distribution of borrowers among economic sectors, thus managing portfolio diversification.

Borrowers are usually grouped according to their financial health, which is used as a proxy to their ability to accomplish the payment. A *default* probability is assigned for each group. The classification into groups is possible due to the large amount of borrowers and historical data available on the financial market.

In order to establish the capital required to maintain a loan portfolio, the CreditRisk+ model uses a Poisson distribution [5] 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 its sensibility to concentration of borrowers from each group.

Our approach is based on the first CreditRisk+ model, which deals with fixed *default* probabilities. In this model, some assumptions are made and they must be considered to determine the limits of our approach. Table 1 presents a comparison between the financial and the software project assumptions. It is important to notice that we do not imply that such conditions are observed in every software project, but that our approach, in its current development stage, can be applied when such conditions hold.

Table 1. Financial and software assumptions

<i>Financial</i>	<i>Software</i>
Each loan has its own risks.	Each software project has its own risks.
States of the borrower: default or not.	States of the software project: canceled or successful.
The amount of money involved in the loan is agreed before the operation is dealt.	Software project's costs are defined in its early development stages.
The payment for the loan is made at the end of the term.	The payment for the software project is made after its conclusion.
In case of a default event, the loss has a fixed value (the money lent) if interest rates accounted for.	In case of software project failure, the loss has a fixed value (project cost) regardless the expected return.
There is no correlation between default events, except for external factors that can affect the borrowers in a similar manner.	There is no correlation between the failures of the software projects, except for external risks that can affect them in a similar manner.

Software projects, as well as loans, have uncertain probabilities of success, represented by their risk levels. A *default* event for a software project may be associated to its cancellation, regardless of what has caused that cancellation. Software projects also have an amount of money involved in their development. To some extent, such money will be lost if a software project is cancelled 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 thus maintain a worthy business even after a (limited) number of project failures.

Mapping credit related concepts to software project portfolios, we observe that is not usual for some organizations to maintain a large number of concurrent projects. This inhibits the creation of projects groups, as borrowers are grouped in the CreditRisk+ models. So, instead of determining *default* probabilities for project groups and estimating the number of defaulters in each group with Poisson distribution (as in the CreditRisk+ models), we calculate a *default* probability for each project using a Bernoulli distribution [5] to describe its final state (successful or canceled). 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 result as frequent as stated by p .

By changing the probability distribution function, the CreditRisk+ analytical formula does not hold for our analysis. Therefore, we propose the use of the Monte Carlo Simulation to estimate the losses and earnings probability distribution for a software project portfolio. By running thousands of simulations and summing the losses incurred by failing projects to the profits observed in successful ones, we calculate the probability distribution of the expected profits provided by the project portfolio.

Assuming the premises presented in Table 1 and by means of Monte Carlo simulations, it is possible to create the losses and earnings probability distribution for any number of projects composing an organization's portfolio, based on their probability of failure (risk level), their costs, and the expected profits to be achieved by executing these projects. The probability distribution estimation is performed as follows:

- The simulator generates a Bernoulli random value for each project comprising the portfolio. Such number indicates if 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 (profit or loss) for all projects are summed. This represents the amount of money lost or earned by the organization from its participation in 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;

- 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 that fit 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.

To illustrate our approach, let us consider the projects in Table 2. Each one has its own risk level (*default* probability), calculated by applying the technique proposed in Section 3, the cost planned for its development, and its expected return. All values are fictitious and presented for the sake of an example.

Table 2. Project with risk levels, costs and returns

Projects	Risk Level	Cost	Return
#1	10%	\$ 10 K	\$ 25 K
#2	20%	\$ 20 K	\$ 35 K
#3	15%	\$ 30 K	\$ 45 K
#4	25%	\$ 40 K	\$ 50 K
#5	20%	\$ 50 K	\$ 75 K

After running the Monte Carlo Simulation ten thousand times, the cumulative probability distribution of the losses and earnings that can be attained by the organization is represented in Figure 1.

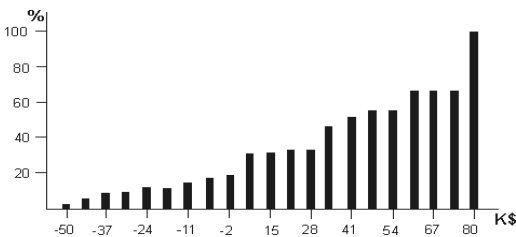


Figure 1. Losses and earnings cumulative probability distribution

From the results, we can observe that the probability of losing money in this portfolio is about 20% (represented by the cumulative probabilities of negative values). We can also observe that the likelihood of earning \$41K or more is about 50%, while the chance of attaining \$80K (maximum possible profit) is about 36%.

Many simulations can be performed by managers, changing the parameters used in the calculation, in order to observe what could be the best composition for the software project portfolio and better application of the organization's resources.

For instance, if the scenario presented above is not affordable for this organization some possibilities can be considered:

- Minimizing the cost of one or more projects;
- Improving the return of one or more projects;
- Canceling one or more projects;

- Performing a contention or contingency plan for the projects, to reduce their risk levels.

Let us suppose the enterprise has decided to perform a combination of such alternatives and has reduced the risk level as well as the cost of some projects. The proposed approach allows the manager to recalculate the losses and earnings probability distribution for this new scenario.

Figure 2 presents a scenario in which the costs of the projects 2, 3, and 4 were reduced to \$15K, \$25K, and \$35K respectively and the risk levels of projects 2 and 4 were reduced to 15% and 20% respectively.

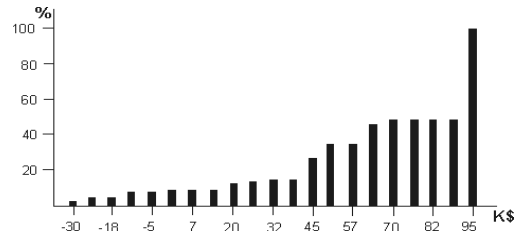


Figure 2. New losses and earnings cumulative probability distribution

In this situation, the likelihood of losing money in this portfolio is around 6%. The chance of earning around \$41K or more has improved to 80%, the maximum possible profit has increased to \$95K, and the chance to attain this value is about 54%.

Many combinations can be tried by managers to determine the best composition for the project portfolio and better application of the enterprise's resources.

3. RISK QUANTIFICATION

To apply the approach proposed in the Section 2, it is necessary to calculate the overall measure of a software project risk. The proposed risk measurement technique [3] classifies software risks as systemic or specific. Systemic risks are product, process, technical, and managerial issues that affect all software projects of a given category. Specific risks, on the other hand, are issues associated to the specific characteristics of a given project.

To identify software risks, we use a questionnaire built by the aggregation of several risk identification taxonomies presented in the literature. The questionnaire conveys 211 questions, which are classified in 10 groups named factors. Each question pertains to a single factor and addresses a particular characteristic of a software project (specific risk), aiming to closely describe the abstract concept represented by the factor (systemic risks) and relating it to more practical elements that can be evaluated by the manager from project's characteristics. Due to its extension, the questionnaire will not be covered in this paper. Further details about the questionnaire and the questions compiled can be found at <http://www.cos.ufrj.br/~heliior/riskquest.html>.

There are seven possible answers for each question, expressing the manager's opinion about its relevance in a project. The value 0 (zero) indicates that the issue covered by the question does not represent a risk for the project at hand (no risk). The other numbers (1 to 5) are coded values for a subjective rating scale, which vary from very low risk (1) to very high risk (5). This coding operation is based on Likert's research [6], where numbers are used to express values originally presented in an ordinal scale.

The manager has also an option to state that some questions are not relevant (NR) for the project.

After fulfilling the questionnaire, the manager calculates the median of the answers for each risk factor. Next, the manager normalizes the median value calculated for each factor, thus transforming it into a number in the [0, 1] interval. In the fourth step, the resulting number is multiplied by the risk factor's weight, which represents the degree of importance of a factor for the success or the failure of a software project. At this point, project specific risks are consolidated and weighted according to their systemic risks relative importance, thus resulting in a single number for each risk factor. In the last step, the manager sums the results for all factors to determine the overall project risk level.

The risk factors' weights are a delicate point in the technique. Due to the many different types of software projects that can be undertaken for an increasing number of domains, it is supposed that the risk factors' weights required by the technique presented 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 improve its precision, the execution of the study was limited to evaluating the risk factors' weights for Information Systems projects. The study (survey) was accomplished with 50 subjects (managers and senior analysts) within 27 different software development organizations in the city of Rio de Janeiro. The average software development experience of the participants was about 12 years, and the average number of projects developed by them was 14.

Subjects have given their opinion about the weights of the risk factors in relation to the selected system category and have agreed to participate and signed in a consent form regarding the study.

Aligned with the experience of the companies and professionals that took part in our study, we have chosen Information Systems as the field for study (that is, the selected system category).

The risk factors' weights achieved as results from the study are summarized in Table 3, according to different project sizes. We have observed that project size is a relevant measure during the study execution: it was noted that different weights were assigned to risks factors according to different project sizes. More details regarding the experimental study are presented elsewhere [3].

Table 3. Risk Factors' weights

Factor	Small (%)	Medium (%)	Large (%)
Analysis	12,36	12,57	10,78
Design	8,59	7,52	6,23
Coding	4,84	4,13	4,00
Tests	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

Risk factor's weights must attend to some properties, which we describe below:

- The sum of all risk factor weight values must be 1 (100%);

- The higher the relevance of a risk 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 each average answer can vary from zero to 5 and is divided, during the second step, by the maximum value that it can assume (5). The second property adjusts the specific risk evaluation (question answers) to the systemic risks presented by risk factors.

Table 4 presents an example of a project risk level calculation. For the sake of the example, only two risk factors are considered. The first factor has 3 questions and a 70% weight. The second factor conveys only 2 questions and a 30% weight.

Table 4. Calculating a project risk level

Answer the questionnaire	Factor 1			Factor 2	
	Q1	Q2	Q3	Q1	Q2
	2	4	3	2	4
Calculate the median of the answers	$\frac{(2 + 4 + 3)}{3} = 3$			$\frac{(4 + 2)}{2} = 3$	
Weight the normalized median	$\frac{3}{5} * 70\% = 42\%$			$\frac{3}{5} * 30\% = 18\%$	
Sum the adjusted average values	Risk Level = 60%				

4. THE RISICARE TOOL

Due to the great amount of questions that must be answered in the questionnaire, the operations needed to calculate a project risk level, and the simulations that generate the losses and earnings probability distribution, it is virtually infeasible for a project manager to achieve these tasks manually. Therefore, a tool was planned and implemented to support the approaches presented in this paper. The tool is divided in five different modules:

- Project characterization: used to provide project specific information such as project name, team size, project duration, cost and expected return;
- Questionnaire: allows the customization of each risk factor set of questions and provides means for the manager to answer them;
- Project Portfolio: allows the user to describe the set of projects that compose an organization's portfolio;
- Risk Level: calculates the risk level for each specific project composing an organization's portfolio;
- Simulation: perform simulations to determine the earnings and losses probability distribution.

Figure 3 shows RISICARE's simulation screen, presenting a set of projects and their losses and earnings probability distribution function.

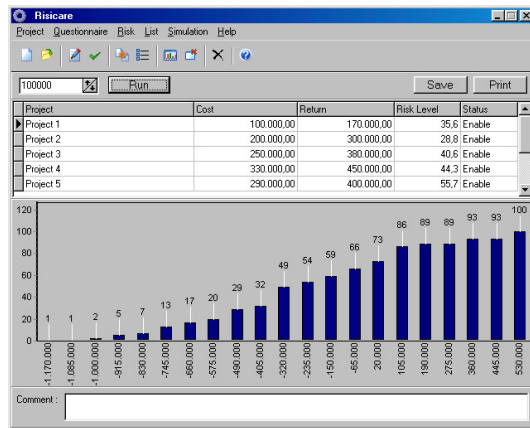


Figure 4. Simulation screen

5. LIMITATIONS AND PERSPECTIVES

We acknowledge some limitations in our approach. The first is that, to enhance its reliability, the empirical study was restricted to Information Systems. To generalize the application of the overall risk measurement, we intend, in a near future, to replicate the study for other system categories, such as components, military systems, and commercial packages.

The next issue regards the limits imposed by the CreditRisk+ model, where the dimension of time and the possibility to analyze the cash-flow along the project execution are not considered. We intend to expand our research toward a model that deals with more complex cash-flow structures. As possible evolutions for the *RISICARE* tool, it would be interesting to aggregate knowledge and intelligence to the tool, in order to help project managers to analyze simulation results, as well as allow them to save lessons learned during the process. It would also be interesting to allow the questionnaire answering activity to be performed by different people, collecting information and opinions from more sources to reduce the risk evaluation bias.

6. ACKNOWLEDGMENTS

The authors would like to thank all the subjects who took part in the empirical study, as well as the Brazilian Air Force, CAPES and CNPq for the financial support to this work.

7. REFERENCES

- [1] Barbanson R., An Introduction to Credit Risk with a Link to Insurance, *AFIR, Working Group*, 2004.
- [2] Boehm B.W., Sullivan K., Software Economics: A Roadmap, in *The Future of Software Engineering, 22nd International Conference on Software Engineering*, June, 2000.
- [3] Costa H.R., Barros M.O., Travassos G.H., Software Project Risk Evaluation Based on Specific and Systemic Risks, in: *Proceedings of the 16th International Conference of Software Engineering and Knowledge Engineering*, June, 2004.
- [4] Favaro J.M., Favaro K.R and Favaro P.F., Value Based Software Reuse Investment, *Annals of Software Engineering* 5, pp. 5 – 52, 1998.
- [5] Johnson R., Bhattacharyya G., *Statistical Concepts and Methods*, John Wiley & Sons. New York, 1977.

- [6] Likert R., *A technique for the Measurement of Attitudes*. Arquivos de Psicologia, 1932.
- [7] Standish Group, *Chaos Demographics – 2004 Third Quarter Research Report*, The Standish Group International Inc., 2004.
- [8] Sullivan K., Chalasani P., Jha S., Sazawal V., 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, 1999.