Using Risk Analysis and Patterns to Tailor Software Processes

Júlio Hartmann, Lisandra M. Fontoura, Roberto T. Price

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

e-mail: {jhartmann,lisandra,tomprice}@inf.ufrgs.br

Abstract. *This paper discusses an approach to tailor software development processes and methodologies based on organizational patterns and risk criteria. The purpose of the approach is to adapt an organizational pattern language to the context of a given project. The most suitable organizational patterns to the requirements of the project should be chosen analyzing the risks and the criticality context of the project. Organizational patterns which describe preventive techniques for the identified risks can be selected by a systematic retrieval system. The process designer evaluates the selected patterns and takes the adaptation decisions. PMT-Tool, a tool that supports the approach, is briefly described.*

1. Introduction

There is a vast literature on software development methodologies. Each methodology was usually developed considering a given context, with its own goals, by people with different backgrounds and experiences, and frequently the basic assumptions are not clearly stated on the texts about them. It is very hard for a process designer to select, elaborate or combine an appropriate software development methodology for a given project. Research of systematic approaches for helping the process designer or the project manager on executing these tasks is required.

Despite all the effort on development methodologies and processes, many software projects still fail to meet its goals, finishing late, out of budget, or without a satisfied user. Some of them fail entirely or are canceled before ever delivering any results. One explanation for the high failure rate is that project managers are not acting to assess and manage the risks involved in these projects [21]. Risk management is a collection of methods aimed at minimizing or reducing the effects of project failure [1].

Risk management involves risk evaluation and risk control [5]. Risk evaluation can be subdivided in risk identification, risk analysis, and risk prioritization [4]. Risk identification results in a list of the possible risks of a project. The risk analysis is an estimation of the probability and the consequence of each of the identified risks [16]. In the risk prioritization, the identified risks are ordered by their importance [4]. Risk control is the process of elaborating and implementing risk resolution plans, monitoring the status of the risks, and developing and documenting strategies to deal with them [16]. Risk resolution is the elimination or minimization of a risk, by executing actions described on the risk management plan [4].

This work uses a risk-driven approach to aid the selection of appropriate methodological constructs for a given project in the form of an organizational pattern language. A pattern describes the essential part of a solution to a recurring problem in a

given context. Organizational and process patterns capture successful management practices of software development [14], and can be used to shape a new organization's development process, or to improve an existing one. A pattern language is a collection of patterns which build on each other to generate a system [10]. In this work, organizational patterns are used to describe resolution techniques to software risks.

There are several documented organizational and process patterns and pattern languages, such as [9], [11], [3], [13], [29] and [17]. This work sets means of structuring a pattern repository, by classification and association of patterns. Risk resolution rules are used to associate each pattern to the risks it is intended to tackle. Each rule is used in association with a given project context. The project criticality context is established in this work, in a similar approach to Cockburn [7] and Boehm [6], by analyzing three criteria: the defects criticality (the possibility of loss caused by a defect), the number of people affected, and the skill level of the development team. Other criteria could be defined, such as the priorities of the project, in terms of productivity and tolerance or of legal suitability [8], but the three criteria were selected in this work for being more meaningful for balancing between agile and plan-driven methodologies [6].

A tool (PMT-tool) for the selection of patterns by the process designer was developed, in which the project's own pattern language is created by selecting the most appropriate patterns from the repository, aided by a selection mechanism. The selection system uses the complete pattern repository and a prioritized risk list as its input. The risk list is obtained through risk identification, risk exposure analysis, and risk prioritization. The resulting selection is a suggested list of organizational patterns that may be applied as actions to prevent or minimize the identified risks. The process designer chooses some of the suggested patterns and adds them to its project pattern language. The project pattern language can be browsed by the project team members, who use it as a source of learning for resolving and preventing risks in the software project. Specific documents and workflows for the project processes may be manually produced from the resulting pattern language, but are outside the scope of this work.

Section 2 explores related work. In section 3 it is explained how to structure a knowledge repository about software development methodologies, with patterns, risks, and risk resolution rules. In section 4, the approach for performing a project risk analysis is explained. Section 5 explains the systematic selection mechanism of the PMT-Tool, while section 6 explores the tool in further detail through a case study. Section 7 concludes the paper and describes some future work.

2. Related work

James Coplien has made available in his web site [11] a large organizational pattern language. The pattern language navigational structure is presented in a graph form. Each node in the graph represents a pattern, while the arcs represent the linked dependencies between the patterns. By clicking on a node, the user navigates to the textual description of the pattern, which also includes hypertext links to the related patterns. The patterns include process descriptions such as *Early and Regular Delivery* and *Scenarios Define Problem*, as well as role descriptions such as *Architect Controls Product* and *Mercenary Analyst*. Team practices and values such as *Code Ownership*, *Compensate Success*, and *Public Character* are also included in the pattern language. Coplien's patterns capture recurring best practices applied to its organization environment. Some of his patterns can be reused to target project risks in a different software project, but because of the

pattern language extension it is a hard work to select the appropriate patterns, and the selection has to be done empirically by the project manager or by the process designer.

Vasconcelos and Werner [30] use process patterns to form a knowledge base about software processes. Their approach is supported by a tool called Memphis – a reuse-based software development environment - which helps to consider different alternatives when processes are composed. However, the tool does not help with the decisions the process designer has to make in order to choose between the process alternatives.

The Open (Object-oriented Process, Environment and Notation) [18] approach uses a process meta-model from which a project specific process is instantiated. It allows a high degree of flexibility for the process user, who has to make several decisions on the instantiation process.

RUP (Rational Unified Process) [26] is a commercial product, and can be considered a full instance of the Unified Software Development Process [20], including material available in HTML in a web site. It is a pre-tailored process, so it is possible to change portions of it, such as expand, modify or remove steps from specific activities, or even remove entire activities. But it should be considered a tailorable method rather than a tailorable methodological framework [18]. Both the Open and the RUP approaches do not help with the decisions the process engineer has to make on designing a specific software development process.

Alistair Cockburn [7] proposes a selection framework for choosing an appropriate methodology for a given project. The author states that having multiple methodologies is appropriate and necessary. The adequate methodology can be chosen by two dimensions: the project staff size and the system criticality. Boehm and Turner [6] extend Cockburn's framework and use risk criteria in order to balance between agile and plan-driven methodologies. This work uses a similar framework in order to select patterns that are appropriate to the context of a given project.

3. Structuring an organizational pattern repository

This section shows how to establish an organizational pattern repository. The repository acts as a knowledge base about software development methodologies. The patterns are selected from the repository in order to resolve certain risks of a software development project. The repository considers alternatives of solutions to the same single problem.

First, the patterns should be described textually using a common format. This work uses a format which is based on James Coplien´s [10] work, and enforces the use of separated fields on the textual descriptions. Each pattern is described by the following fields: *Name*, *Classification*, *Problem*, *Context*, *Forces*, *Solution*, *Relations*, *Rationale* and *Source*.

The patterns are classified using two criteria. This classification is helpful to the process designer when using the selection mechanism. She can search for patterns in all of the categories or may narrow the search for a specific category where she is seeking for advice.

- **By Process Discipline**: the patterns are classified accordingly to the process discipline they tackle. The process disciplines that are used for classification in this work correspond to the Rational Unified Process (RUP) disciplines, because it is a well known and a reasonable complete process [22]: Business Modeling,

Requirements, Analysis and Design, Implementation, Test, Deployment, Configuration and Change Management, Project Management, and Environment.

- **By Mechanism**: an organizational pattern is further classified by the mechanism it uses to describe a solution to the organization of software development work. The mechanisms established in this work are process, role, technique and team values. A process pattern is an organizational pattern that structures its solution as a sequence of activities, i.e. a workflow. A role pattern describes one or more roles that people play in developing software. A technique describes the specific procedures people use to accomplish certain tasks [7]. Some of them apply to a single person (designing a class or test case), while others are aimed at groups of people (code ownership, planning sessions). The team values govern the methodological elements. An example of a team value is the Extreme Programming forty-hour week convention [2].

While there may be other classification possibilities, the two criteria described above have been selected because they represent the target problem of the pattern (process discipline) and the type of solution it describes (mechanism). The tool discussed on section 5 allows the configuration of a user-defined classification criterion for the patterns.

The next section presents a systematic approach for identifying and analyzing the risks of a software project. Section 5 will show how the pattern selection mechanism works. For the selection system to work, the pattern repository has to relate the patterns with the risks they intend to solve. This is achieved through risk resolution rules. A risk resolution rule relates risks with patterns that may be applied for minimizing or solving the risk. Each rule is associated with a project context where it works best, and it has an accuracy factor. A project context is established in terms of three criteria, as further explained on the next section: system criticality and, staff size, as suggested by Cockburn [7], and the team skill level as suggested by Boehm [6]. The accuracy factor (AF) is a 0-10 grade that is used to tune the selection mechanism. It can be adjusted by users of the tool after they have experimented with the patterns. Examples of risk resolution rules are presented on the next section.

4. Analyzing the project risks and its criticality context

Several studies have been conducted in order to identify the main risks of software projects. Some of these studies consider only the risks the process designer has control of, while others consider all risks. In this work we compared and compiled the risks identified by Keil [21], Boehm [4], Addison [1] and the CMMI [27], in order to obtain a comprehensive risk list for software projects. The details of the comparison are further discussed in another article [15].

The risk list for software projects was classified using an adapted criterion from the risk classification framework presented by Keil [21]. Using this classification, the risks can be:

- **Customer risks**: these risks are originated from the involvement of the customer with the system that is being developed.
- **Requirements risks**: one of the most important parts of the process, because if the software does not satisfies its requirements, it is not going to fulfill the customer needs.

- **Planning risks**: these risks monitor the initial planning of the software project.
- **Execution risks**: the execution risks describe situations that may occur while the software is being developed, and may substantially impact the final quality of the software.

The risks compilation is shown on Table 1. In this work, the project management risks have been separated in planning and execution risks, because some of them must be resolved at the beginning of the project, while others must be monitored and tackled during the execution of the project.

The risk list of Table 1 is used in this work as a common checklist of most frequently occurring risks, for identifying the risks of a software project. The risk identification process should not be limited to this list, however, and may consider many more risks. To perform risk identification, team members should be interviewed and group sessions should be held with the project team and other people associated with the project, in order to identify all the things that could go wrong with the project [1]. The result can be a long list of risks which should be organized in the four categories of Customer, Requirements, Planning and Execution. The risk classification may be used with the pattern classification as parameters for the pattern retrieval system, as will be explained in the section 5.

Table 1: The risk checklist

Customer	**Requirements**
Lack of user involvement	Misunderstanding the requirements
Failure to gain user commitment	Scope and goals are not clearly defined
Failure to manage end user expectations	Requirements instability
Conflict between user departments	
Planning	**Execution**
Lack of top management commitment to the project	Introduction of new technology
Lack of required knowledge/skill in the project personnel	Gold plating
Insufficient/inappropriate staffing	Wrong development of functions or user interfaces
Non-realistic schedule and budget	Subcontracting
Lack of a methodology for the project	System use of resources and performance
	Infeasible design

After the risk identification, the risks should be analyzed and prioritized using the risk exposure technique (RE) [15]. The risk exposure, also called risk impact or risk factor, is the product of the probability of a non-satisfactory result to occur, and the loss associated to this non-satisfactory result [5]. This work uses a 0 to 10 scale in order to measure the probability and the loss of each risk. Coppendale [12] suggests some simple rules to determine the risk probability and loss:

- **Probability**: 0 (zero) represents a probability of up to 5% of the risk occurring. 5 represents a probability of about 50%, while 10 represents a probability of 95% or more.

- **Loss**: 0 (zero) represents no increase in the time or in the cost of the project, while 10 represents a very representative impact in the time or in the cost of the project, such as a 100% increase.

When each of the identified risks is quantified, they are ordered by the risk exposure. The process designer should define the criterion for prioritization of the most important risks, establishing the Risk Prioritization Factor (RPF). This means that only the risks with RE >= RPF will receive immediate attention. The pattern systematic retrieval system described in the next section allows for the process designer to configure the required RPF.

After prioritizing the risks, preventive actions should be selected and elaborated [15]. PMT selects organizational patterns from a pattern repository as suggested actions the process designer or the project manager should apply to prevent, resolve or minimize each of the prioritized risks. The pattern selection is driven by the risk analysis results, with the application of risk resolution rules. PMT also uses an analysis of the context of the project, because it is important to consider it when selecting a methodology, as discussed by Cockburn [7] and Boehm [6]. The context of a project is identified in terms of three criteria:

- **Defects criticality**: The possibility of loss associated with the occurrence of a defect. It ranges from a loss of comfort (1) to the loss of many human lives (5). According to Cockburn [7], the more critical a system, the more density is needed on the methodology, i.e. the intermediate artifacts produced must be more carefully detailed, reviewed and checked against each other.
- **Team size**: The number of people involved is also an important factor considered by Cockburn [7]. The larger the project team, the larger the methodology needed, i.e. more intermediate documents must be produced to coordinate the work.
- **Team skill**: Barry Boehm [6] extends Alistair Cockburn´s classification of people in levels of understanding [8], and uses it as an important factor to balance between agile methodologies, such as Extreme Programming [2], and plan-driven methodologies, such as the ones which follow the CMM model [25]. The classification is: *-1. People who are unable or are unwilling to collaborate; 1B. People who can perform on a plan-driven environment, executing simple tasks by following procedural steps, but cannot contribute to an agile team; 1A. People who can contribute both to an agile team and to a plan-driven team; 2. People who can lead a small team of developers and tailor a method to fit a precedented new situation; 3. People who can revise a method, breaking its rules to fit an unprecedented new situation.* Boehm uses the percentage of level 1B people in the project as one factor for balancing the methodology. PMT uses a measure of the team skill by considering the percentage of people classified in all levels of understanding, from 1B to 3. A team skill of 1 means a high skilled team with most people in levels 2 or 3, while a team skill of 5 means most of its staff is in level 1B.

The project criticality context is established in PMT by identifying the values for the three-dimensions shown above. This analysis helps the approach to select patterns that are more likely to succeed in an agile environment and those in a plan-driven one. If the project is near the origin of the three dimensions, meaning a small high skilled team working on a low criticality project, the most appropriate patterns will typically represent practices of agile methodologies. For a project with a high level of defects

criticality, and a large team with less than average skill, the retrieval system will choose organizational patterns that reflect practices of plan-driven methodologies. For projects that do not match any of the extreme opposites in the three-dimensional space, the approach helps to balance the methodology by selecting the patterns that are more adequate to the criticality context of the project and which target its most important risks.

In the next section the pattern retrieval system is examined. The retrieval is based on risk resolution rules, which must be recorded in the repository with the patterns and the risk checklist. Each rule associates one or more patterns with one or more risks they intend to prevent, resolve or minimize. A rule is also associated with the project criticality context where it works best. A rule also has an accuracy factor (AF) which measures how much the associated patterns help to resolve the associated risks, helping to tune the selection system. A factor of 0 (zero) would mean the patterns do not help at all, while a factor of 10 means the patterns can be applied to completely eliminate the associated risks.

Table 2 presents examples of risk resolution rules. The examples of rules and of patterns have been elaborated from practices described in existing published methodologies, from the experience of the authors, and from previous work [15].

Table 2: Examples of risk resolution rules. Each rule is applied in a project context (PC) and has an accuracy factor (AF).

#	Comments	Defects Criticality	Team Size	Team skill
A	Agile characteristics	2 (Discretionary Money)	1 (1-6)	4 (High)
B	Plan-driven characteristics	5 (Many lives)	4 (41-500)	2 (Low)

#	Risk(s)	PC	Pattern(s)	AF
1	Lack of top management commitment to the project	A	*SprintPlanningMeeting* *PlanningGame*	7
2	Lack of top management commitment to the project	B	*SeniorManagementReview*	5

Different rules may target the same risk in different contexts (PC column). As an example, the rule number 1 targets the risk of lack of top management commitment to the project, in the context of a project with agile characteristics – low criticality of defects, and a small and skilled team. The participation of the senior management during iteration planning meetings can be used to minimize this risk [15], such as in the Extreme Programming practice of "Planning Game" [2], or the Scrum practice of "Sprint Planning Meeting" [28]. The rule number 2 targets the same risk in a different context – a project in the plan-driven home ground - with the CMM practice of the senior management reviewing the project commitments with the project manager [25].

5. The Pattern Selection Mechanism

This section describes the systematic selection mechanism implemented on the PMT-Wiki tool. The tool records and maintains a pattern repository with the characteristics presented on section 3. It also helps with the risk identification, risk exposure analysis

and prioritization, and the project context analysis, as described in section 4. The tool uses the repository, the project analysis results, and the risk resolution rules in order to perform the selection of patterns, resulting on a suggested list of organizational patterns that the process designer may apply in order to tackle the risks of the project. The pattern selection is complemented with exploratory browsing on the pattern repository.

The PMT-Tool tool is implemented in Java using a MySQL relational database to store its data. The user interface is all web-based, facilitating the exploratory browsing of the organizational patterns, because the relationships between the patterns are navigated through hypertext links.

The retrieval system assumes the pattern repository – called the base pattern language - is already recorded. The organizational patterns in the base pattern language have a textual description, a classification and are related to each other. There is also: a recorded list of risks, a list with different combinations of project criticality contexts, and the risk resolution rules.

A retrieval session may start with the process designer recording a new project on the tool, or using a previously recorded project. For a new project, the project context must be established and the risks must be identified and prioritized using the risk exposure and the risk prioritization factor (RPF), as further explained in section 4. The risk and context information may be updated for an existing project before a new retrieval session starts.

The selection system is configured with parameters. The risk prioritization factor (RPF) is mandatory. As explained on section 4, the risks of the project are identified and are ordered by the risk exposure (RE), but only the risks with a RE value greater than the RPF are considered by the selection mechanism. The other parameters use the classification of the patterns and of the risks, as described on sections 3 and 4, and are optional. The patterns are classified by the target process discipline, the pattern solution mechanism type, and possibly a user-defined classification. The risk type can also be used as a parameter to the selection system. The classification parameters help the process designer to narrow the search to solve a problem in a specific area where he or she is seeking for advice.

The first step of the pattern selection algorithm is to evaluate each of the risk resolution rules that are associated to the prioritized risks. For each rule, two factors are evaluated:

- **Risk exposure**: the average risk exposure for the risks associated with the rule. The risk exposure is a 0 to 100 value, because it is the probability (0-10) multiplied by the loss (0-10).
- **Context proximity**: the rule is associated to a project context where it works best, but this context may be different than the context of the target project. A project context is evaluated using three criteria (a 1-5 value each), as explained in section 4: the system criticality, the team size and the team skill. The context proximity evaluates how close the context of the rule is to the context of the project. The proximity is calculated using the formula $p = 100 - (\sqrt[2]{(d1^2+d2^2+d3^2)} * 100/8)$, where *d1*, *d2* and *d3* are the distance from the context of the rule to the context of the project in each axis, ranging from 0 to 4, because the minimum value for each criterion is 1 and the maximum value is 5. The formula uses the Pythagoras theorem to calculate the distance in the three dimensional space, and converts the result to a value in the range 0-100. The result is rounded down to the nearest integer value. An

exact match (zero distance) would mean proximity of 100, while the maximum distance *(d1=d2=d3=4)* will result in zero (0) proximity.

The two factors above are evaluated for each rule, resulting in a rule evaluation table, as exemplified by Table 3. The resulting score for each rule is calculated by multiplying the risk exposure value, the context proximity value and the accuracy factor of the rule. The resulting table is ordered by the scores of the rules. The target result, however, is not a list of rules, but a list of patterns ordered by relevance of application to the given project. Therefore, the next step of the retrieval algorithm is to create a pattern comparison table.

Table 3: Calculating the resulting score for each risk resolution rule

Rule #	Exposure (0-100)	Proximity (0-100)	AF (0-10)	Score (0-100000)
1	65	59	7	26,845
2	82	23	6	11,316
3	91	68	8	49,504

A pattern comparison table lists all the patterns that are associated to at least one risk resolution rule, and the resulting score of the pattern. For patterns that appear only once on the rules list, the score of the pattern is the score of the associated rule. When a pattern appears in more than one of the applied rules, the resulting score for the pattern is the sum of scores of the rules in which it appears. The resulting table of patterns, as exemplified by Table 4, is ordered by the score column and presented to the process designer in descending order - the top scores appear first.

Following a hypertext link on the pattern name, the user may navigate to the complete textual description of each pattern. Another hypertext link exists on the score column, which details the information about the score of the corresponding pattern, such as which rules were applied, which risks the pattern is intended to solve, and how the score was calculated.

Table 4: An example of a resulting list of suggested patterns for the process designer

Pattern	Score (0-100,000)
EarlyAndRegularDelivery	89,023
SprintPlanningMeeting	49,302
ScenariosDefineProblem	11,983

A retrieval session is executed on the context of a specific software development project. The tool creates a separated workspace for the process designer to elaborate an organizational pattern language for the project. This workspace is named in PMT the project pattern language, which is a separated system of patterns. It helps the process designer to document the methodology of the project in the form of organizational patterns. The pattern repository is named the base pattern language. After a retrieval session, the process designer may select some of the suggested patterns for inclusion on the project pattern language. A pattern relationship will also be transferred, in case both of the patterns it relates are included on the target pattern language.

After the project workspace (the project pattern language) is populated with patterns, new retrieval sessions may be executed. The process designer may experiment with different parameters for the selection, such as decreasing the risk prioritization factor

(RPF) in order to consider more risks to be targeted. The patterns that have already been included in the project pattern language will not be considered for selection on further retrieval sessions.

The retrieval mechanism of the PMT-Wiki tool is complemented with exploratory browsing on the pattern repository. The process designer can navigate on the repository using hypertext links representing the relationships between the patterns, and empirically selecting patterns for inclusion on the project pattern language.

Another possibility is the selection of all the patterns which belong to the same pattern source. This operation is helpful when the process designer wants to use all the patterns of a given methodology as a start, such as the Scrum patterns [3]. After populating the project pattern language with them, the process designer may complement the methodology by targeting the risks that are still unaddressed with a pattern retrieval session.

When the process designer believes she has completed tailoring the project pattern language, she will allow the project team members to access it. Most of the project team members will browse only through the patterns and the relationships that exist on the separated workspace of the project, and will not need to know the existence of the pattern repository. They might use the patterns as a source of learning in order to solve the particular problems that occurs in the software development project. The process designer will continue to monitor the risks of the project and may execute new retrieval sessions and publish new patterns in the project pattern language in the case it shows to be necessary.

6. A Case Study with the PMT-Tool

This section of the paper explores the PMT-Tool further, using a case study as an example. The approach helped the design of a methodology to be applied by an offshore software development organization. The case study was held at a subsidiary of a large information technology company. The impressions of the application of the approach and of the tool are discussed at the end of this section.

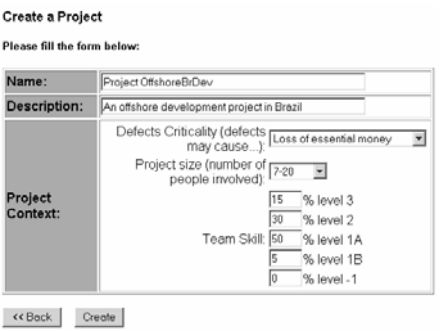


Figure 1: Creating the project

The approach was applied to aid the definition of the software development methodology for a small-to-medium project (12 people). The development team was distributed across two countries, with a project manager and three business analysts

located in the United States, and a development manager and seven programmer/analysts located in Brazil. The project was registered in the PMT-Tool, which aided with the assessment of the project criticality context (Figure 1). The resulting criticality context is: *defects may cause loss of essential money (3), size is small to medium (2), with a highly skilled team (2)*.

The next step was to perform risk identification, risk exposure analysis, and risk prioritization. These activities were executed by the project manager with the help of key project members. The risk checklist of section 4 was the primary source of risks. Figure 2 shows the risks which were identified for the project. After the risk exposure analysis was finished, the project manager decided to prioritize the risks with risk exposure higher than 40. He reasoned that the costs of managing risks with risk exposure lower than 40 were too high as compared to the small effort of dealing with them in the case they happen to materialize during the project.

The risk of *Introduction of new technology* was assessed as the most important risk of the project, because most of the Brazilian developers were new to the technology – they had just been hired by the company. Due to the geographically dispersed nature of the project, the risks of *Lack of user involvement* and of *Difficulty of communication* were also highly scored.

The organization had a history for *Conflict between user departments* and for *Failure to gain user commitment*, and these risks were also prioritized. The risks *Lack of a methodology for the project, Defects coming back* and *Misunderstanding the requirements* were also prioritized, because the project was new, the development methodology was still immature, and the developers were not used with the application domain.

The next step was to create a pattern language for the project, which was initially empty. The task of designing a methodology using the PMT-Tool consists of selecting patterns from the base pattern language (the repository), and adding them to the project pattern language. The selection is performed by the process engineer, who uses the selection mechanism of the tool to help him to select the appropriate patterns.

RPF - Risk Prioritization Factor (0-100)*: 40

Risk	Risk Category	Probability (0-10)	Loss (0-10)	Risk Exposure (0-100)
Introduction of new technology	**Execution**	10	8	**80**
Lack of user involvement	**Customer**	8	9	**72**
Dificulty of comunication/conflict between diferent functions (analyst, programmer, tester)	**Execution**	8	8	**64**
Defects coming back	**Execution**	9	7	**63**
Lack of a methodology for the project	**Planning**	8	7	**56**
Conflict between user departments	**Customer**	7	7	**49**
Failure to gain user commitment	**Customer**	6	8	**48**
Misunderstading the requirements	**Requirements**	6	7	**42**
Requirements instability	Requirements	5	7	35
Gold plating	Execution	4	6	24
Wrong development of functions or user interfaces	Execution	3	6	18

Figure 2: Risk Exposure Analysis

The initial pattern retrieval session results are shown in Figure 3. The patterns with the left checkbox marked were added to the project pattern language, after exploratory browsing on their textual descriptions.

Figure 4 shows an example of the score detail for the pattern *EarlyAndRegularDelivery*. This pattern had a high score because two risk resolution rules could be applied which included it as the solution. The highest scored rule applied the pattern for minimizing the risk of *Introduction of new technology.* Another example is the selection of the pattern *ProductOwner*. This pattern was selected in order to resolve the risk of *Conflicts between user departments*. The patterns *ScrumTeam* and *HolisticDiversity* – organizing the project in small, cross-functional, cross-geographical teams - were selected to minimize the risks of *Difficulty of communication* and of *Lack of user involvement*.

☐	Pattern	Score
☑	ChangesToWorkProductsAreControlled	79695
☑	EarlyAndRegularDelivery	76600
☑	ScrumTeam	71504
☑	HolisticDiversity	71504
☐	CustomerAlwaysAvailable	53136
☑	RegressionTesting	41328
☑	StatusIsRecorded	41328
☐	DoTheSimplestThing	39360
☐	PairProgramming	39360
☑	SoftwareConfigurationManagement	38367
☑	Inspections	38367
☑	ProductOwner	32144
☑	ScenariosDefineProblem	26712
☐	RequirementsAreValidated	14112
☐	DocumentedConfigurationManagementPlan	14112
☐	SoftwareLifeCycleIsDefined	13776

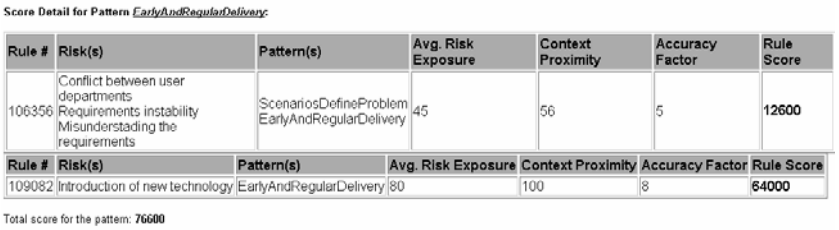
Figure 3: The suggested list of patterns

The last patterns represented processes and techniques found in modern agile methodologies. Other patterns were selected which represented more rigorous processes, such as *StatusIsRecorded, ChangesToWorkProductsAreControlled*, and *SoftwareConfigurationManagement.* These patterns were elaborated from CMM [25] Software Configuration Management (SCM) key process area, and were used to minimize the risks of *Defects coming back* and of *Lack of a methodology for the project*. The technique of *Inspections* was also selected to minimize these risks. A factor contributed to the selection of these last patterns, which is the defects criticality of the project – *defects may cause the loss of essential money*.

After the initial selection of patterns was made by the process designer, with the aid of the systematic retrieval system of the PMT-Tool, more patterns were added to the project pattern language. The process designer performed exploratory browsing on the pattern textual descriptions and the pattern relationships. He evaluated other patterns which were not included in the tool suggestions, and completed the project pattern language with it. Other Extreme Programming [2] engineering practices were included, such as *UnitTests*, *WhenABugIsFound*, *IntegrateOften*, and *CollectiveCodeOwnership*. The management practices were completed with Scrum's [3] *Backlog*, James Coplien's [9] *CompensateSuccess*, and XP's [2] *DailyStandupMeeting*.

The tool web interface was considered easy to use by the project manager who experimented with it. The exploratory browsing on the base pattern language was made easy through the use of hypertext links. The pattern list can be filtered by either the

pattern mechanism type (Role, Value, Technique, or Process), by the process discipline, or by the pattern source. From one pattern, the user can easily navigate to the related patterns by clicking in one of the pattern relationships.



Score Detail for Pattern *EarlyAndRegularDelivery*:

Rule #	Risk(s)	Pattern(s)	Avg. Risk Exposure	Context Proximity	Accuracy Factor	Rule Score
106356	Conflict between user departments Requirements instability Misunderstading the requirements	ScenariosDefineProblem EarlyAndRegularDelivery	45	56	5	**12600**

Rule #	Risk(s)	Pattern(s)	Avg. Risk Exposure	Context Proximity	Accuracy Factor	Rule Score
109082	Introduction of new technology	EarlyAndRegularDelivery	80	100	8	**64000**

Total score for the pattern: **76600**

Figure 4: Score detail example

The impression with the use of the tool was reported as being satisfactory. Clearly there is still a lot of work to be performed by the process engineer. She still has to browse through the pattern textual descriptions, and use her experience and careful consideration when designing the methodology to be deployed. But she can count on a systematic framework – built on past experiences – to aid her on this complex task. The base pattern language consists of patterns extracted from several sources, forming a comprehensible knowledge base about software development. Of course this repository is not complete and has to be checked and updated. By starting to fill the project pattern language with patterns which were targeted to minimize the main project risks, the process designer saved her time – she didn't had to browse though dozens of patterns. By performing the project criticality context and the risks analysis prior to performing the selection, she had to reason about the project risks and about how to deal with them. The methodology design was driven by the project risks, either by counting on the knowledge recorded in the PMT repository, or through the empirical work of the process designer.

After the patterns had been selected by the process designer, the resulting pattern language was complemented with other types of documents, such as sample documents for use cases scenarios and test scripts, and also standards for coding and testing. Specific configuration management procedures which included project specific information, such as machines, passwords, and version control software specific commands, were created to complete the methodology.

7. Summary and conclusions

This work presented *Pattern-based Methodology Tailoring*, an approach for aiding the adaptation of development methodologies to the needs of a particular software project, using known patterns and risk criteria. It structures a pattern repository as a knowledge base about successful practices for managing the work of software projects. The process designer performs an analysis of the risks and of the criticality context of the project. The selection mechanism implemented in the PMT-Tool then suggests a list of suitable patterns for the project which target the identified risks and context.

The selection mechanism of the PMT-Tool helps the process designer on performing the methodology adaptation. However, the process designer role is fundamental, and her careful empirical consideration and evaluation is necessary in order to tailor an adequate

methodology for a given project. The tool is helpful because it suggests the most suitable patterns to the project, accordingly to the data which is recorded in its repository. It saves the time of the process designer to browse through dozens or even hundreds of patterns which are available. It also influences her choices by selecting patterns which target the project risks that she has assessed. This work provides a structured framework for the process designer to think about the risks and the criticality context of the project before designing the methodology, and to design the methodology using organizational patterns as a form of documentation.

The PMT approach can be helpful to mix different techniques and processes of software development. It guides the process designer on identifying the project criticality context, which can be used as a framework to balance between agile and plan-driven methodologies. Both agile and planned approaches have situation dependent shortcomings that, if left unaddressed, can lead to project failure. The challenge is to balance the two approaches to take advantage of their strengths in a given situation while compensating for their weakness [6].

Organizational patterns can be an efficient documentation form, by describing the essential part of a solution to a recurring problem, which can be adapted to solve a particular problem of a software project. They may avoid process documents from becoming inefficient paperwork that do not reflect the process that are actually performed, have descriptions that are ambiguous or incomprehensible, or are too high level to be used in practice, which is a situation found on many organizations [30].

The pattern repository should be elaborated from existing sources of organizational patterns, such as Coplien's [9] [11], Harrison's [17], Beedle et al Scrum org. patterns [3], DeLano and Rising test pattern language [13] and Stevens and Pooley systems reengineering patterns [29]. It should also be complemented with the knowledge documented on published software development methodologies, such as XP [2], RUP [26] and the CMM [25]. Patterns may also be obtained by documenting the experience of software project managers or process engineers on solving recurring problems. However, careful consideration is important when adding a new pattern to the repository, in order to maintain its quality as a source of meaningful and efficient information.

Special consideration is also necessary when adding new risk resolution rules to the repository, which are used by the selection system to choose the patterns which are most suitable to tackle the risks of the chosen project. The rules should also be elaborated from the experience of project managers on resolving project risks, or from the existing knowledge published on the literature. The risk resolution rules should be adjusted and tuned with the experimentation of the approach. The feedback of process designers should be collected and evaluated in order to improve the knowledge of the PMT repository.

This work is an ongoing project. There are a number of limitations that have to be overcome, as risk metrics, metrics collection, adaptation mechanisms for the standard process in use by an organization, evaluation of recommended practices for given risks, and other points. Future work may include using the GQM (*Goal-Question-Metric*) approach [15] for monitoring the risks during the execution of the project and suggesting corrective actions. The authors are working on a tool for collecting data and using it for evaluating and improving the suggested risk resolution rules and to plot the tendency of evolution of the risks.

Other extension is the use of meta-models to represent both the patterns and the processes. This comes with the idea of considering each project resulting own process as an adaptation of the standard process of the organization. The use of a standard process is helpful to large organizations, to minimize the costs of culture change, help with the collection and analysis of metrics, ease the training of new staff, and many other reasons. The meta-model is used as a common representation for the processes activities, deliverables, roles, and workflows. A project specific process is tailored by the combination of the organization's standard process with the selected patterns for the project. The goal of the selection is to minimize the risks of the project.

The approach presented in this paper is limited by the difficulties of validating the empirical knowledge of experienced process designers, software engineers and project managers, expressed as patterns and risk resolution rules. Nevertheless, it has a strong and original contribution to structure a systematic approach for capturing this knowledge and assisting process designers and project managers on leveraging it. The approach here reported suggests that the use of risk analysis combined with organizational patterns is a promising way of overcoming the limitations of existing software process improvement frameworks.

8. References

[1] Addison, T., Vallabh, S. *Controlling Software Project Risks – An Empirical Study of Methods used by Experienced Project Managers*. Proceedings of the South African Institute of Computer Scientists & Information Technologists, Port Elizabeth, South Africa, September 2002, p.128-140.

[2] Beck, Kent. *Embracing Change with Extreme Programming*. IEEE Computer, 32:70--77, Oct. 1999.

[3] Beedle, M. et al. *SCRUM: an extension pattern language for hyper-productive software development*. In: Pattern Languages of Program Design 4. Addison-Wesley, 1999.

[4] Boehm, B. *Software Risk Management: Principles and Practices*. IEEE Software, v.8, n.1, p. 32-41, January 1991.

[5] Boehm, Barry. *Get Ready for Agile Methods, with Care*. IEEE Computer, v.35, n.1, p. 64-69, January 2002.

[6] Boehm, Barry; Turner, Richard. *Using Risk to Balance Agile and Plan-Driven Methods*. IEEE Computer, June 2003.

[7] Cockburn, Alistair. *Selecting a Project's Methodology*. IEEE Software, July/August 2000.

[8] Cockburn, Alistair. *Agile Software Development*. Addison-Wesley, 2001.

[9] Coplien, J. *A Development Process Generative Pattern Language*. In: James Coplien, Douglas Schmidt, eds. Pattern Languages of Program Design. Addison-Wesley, 1995.

[10] Coplien, James. *Sofware Patterns*. Originally published by SIGS Books and Multimedia. http://www1.bell-labs.com/user/cope/Patterns/WhitePaper/. 1996.

[11] Coplien, James. *Organizational Patterns web site*, as viewed in August 7th, 2004. http://www1.bell-labs.com/user/cope/Patterns/Process/OrgPatternsMap.html

[12] Coppendale, J. *Managing Risk in Product and Process Development and Avoid Unpleasant Surprises*. Engineering Management Journal, v.5, n.1, p. 35-38, February, 1995.

[13] DeLano, David E., Rising, Linda. *System Test Pattern Language*. Last visited in August 7th, 2004.
http://www.agcs.com/supportv2/techpapers/patterns/papers/systestp.htm
[14] Devedzic, Vladan. *Software Patterns*. in: Chang, S.K. (ed.), "Handbook of Software Engineering and Knowledge Engineering Vol.2 – Emerging Technologies", World Scientific Publishing Co., Singapore, 2002, pp. 645-671.
[15] Fontoura, Lisandra M., Price, Roberto T. *Usando GQM para Gerenciar Riscos em Projetos de Software*. XVIII Brazilian Simposium on Software Engineering, 2004.
[16] Hall, E. *Managing Risk: Methods for Software Systems Development*. New York:Addison-Wesley, 1998.
[17] Harrison, Neil. *Organizational Patterns for Teams*, In John Vlissides, James Coplien and Norm Kerth, eds., Pattern Languages of Program Design 2. Addison-Wesley, 1996.
[18] Henderson-Sellers, B. and Mellor, S. *Tailoring process-focussed OO methods*. JOOP/ROAD, 12(4), 1999.
[19] *The Hibernate project web site***.** Last visited on August 15th, 2004. http://www.hibernate.org
[20] Jacobson, Ivar; Booch, Grady; Rumbaugh, James. *The Unified Software Development Process*. Addyson Wesley Professional, 1999.
[21] Keil, M. et al. *A Framework for Identifying Software Project Risks*. Communications of the ACM, v. 41, n.11, p. 76-83, November 1998.
[22] Manzoni, Lisandra V.; Price, Roberto T. *Identifying Extensions Required by RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2 and 3*. IEEE Transactions on Software Engineering, February, 2003.
[23] Martin, James. *Recommended Diagramming Standards for Analysts and Programmers*. Prentice-Hall, 1987.
[24] Palmer, Steven, and Felsing, John. *A Practical Guide to Feature-Driven Development*. The Coad Series. 2002.
[25] Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Addison-Wesley, 1995.
[26] Rational Software Corporation. *Rational Unified Process*, v. 2001. Cupertino, 2001.
[27] Software Engineering Institute. *Capability Maturity Model Integration (CMMI)*, Version 1.1, 2002.
[28] Schwaber, K., Beedle, M. *Agile Software Development with Scrum*. Upper Saddle River: Prentice Hall, 2001.
[29] Stevens, P., Pooley, R.. *Systems Reengineering Patterns*. Proceedings ACM-SIGSOFT, 6th International Symposium on the Foundations of Software Engineering, pp.17-23, 1998.
[30] Vasconcelos, F.M de; Werner, C.M.L.; *Organizing the Software Development Process Knowledge: An Approach Based on Patterns*; Int. Journal of Software Eng. & Knowledge Eng., World Scientific Publishing Company (Vol. 8, n°. 4, 1998)