Scenario Oriented Project Management Knowledge Reuse within a Risk Analysis Process

Márcio de Oliveira Barros
Cláudia Maria Lima Werner
Guilherme Horta Travassos

COPPE / UFRJ – Computer Science Department
Caixa Postal: 68511 - CEP 21945-970 - Rio de Janeiro – RJ
Fax / Voice: 5521 590-2552
{marcio, werner, ght}@cos.ufrj.br

Abstract

Complex software development is a risky job. The number of unsuccessful projects surpasses the number of successful developments, particularly when large projects are analyzed. Many studies relate this situation to non-technical problems, especially to inadequate project management. Such studies point to communication problems, wrong allocation of team skills, insufficient training, and manager inability to predict and adjust project behavior. However, current project management techniques do not present many significant enhancements when compared to techniques used forty years ago.

This paper focuses the construction and reuse of project management scenario models. Such models hold the knowledge about management theories, actions, procedures, and strategies that can be applied or imposed upon a project. They compose a very important reusable knowledge base, which is mainly useful for project managers. We present an application of such approach within the context of software risk management, highlighting the processes used to coordinate the identification, documentation, and scenario impact evaluation activities.

Keywords: Knowledge reuse, project management, knowledge management.

1 Motivation

The state of practice and contemporary literature report the persistence of a recurrent problem in software development: most projects use more resources than planned, take more time to be concluded, provide less functionality and less quality than expected. Software unsuccessful stories can be found in several documented case studies and experiments over the last years [8].

Two different philosophical paradigms list responsible factors for the continuing software faults in industry. The first one relates software development problems to technological limitations, augmented by the increasing current software projects complexity [3]. The second one transfers the responsibility to management problems, bad communication, and difficulties in handling the uncertainties that are present in innovative, complex or large software projects [16].

Most of the techniques currently applied to software project management are dated from the 1950’s and 1960’s. These techniques include work breakdown structures, PERT/CPM charts, and Gantt diagrams. They are based on a set of fundamental assumptions, necessary for their accuracy [8]. Some of these techniques require that projects have clear and delimited objectives, there exists at least one solution to the problem at hand, development time and resources can be precisely stated before the project starts, the operational environment is well known and defined, and that quality metrics can be quantified for the project. Due to many successes resulted from using these techniques in large projects, project managers tend to passively take their underpinning assumptions for granted in every project [8]. This misconception is very common and dangerous in software project management.

Baseline assumptions from current project management techniques require a project behavior to be known from the beginning. The development of projects for innovative application domains with high requirement volatility, the need for domain integration to achieve several objectives, ambiguity, complexity, discontinuity, diseconomy of scale, nonlinearities, and complex feedback loops are characteristics of large projects. These characteristics undermine the assumptions of traditional techniques. Invalidation of these assumptions, and consequently of these techniques, creates demand for new management paradigms and technologies.

We present Scenario Based Project Management, an innovative paradigm for project management that assumes
that a project manager should define and document an expected behavior for a project process. Since this behavior can be affected by unexpected events during project development, the manager should test its sensibility to several combinations of such events, getting feedback about possible risks that can challenge the project success. Project management scenarios represent such events, conveying knowledge that can be reused in several projects. Scenario Based Project Management is strongly supported by risk management, formal process modeling, and simulation techniques.

This paper is organized in six sections. The first section comprises this motivation. Section 2 presents the fundamentals of the Scenario Based Project Management paradigm. Section 3 presents a risk management process based on documenting risks as scenarios and using their simulation capabilities for risk evaluation. Section 4 presents an application study of the proposed techniques. Section 5 compares the proposed approach to related works. Finally, section 6 presents final considerations and future perspectives of this work.

2 Scenario Based Project Management Paradigm

The following statement briefly describes the traditional view of software project management: plan the flight and fly the plan. This guideline is deeply rooted in the assumptions taken by the currently applied software management techniques. It presumes that the project manager can provide a detailed strategy to guide the development team from early product specifications to the final software system, within predefined schedule, quality, and budget constraints. The accomplishment of such task requires a complete understanding of the development effort, which may be unavailable in innovative projects.

Traditional planning considers only one route from early system requirements descriptions to the final deployable software product. We advocate that uncertainties presented along the development process, such as little user commitment to the project or errors crippling for “finished” products, must be considered in the project plan as alternative routes to project behavior over time. The plan is therefore a dynamic artifact and is susceptible to change along the development process, while project uncertainties flourish to problems or are contained.

The Scenario Based Project Management paradigm comprises a set of techniques that allow a software project manager to define a project expected behavior and several different scenarios that might occur along its development. Combinations of such scenarios are applied upon the project and used to evaluate their impact upon its behavior.

The architecture of the proposed paradigm is presented in Figure 1. It is centered on two artifacts: the project model (represented by the “PM” blocks in Figure 1) and scenario models. The project model defines the project expected behavior, while scenario models describe alternative routes that the project may follow, due to the occurrence of unexpected events. Both artifacts are formal software project models. A formal model describes the relationships among real world elements through mathematical or logical postulations. The advantages and applications of formal project models are discussed elsewhere [4]. Scenario Based Project Management uses system dynamics software project models, which are described in terms of mathematical equations.

System dynamics is a modeling technique and language to describe complex systems, focusing on their structural aspects [9]. This technique identifies and models cause-effect relationships and feedback loops. These relationships are described by using flow diagrams composed by four basic elements: stocks, rates, flows and processes. Stocks represent elements that can be accumulated or consumed over time. A stock is described by its level, that is, the number of elements in the stock at a given moment. Rates describe stock variations, formulating their raises and depletions in an equation. Processes and flows complement complex rate equations and calculate intermediate model variables.

The project model is based on the system dynamics software project model created by Abdel-Hamid and Madnick [1], which formalizes the effects of management

![Figure 1 – Scenario based project management paradigm architecture](image-url)
Traditionally, system dynamics project models blend different combinations of uncertain events and theories. Project models allow a manager to test project sensitivity to models. Breaking this uniformity makes model developers able to represent the particular features of each software project among its developers, activities, resources, and artifacts. For instance, all experienced developers are supposed to have the same productivity, developers are assumed to be in the same exhaustion level, activities are supposed to take the same time to develop, and so on. We assume that this model simplification is due to system dynamics inherent incapability to describe element’s particular features. However, we believe that, if such a model is to be used for operational management, it must capture the differences among its developers, activities, resources, and artifacts. Breaking this uniformity makes model developers able to represent the particular features of each software project element, which allows the construction of richer project models.

Finally, separation of facts from assumptions within a project model allows a manager to test project sensitivity to different combinations of uncertain events and theories. Traditionally, system dynamics project models blend known facts about the real-world elements that compose a project with several assumptions upon their behavior and interaction. This framework allows a manager to analyze the implications of such assumptions upon project relevant variables, such as conclusion time and cost. However, it is generally difficult to test other assumptions than those provided with the model.

By separating facts from assumptions, Scenario Based Project Management supports the evaluation of several combinations of assumptions upon the provided facts. The project model represents known facts, while assumptions are separately modeled by scenarios.

Scenarios hold the representation of events, policies, management procedures, actions, and strategies that cannot be considered part of a project, but practices imposed or applied to the project and exceptional situations that may happen throughout its development. Scenarios can be used to modify a project model, which may impose project behavior deviation due to the events they represent. Since such assumptions are valid for several distinct projects, scenarios convey potentially reusable management knowledge.

Figure 2 presents the equations that compose a simple scenario. This scenario represents productivity losses due to communication overhead in a software development project where two or more developers work together in a single activity. Its equations and assumptions were adapted from [1]. The scenario uses an extended system dynamics syntax, which is extensively described in [5].

| TABLE COMM_OVH 0, 0.015, 0.06, 0.135, 0.24, 0.375, 0.54; |
| PROC PROD_LOSS LOOKUP (COMM_OVH, Count (Team), 0, 30); |
| AFFECT PRODUCTIVITY PRODUCTIVITY * (1 – PROD_LOSS); |

Figure 2 – Equations that compose a communication overhead scenario

Simulation is the technique that allows project model behavior evaluation. Since project models are described by mathematical formulations, their equations can be evaluated to reveal the underlying project behavior. The project model is called a concrete model, in the sense that it can be directly simulated, without any further integration to any scenario. The simulation of the isolated project model describes project expected behavior, without the influence of the events and theories selected by the project manager.

The integration of scenarios to a project model may reveal changes in the project behavior. Simulating a project model with a specific scenario demonstrates the potential impact of the scenario occurrence in the project behavior. This kind of simulation allows the manager to test project
sensitivity to the events, theories, and strategies described by the scenario model. Scenario models, which can only be simulated when integrated to a project model, are called abstract models.

Scenarios are developed to adjust the equations of rates and processes defined in the project model. In the scenario presented in Figure 2, the AFFECT clause indicates that the scenario modifies the PRODUCTIVITY equation defined by the project model. Since several scenarios can adjust the same equation, scenario integration ordering is relevant.

Scenarios can be organized in an organization-wide knowledge base to project managers. They allow formal documentation of assumptions and proven information about managing software projects. Senior project managers can create scenarios, based on their personal experience, allowing less experienced managers to share their knowledge by retrieving and reusing such scenario models. During application development, when a project manager determines the current application domain, technologies to be applied during the development, developer roles, artifacts to be built, and so on, scenarios associated to these elements help the manager to explore project uncertainties. The manager retrieves those scenarios that he believes can affect his project and evaluates their impact upon the project behavior. Scenarios can be analyzed in isolation or in groups, representing their combined impact upon the project behavior.

3 Scenario Based Risk Management

The techniques presented in the previous section describe how scenarios are formulated, integrated within a project model, and how project behavior is evaluated with scenario combinations to determine their impact upon it. By representing the risks faced by a project as scenarios, we use such techniques to define a risk management process.

Generically, risk is defined as the likelihood of danger or loss. Risk management techniques were emphasized in the software development context by the end of the 1980’s, by the spiral model [7]. In the last decade, the work on software risk management has followed several distinct directions: new qualitative risk management processes [14] [12], quantitative processes [11], knowledge based processes [15], and risk information databases [10].

In the context of the Scenario Based Project Management paradigm, risk management is based on documenting risks through scenarios, reusing these scenarios along several projects, and simulating scenario combinations for risk evaluation within a particular project. These activities imply two different concerns in the risk management process. First, risks are identified and documented using scenarios. Next, risks are reused along several projects, allowing the simulation of their impact upon the project behavior.

These two different concerns resemble a generic reuse process framework, as depicted in Figure 3. This framework is based on two sub-processes: one to develop reusable artifacts (i.e., development for reuse), identifying and documenting these software assets, and the other for application development based on reusing previously crafted artifacts (i.e., development with reuse), selecting and adapting them from a reusable information base. Applying the generic process framework to the Scenario Based Project Management paradigm, its risk management process is divided into two sub-processes: one for risk identification, and the other for risk reuse during application development.

The main objective of the risk management process for risk identification is to describe common risks associated to a specific project element, such as an application domain, technologies, developer roles, software artifacts, and resources. This risk management process organizes such risk information and allows its reuse in several application development processes. The first risk management process is executed when an organization explores a new project element that can impose risks to a development effort.

The main objective of the risk management process for application development is to identify and evaluate risks that can affect a project due to its specific project elements.

Figure 3 – Generic process framework for software reuse
This process reuses the risks associated to project elements, identified and documented by the former risk management process. It occurs in parallel with an application’s development life cycle, tracking the evolution, and resolving its risks.

The risk management processes enforce quantitative risk analysis, modeling risk impact and resolution strategies efficacy as scenario models. By forcing risk documentation through scenarios, we expect the risk identification team to precisely state their assumptions about the effects of these risks upon the project. Without formalizing the risk documentation, this reasoning may be highly subjective, eventually unusable to future projects.

The fundamental reusable artifact connecting the two risk management processes is the information describing a risk, represented as risk archetypes. A risk archetype is a description of a potential recurring problem, which can cause some kind of loss to a software product or development process. It includes a description of the context where the problem can occur and provides solution strategies that can be applied before and after the problem occurrence. Risk archetypes serve as an awareness mechanism to the project manager and as containers for scenarios that model risk impact, contention and contingency strategies.

Risk archetypes identify, document and evaluate potential events that can negatively affect software development projects. The experience of forecasting, identifying and resolving the risk is documented by the risk archetype, which can be effectively reused in future projects. Risk archetypes determine the information that must be stored with the potential problem description, presenting its characteristics in a structured and standard form. The use of a standard representation promotes risk communication, understanding, and a clear distinction among several risk events.

<table>
<thead>
<tr>
<th>Archetype Identification</th>
<th>Contention Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Name: Validation risk</td>
<td>1. Plan: Development of prototypes</td>
</tr>
<tr>
<td>2. Alias: Incorrect requirements</td>
<td>2. Condition: Project has low planned validation mechanisms</td>
</tr>
<tr>
<td>1. Potential problem: developing a system whose functionality is not accepted or do not satisfy the client.</td>
<td>3. Effects: Schedule overrun (time invested in prototyping)</td>
</tr>
<tr>
<td>2. Effects: quality loss of the software product</td>
<td>4. Impact: system dynamics scenario model</td>
</tr>
<tr>
<td>3. Impact: system dynamics scenario model</td>
<td>1. Plan: Application of JAD techniques</td>
</tr>
<tr>
<td>2. Condition: user involvement is low</td>
<td>3. Effects: Schedule overrun (increased communication)</td>
</tr>
<tr>
<td>4. Impact: system dynamics scenario model</td>
<td>1. Plan: Allocation of more time for requirement specification</td>
</tr>
<tr>
<td>2. Condition: many ambiguities and/or “TBD” requirements</td>
<td>3. Effects: Schedule overrun (dedication to requirements)</td>
</tr>
<tr>
<td>4. Impact: system dynamics scenario model</td>
<td>1. Plan: schedule extension</td>
</tr>
<tr>
<td>2. Effects: allow prototyping and requirement refinement;</td>
<td>3. Impact: system dynamics model</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identification Mechanisms</th>
<th>Contingency Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Context: projects where the development team has little interaction with the clients and final users, projects that don’t develop prototypes, projects with several distinct clients, innovative projects.</td>
<td>1. Induced Risks:</td>
</tr>
<tr>
<td>2. Checklist:</td>
<td>• Schedule overrun</td>
</tr>
<tr>
<td>• Evaluate the level of ambiguity of system requirements</td>
<td>• Cost overrun</td>
</tr>
<tr>
<td>• Evaluate the number of “to be defined” requirements</td>
<td>• Reputation</td>
</tr>
<tr>
<td>• Evaluate user involvement in the development project</td>
<td>2. Known cases:</td>
</tr>
<tr>
<td>• Is any validation technique, like operational simulation or prototyping, to be applied during the project?</td>
<td>• Project 105: polymer classification / analysis</td>
</tr>
<tr>
<td>3. Known cases:</td>
<td>• Project 112: e-supermarket</td>
</tr>
<tr>
<td>• Project 105: polymer classification / analysis</td>
<td>2. Similar Risks:</td>
</tr>
<tr>
<td>• Project 112: e-supermarket</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4 - Validation risk archetype
We expect that the structured form promotes information completeness, so that a risk archetype conveys all the information needed for future risk analysis. Information completeness is very important to reduce psychological barriers to risk analysis: risk is an abstract concept [14], and an incomplete description of the risks to which a project is subjected inhibits the motivation for risk management. Figure 4 presents an example of a risk archetype for the validation risk, that is, the risk of developing the wrong system.

4 An Application Example

This section presents an example of how the proposed risk management processes can be applied in real world projects. The results presented here derive from a study using both risk management processes in two projects developed within the same application domain. The selected domain was the financial market risk management. The selected projects convey the specialization of a risk management software system for the investment, debt, and operational portfolios of two large-size Brazilian enterprises.

Before the accomplishment of the first project, there was little information about the risks related to the development of a risk management software system for a Brazilian enterprise. The team selected for the project had previous experience in implementing a financial market risk management software system for an investment bank. However, the specialization of such software for an enterprise presented several differences:

- Risk horizons are very different: in general, investment banks are interested in short-term market fluctuations, such as the variations of exchange rates and equity prices in a single day. They want to know what is their potential loss due to market variations from today to tomorrow. On the other hand, enterprises have their medium and long-range debt portfolios in foreign countries. Such portfolios extend for multiple years and are dependent of several future exchange rates. Enterprise are more interested in what will be their profit variation due to market fluctuations along one fiscal year or semester;

- Commodities: by the time of the first project, the investment bank had little interest in commodities, such as coffee, gold, petroleum, or steel. However, these commodities have great importance for enterprises that sell such products in an open, global market. Since the market dictates commodities prices, the company profit or loss is subjected to market prices variations. Even if the company sells its products only in local markets, it may need to import incomes for its production lines. Future markets, such as London or Chicago Mercantile Exchange, can also regulate those incomes.

These are only two examples to show the different visions of risk management for an investment bank and an enterprise. The first project in an enterprise was susceptible to some risks related to these characteristics, along with some generic software development risks. The domain-dependent risks were documented as risk archetypes and associated to the financial market risk management domain. Although a complete description of their risk archetypes is beyond the scope of this paper, short examples of such risks are shown below:

- Commodities modeling: due to their saisonality and dependency on external factors, such as weather conditions, provider organizations, plagues, and diseases, the uncertainties associated with specific commodities are hard to model. Complex stochastic processes describe the behavior of some commodities, such as petroleum and gold. Measuring risks depending on such processes is also hard, specially in commodities derivatives;

- Monte-Carlo based risk evaluation: Monte-Carlo simulations are frequently used to measure risks when financial market instruments cannot be described by closed-form equations. While they represent a very flexible and reliable risk evaluation framework, Monte-Carlo simulations require a lot of computing resources, slowing down risk evaluation. The dependence on such simulations for large portfolios can reduce the risk management software quality, since the user might wait too long for every analysis he decides to perform;

- Portfolio composition volatility: denoted here by the frequent inclusion of new types of financial market instruments in an enterprise portfolio. Portfolio composition volatility is a domain-dependent facet of the requirement volatility risk. Every time a new instrument is inserted to the risk management software, its risk mapping must be defined (designed and coded), and maybe new risk factors and historical information needs to be added.

Even without a formal assessment of its risks, the first project was rated as a success, being concluded only one week behind its four-month schedule. The project was divided in four major phases: portfolio elicitation, statistical analysis, portfolio mapping to risk factors, and risk mapping evaluation. Risk archetypes documentation, accomplished by executing the risk management processes for risk identification, was performed in a post-mortem fashion, after the first project conclusion.
The second project executed the risk management process for application development, using the risk archetypes knowledge base created after the first project. Such risk archetypes were organized in a printed document, distributed to the project participants. Project development has shown the need for a better distribution media. The printed document was not always available when needed and navigation through risk archetypes required time-consuming references to the document index. To minimize such problems, we intend to develop tools to allow risk archetype documentation through HTML hyper documents containing navigation hyperlinks. Such documents are to be distributed within a software development environment or within the Internet to help the availability problem.

The second project was a larger one, since the company had several operational portfolios, distributed along its subsidiaries and differing on the products it sells, buys or just distributes. It was scheduled to take six months. Again, the project was concluded one week later, but we believe it would be much more behind schedule if it were not the risk archetypes and the experience gained from the first project.

The risk archetypes were used as risk warnings early in the second project development process. When the portfolio elicitation team (mainly composed by financial engineers) came to the risk mapping team (mainly composed by designers and programmers) with a list of market instruments, the risk archetypes helped the developers to pinpoint instruments that could represent risks to the project. The portfolio analysts then concentrated in the decomposition of such instruments, replicating their behavior from known market instruments where possible. Monte-Carlo simulations, which were needed for such complex instruments, were provided for fine-grained analysis as a separate tool, while good closed-form approximations were developed for day-to-day risk monitoring.

The second project also created new risk archetypes, which again were documented in a post-mortem fashion. The present study has shown the usefulness of risk archetypes to document project management experience and the need for tools to improve the proposed techniques applicability. Both the first and the second projects have not used project and scenario models to evaluate risk impact over project behavior. However, both projects were tracked and activity duration, error generation, and portfolio volatility information is available for scenario development. We intend to run another project in the same domain, this time using dynamic models.

5 Related Works

The proposed approach to risk management has several innovations when compared to previous software risk management processes. First, the proposed approach considers the relationships between risk management and particular project elements, such as application domains, technologies, software artifacts, resources and developer roles. These relationships are built upon risk archetypes, which are linked to project elements when created and retrieved when an application using such project elements is developed.

Second, the risk management processes clearly embed software reuse, following a generic reuse framework. The first process allows a project manager to document a particular risk or to analyze a specific project element to document these risks. The second process helps a project manager to identify the risks within an application, reusing the available knowledge from previously documented risks. Kontio [14] also proposes risk information reuse. Although they do not describe a risk management process, Althoff et al. [2] propose mechanisms to organize, retrieve, and reuse software development experience. Managers can use such experience to predict problems early in the software development life cycle.

Next, risk archetypes offer a structured way to document risk information. We expect that this structured documentation helps to break through psychological barriers to risk analysis, due to incomplete information about risks. Garvey [10] and Jones [13] present standard information structures to document risks, but their information structures do not allow risk impact simulation within a specific project. Hall [12] proposes several risk forms or databases to store static risk information. Again, this static information does not allow an integrated analysis of risk impact.

Finally, risk impact, contention and contingency plan effects can be aggregated to project models in a way that they can be evaluated. Such evaluation captures the possibly nonlinear risk and plan impact over the process. For instance, consider a risk whose outcome is to reduce a project team reduction effect can be directly evaluated from the project team by a predefined amount of developers. This team reduction effect can be directly evaluated from the project model, capturing all the relationships with project current status and other model variables. Kontio [14] quantifies the effects of risk scenarios using utility functions. However, these functions become very complicated when several process aspects are taken into account, due to nonlinear relationships among scenario elements and feedback loops inherent to the software development process. Risk impact evaluation through simulation offers a better way to understand the cascade of impacts imposed by the risk upon project behavior.

6 Final Considerations

This paper described an approach to develop, retrieve, and reuse project management knowledge and experience.
Such knowledge is represented in scenario models, which are formal models of management actions and theories. Scenarios are used to model risk impact and resolution strategies efficacy within risk archetypes. A risk archetype is an information structure that holds knowledge about software development risks.

The main contribution of this paper is showing how risk archetypes and scenario models can represent reusable project management knowledge. We also present a risk management process that uses risk archetypes and scenario models to identify and evaluate software development risks. The proposed process resembles a reuse process framework, where two sub-processes are respectively responsible for identifying and reusing risk information. An application of the proposed techniques and processes along two distinct projects is also presented.

Scenario Based Project Management is an ongoing work. We have made some preliminary studies in two similar projects to understand the relationship between risks and application domain. However, since these studies were limited to a small project and a single domain, they are not conclusive, but motivating to continue the work towards a more general evaluation.

Next steps are regarding the definition of more complex project models and applying them to operational project management situations. After understanding the process, we also intend to finish the implementation of tools supporting project model definition, scenario development, scenario integration, and a simulation environment.

Besides, some experiments have been planned, such as a single object, multi-test experiment [17] that will run in an academic environment. The experiment goal is to verify the feasibility of using such approach in real projects.

Acknowledgements

The authors would like to thank CNPq and CAPES, for their financial investment in this work.

References