

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <<http://www.upgrade-cepis.org>>

UPGRADE is the anchor point for UPENET (UPGRADE European NETWORK), the network of CEPIs member societies' publications, that currently includes the following ones:

- Mondo Digitale, digital journal from the Italian CEPIs society AICA
- Novática, journal from the Spanish CEPIs society ATI
- Piroforiki, journal from the Cyprus CEPIs society CCS
- Pro Dialog, journal from the Polish CEPIs society PTI-PIPS

Publisher

UPGRADE is published on behalf of CEPIs (Council of European Professional Informatics Societies, <<http://www.cepis.org/>>) by Novática

<<http://www.ati.es/novatica/>>, journal of the Spanish CEPIs society ATI (Asociación de Técnicos de Informática <<http://www.ati.es/>>).

UPGRADE is also published in Spanish (full issue printed, some articles online) by Novática, and in Italian (abstracts and some articles online) by the Italian CEPIs society ALSI

<<http://www.alsi.it/>> and the Italian IT portal Tecnoteca <<http://www.tecnoteca.it/>>.

UPGRADE was created in October 2000 by CEPIs and was first published by Novática and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <<http://www.svifsi.ch/>>).

Editorial Team

Chief Editor: Rafael Fernández Calvo, Spain, <rfoalvo@ati.es>
Associate Editors:

- François Louis Nicolet, Switzerland, <nicolet@acm.org>
- Roberto Carniel, Italy, <carniel@dgt.uniud.it>
- Zakaria Maamar, Arab Emirates, <Zakaria.Maamar@zu.ac.ae>
- Soraya Kouadri Mostéfaoui, Switzerland, <soraya.kouadrimostefaoui@unifr.ch>

Editorial Board

Prof. Wolfgang Stucky, CEPIs Past President
Prof. Nello Scarabottolo, CEPIs Vice President
Fernando Piera Gómez and Rafael Fernández Calvo, ATI (Spain)
François Louis Nicolet, SI (Switzerland)
Roberto Carniel, ALSI – Tecnoteca (Italy)

UPENET Advisory Board

Franco Filippazzi (Mondo Digitale, Italy)
Rafael Fernández Calvo (Novática, Spain)
Panicos Masouras (Piroforiki, Cyprus)
Andrzej Marciniak (Pro Dialog, Poland)

English Editors: Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green, Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson.

Cover page designed by Antonio Crespo Foix, © ATI 2004

Layout: Pascale Schürmann

Editorial correspondence: see "Editorial Team" above

Advertising correspondence: <novatica@ati.es>

Upgrade Newsletter available at

<<http://www.upgrade-cepis.org/pages/editinfo.html#newsletter>>

Copyright

© Novática 2004 (for the monograph and the cover page)

© CEPIs 2004 (for the sections MOSAIC and UPENET)

All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, contact the Editorial Team.

The opinions expressed by the authors are their exclusive responsibility.

ISSN 1684-5285

**Next issue (December 2004):
"Cryptography"**

(The full schedule of UPGRADE is available at our website)

- 2 Editorial
Four Years of UPGRADE

SPT, Software Process Technology

Guest Editors: Francisco Ruiz-González and Gerardo Canfora

Joint monograph with Novática*

- 3 Presentation
Software Process Technology: Improving Software Project Management and Product Quality – *Francisco Ruiz-González and Gerardo Canfora*
- 6 Software Process: Characteristics, Technology and Environments – *Francisco Ruiz-González and Gerardo Canfora*
- 11 Key Issues and New Challenges in Software Process Technology – *Jean-Claude Derniame and Flavio Oquendo*
- 17 A Taxonomy of Software Engineering Environment Services: The Upcoming ISO/IEC Standard 15940 – *Dan Hyung Lee and Juan Garbajosa-Sopeña*
- 22 Open Source and Free Software: A New Model for The Software Development Process? – *Alfonso Fuggetta*
- 27 Applying The Basic Principles of Model Engineering to The Field of Process Engineering – *Jean Bézivin and Erwan Breton*
- 34 Software Process Modelling Languages Based on UML – *Pere Botella i López, Xavier Franch-Gutiérrez, and Josep M. Ribó-Balust*
- 40 Supporting the Software Process in A Process-centred Software Engineering Environment – *Hans-Ulrich Kobialka*
- 47 Managing Distributed Projects in GENESIS – *Lerina Aversano, Andrea De Lucia, Matteo Gaeta, Pierluigi Ritrovato, and Maria-Luisa Villani*
- 53 Software Process Measurement – *Félix García-Rubio, Francisco Ruiz-González, and Mario Piattini-Velthuis*
- 59 Process Diversity and how Practitioners Can Manage It – *Danilo Caivano and Corrado Aaron Visaggio*

MOSAIC

- 67 Data Architecture
A Disquisition on The Performance Behaviour of Binary Search Tree Data Structures – *Dominique A. Heger*
- 75 News & Events: News from CEPIs and EUCIP; SCI 2005 (Call for Papers)

UPENET (UPGRADE European NETWORK)

- 77 From **Novática** (Spain):
IT and Disabilities
Braille and The Pleasure of Reading: We Blind People Want to Continue Reading with Our Fingers – *Carmen Bonet-Borrás*
- 84 From **Piroforiki** (Cyprus):
Information Technology in Today's Organizations
Is the IT Productivity Paradox Resolved? – *Kyriakos E. Georgiou*

* This monograph will be also published in Spanish (full issue printed; summary, abstracts and some articles online) by **Novática**, journal of the Spanish CEPIs society ATI (*Asociación de Técnicos de Informática*) at <<http://www.ati.es/novatica/>>, and in Italian (online edition only, containing summary abstracts and some articles) by the Italian CEPIs society ALSI (*Associazione nazionale Laureati in Scienze dell'informazione e Informatica*) and the Italian IT portal Tecnoteca at <<http://www.tecnoteca.it/>>.

Process Diversity and how Practitioners Can Manage It

Danilo Caivano and Corrado Aaron Visaggio

Since IT projects are unique regarding their combination of specific goals, technologies in use, and characteristics, providing 'general' processes it is not an effective solution. Instead effective and efficient processes custom tailored to a project and based on experience collected during past projects execution are required. This is in contrast with the industry practices where reuse-oriented process descriptions and goal-oriented planning are often missing. Usually a process can undergo a certain numbers of modifications, due to the different operative contexts in which it is executed. The modifications generate many different versions of the process, named specialized processes. Each one of these must be managed properly in order to govern a just evolution consistently with all the others. Considered the dimension of the actual scenarios, maintaining all the processes and their specialized versions is not a trivial task. We have defined a process pattern based framework to accomplish this purpose. In this paper we present the framework, that we are realizing with an Italian enterprise, and an explanatory case study we are developing within the Research Centre on Software Technology in Bari, Italy.

Keywords: Empirical Software Engineering, Software Engineering, Software Process Management, Statistical Process Control.

1 Introduction

The literature proposes many kinds of processes ranging from business processes to software development and maintenance process.

These definitions describe processes at a coarse grain: they formalize a paradigm, as a general solution to a problem. These directives must be properly developed by designing processes aiming at fulfilling specific constraints and business goals. This operation is usually named *specialization or customization of Software Process*. When dealing with process specialization, three kinds of concerns should be addressed.

The first one regards the *culture of the organization*. With culture we mean the way activities are usually performed in the Organizations, due to: Quality System, methodology adopted, technologies used, coding standards, kinds of documentation produced.

These factors can affect the design of one process much in depth, up to significantly change definition and sequence of activities also if well and precisely codified. E.g., the unit test can be performed in many different ways if considering different testing tools, different artifacts for reporting test beds and results, procedures to calculate the paths to be tested and guidelines in use. Consequently, the same task (e.g., unit test) can be reached with different sub-processes.

Another common concern should be taken into account: the *re-use of sub-processes*. Large enterprises usually exploit hundreds of processes. A typical situation is that many departments execute the same sub process without sharing any information about it, such as: the process model, results of monitor-

ing, improvement initiatives. As an effect, the evolution of the same sub process follows different directions in the same organization, with a clear disadvantage. The solution for this problem is a formal sharing of the process knowledge, so that each department can benefit from the experiences realized in the other departments.

Danilo Caivano received his BSc in Computer Science and his PhD degree in Software Engineering from the *Università degli Studi di Bari*, Italy. He is affiliated to the Software Engineering Research Laboratory, <<http://serlab.di.uniba.it>>, and Research Centre On Software Technology – BARI. He is Assistant Professor at the Department of Informatics, University of Bari. He has also been consultant for many small to medium companies, where he has carried out both empirical investigations for validating research results in industrial contexts, and technological transfer through pilot projects. His research interests fall in the area of software process management and improvement within co-located and distributed contexts, software estimation, and empirical software engineering. At the moment he is involved in a research project for evaluating the efficacy of Statistical Process Control as a means for monitoring geographically distributed software processes. <caivano@di.uniba.it>

Corrado Aaron Visaggio obtained his BSc in Electronic Engineering at the *Politecnico* of Bari, Italy, in 2001. He developed his Master Thesis at the *Fraunhofer IESE*, Germany. In 2002 he started attending PhD courses in Software Engineering at *Università degli Studi del Sannio*, in Benevento, Italy. Currently he is working as researcher at the Research Centre of Software Technology (RCOST), in Benevento. His main topics of research are: software process modelling and management, agile methodologies for developing software, and knowledge management in Software Engineering. <visaggio@unisannio.it>

A third concern regards the *technology used for managing the process*. Process management is supported by different kinds of technologies targeted to validate, simulate, and activate the related workflow. In the same organization on the same process different technologies can be used for the same functionality. E.g., different workflow engines, different modelling tools, different simulators.

The three concerns are related to *Process Diversity* [1][17]. As pointed out in [2][18], many factors, from here on called “*diversity factors*”, are essential components in forming process diversity, and affect process management: business environment, technology, industrial standard, quality program, vision, budget, size, structure and culture of organization. By coupling each diversity factor with the appropriate values, the actual context can be rigorously defined. This can be referred as the “*profile of an operative context*” (from here on referred to as context profile for conciseness).

Theoretically, each context profile requires a custom tailored process, but this implies an understanding of process variations and knowledge about when to use which process variants. In fact, according to [18]:

- each process alternative needs to be elicited and explicitly described;
- process alternatives need to be characterized and constraints/rules on their use need to be formulated;
- a characterization of the context is needed in order to be able to select a process variant.

A framework for managing process models in highly variable context profiles and for accomplishing reuse of experience acquired in previous process modelling cases is needed as a means to make changes in a safe and economic way.

In order to address these needs we have developed ProMisE. The key ideas are:

1. to use a framework based on process patterns (a tuple problem-solutions) to manage process diversity and thus to face process customization in an effective way. Given a problem, that is a product or a service to be delivered, the pattern allows process engineer to associate a family of processes, each of them corresponding to a process variant of the root-process, as its solution. The root-process describes the more generic solution. On the other hand, each variant represents a specialization of the root-process with respect to a specific context profile; the overall set of variants encloses those ones that have been experimented till the current moment.
2. to define a package of functions for managing the repository of the defined patterns. The package contains the functions mainly corresponding to the phases of the process pattern management process and spans its entire lifecycle.

The framework was presented in [3] and now the authors are driving the realization of its technological support within a company operating in the field of Enterprise Resource Planning. In this paper we will present the framework together with *an explanatory case study we are developing within Research Centre on Software Technology, Bari, Italy*. We will also discuss the problems we are coping with during its production. The paper proceeds as follows: Section 2 presents some

related work, Section 3 introduces the framework, Section 4 discusses an exemplar application of the framework, and, finally, Section 5 draws conclusions.

2 Related Work

Some researchers have been focusing their attention on the “*pattern concept*”, trying to define a complete description [4][5]. In [6] the authors give a definition for a software pattern: it “*identifies a recurring problem and a solution, describing them in a particular context to help developers understand how to create an appropriate solution. Patterns aim to capture and explicitly state abstract problem-solving knowledge that is usually implicit and gained only through experience.*”

In [7] the authors propose an approach to structure and store experience in order to enable effective reuse: “*The main idea of this approach is a rearrangement and reprocessing of captured experience into quality patterns which are based on a problem-solution strategy.*”

As shown in [8][9][10], mostly patterns are applied for supporting software development lifecycle.

This paper refers to the same well known concept of pattern and applies it to processes instead of products.

As shown in the following, other authors have studied how diversity factors affect process designing and management, highlighting the need for properly customizing the software models rather than designing new ones as context profile changes.

In [11] Sutton states “*The ability of repeating a process can critically affect a start-up’s success*”. The author refers to a start-up company producing a family of products that the company treats more or less similarly. Repeatable processes span over the life cycle, including development, quality assurance, documentation, and training. “*In such case specializing readily processes and test plans is more useful than customizing processes and plans for each product-family member. You can apply some technologies in various ways in many contexts and reuse them as the organization and its processes evolve.*” Sutton doesn’t recommend adopting procedures, technologies, and protocols for one product or life cycle phase unless special needs exist or they can be reused for other products or phases.

In [12] how process diversity affects the field of software reuse is explained. The authors have focused on the reuse processes applied to four companies; during this work they have identified the reuse process characteristics for each company: reuse approach, reuse technology, reuse processes and roles, which develop assets, when assets are developed, reuse processes added, non-reuse processes modified. All these parameters can be used to tailor a reuse process to the particular organization according to its specific characteristics.

In [17] several articles are presented that show how process diversity affects software maintenance and the need for customizing maintenance process to context characteristics.

Finally in [18], the author presents a tool-based technique for customize a process model to project constraints.

All these works motivate researchers and practitioners to further investigate this area of interest.

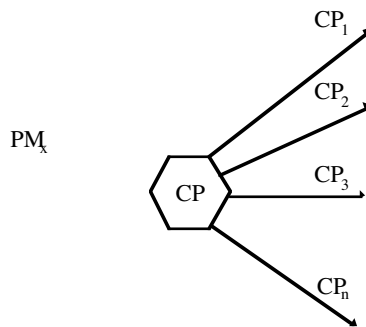


Figure 1: Specializing the Process Model.

3 ProMisE Framework

The ProMisE framework can be summarized as follows: given a process PM_x , from here on called *root-process*, a specialized process SP_{xj} is defined as $SP_{xj} = f(PM_x, CP_j)$; where $f()$ is the mechanism, CP_j is the context profile dictating specialization needs and the consequent specialization actions. Each specialized process is a variant of the correspondent process model. This was necessary in order to adapt the latter to a specific context profile. A context profile is a set of instantiated diversity factors DF_i . The generic DF_i where $i=1 \dots n$ is a diversity factor having a definition domain $[DF_i] = \{FI_{i1}, FI_{i2}, FI_{i3}, \dots, FI_{ik}\}$. FI_{ih} where $h=1 \dots k$ is a factor instance of $[DF_i]$. Let's define a generic characterization of CP , indicating $CP_j = \{[DF_1]_j, [DF_2]_j, [DF_3]_j, \dots, [DF_n]_j\}$ where $[DF_i]_j$ is a factor instance $FI_{ih} \in [DF_i]$ with $n =$ number of diversity factors included in CP_j . After that, the concept of a *context profile* can be generalized as $CP = \{CP_1, CP_2, CP_3, \dots, CP_N\}$ where $\forall i, j \in [1, N] : CP_i \neq CP_j$.

This characterization allows to explicit and formalize the diversity among the processes that usually result as implicit information within the same processes.

3.1 The Framework

The framework consists mainly of a collection of process patterns (association between a problem and a family of solutions) and a package of functions to manipulate the process patterns.

It provides the process engineer with a mechanism that, given a starting problem, allows to:

1. choose the candidate root-process model solving the problem when present in the dedicate database;
2. identify all the process model's variables allowing the process model be specialized, if necessary, to diverse context profiles;
3. guide the process engineer in choosing the suitable variant (of the root-process) for the context profile mirroring the real world environment, by using a decision model that points out the changes carried out on the beginning process.

A process model describes a set of methods, practices, skills, tools and the relationships among them to define a product or a service with a certain degree of quality.

The activity of tailoring a process model consists in specializing a process model to the context profile of interest. Formally speaking, given a process model PM_x , it is necessary to define the context profile in which the process PM_x can be executed. By establishing the proper values FI_{ih} of each diversity factor $[DF_i]_j$, the correspondent context profiles may be obtained: $CP_1, CP_2, CP_3, \dots, CP_n$. As a consequence the PM_x is modified in order to reflect the differences among the diverse context profiles. For this reason, a reviewed version of the original process model can be associated to each context profile, determining the specialized processes $SP_{x1}, SP_{x2}, SP_{x3}, \dots, SP_{xn}$. In Figure 1 this concept has been depicted.

3.2 Process Patterns

Starting from the problem to solve, a process pattern should show an initial solution, represented by PM_x and the path of actions (mainly process fragments) to be applied in order to obtain the correct SP_{xj} according to a specific CP_j . In this way, the ProMisE Process Pattern expresses the pair problem-solutions for defining a family of solutions (the variants SP_{xj} of the PM_x) related to a problem. It should present the following elements:

Name: identifier of the pattern.

Problem: description of the problem supported by the pattern.

Process Model (PM_x): the root-process model for which the specialized ones are created.

Decision Model: the decision model defines the path of actions for specializing the PM_x according to a predefined CP_j .

Solutions: the specialized processes SP_{xj} .

Relationships: they consist of other process patterns (PP_i) that specialize the current one. These patterns refer to sub processes that detail an SP_{xj} phase.

Experiences: experiences reported in using the pattern.

Each specialized process (SP_{xj}) contained in a pattern can also have relationships with many other patterns. This can occur when a SP_{xj} 's phase is detailed by a sub-process associated to a pattern.

Every time a pattern refers to other patterns, the last one details the first one. If a given pattern doesn't refer to other ones, then it is either at its highest level of detail or it lacks a detail pattern.

The Decision Model can be represented by using a variant of a decision table that emphasizes which actions must be accomplished on the process model for specializing it to a context profile:

1. the conditions represent the diversity factors (DF), i.e. the features of the context profile in which the process will be carried out;
2. the actions represent the "specialization action" to be accomplished in order to obtain a specialized process SP_{xj} of the PM_x ;
3. the rules (represented by the columns) merge a combination of factor instances FI_{ih} (expressing the context profile CP_j) with the specialization action to perform, in order to obtain the appropriate SP_{xj} for the context profile CP_j ;

Continuous reuse of a pattern will most likely provide the organization with three fundamental advantages.

First of all, like all other patterns, it is easily traceable within the repository because it is characterized according to the problem it refers to, and at the same time it probably corresponds to the problem the process engineer is trying to solve.

Second, pattern reuse may highlight variants that haven't been forecasted in the context profile that the decision model is based on; in this case the model itself is extended with the unexpected context model. Therefore, the pattern with reuse extends the number of variants it refers to and consequently becomes more complete.

Third, it may point out that a variant can be formalized in a more appropriate way within the context it refers to. This increases the pattern's efficacy and extend the process knowledge.

According to what has been previously mentioned, a pattern is a collector of knowledge generated by various sources and transferable independently from who generated it in first person. In other words it is an experience package. For this reason, ProMisE framework can be considered itself as an experience base [13].

3.3 Functions

We have defined the main functional requirements that the technological support of the framework should own. In this section an overview is provided. The functions related to the management and use of the pattern in the experience base are listed below.

Pattern Creation. This function must be used when a new problem arises. All the components of the pattern must be defined. Particular attention must be paid in defining the relationships that may exist between the pattern just created and the other ones already stored inside the experience base. When necessary, the process engineer must update or create the decision model from scratch and describe how the new pattern is linked to the context profiles involved.

Update Pattern. This function modifies a pattern already present in the repository. A modification can consist in adding a context profile; adding or deleting a factor instance in a diversity factor included in the pattern; modifying the specialization actions or the rules of the decision model; adding new relationships etc.

Select Pattern. It selects the pattern corresponding to the problem that the user needs to solve.

Generate Process Model. After having selected a pattern we have identified a process model (PM) and its variants; from here on referred to as root pattern. By assigning values to the diversity factors, the changes that must be carried out on PM are identified. This allows to produce a more specific process model (SP_{x_j}) for the operative context CP_j . Also, with refer to the combination of values of the diversity factors, the decision model may identify one or more patterns (PP_i) the process is related to. Each of these patterns PP_i corresponds to a family of variants of a process detailing an SP_{x_j} component (for example, an SP_{x_j} phase). Thus throughout the relationships, the root pattern is specialized in further levels of detail leading to the specification of another process $SP_{x_{j1}}$ detailing SP_{x_j} . This mechanism goes on throughout the relationships chain, pattern

after pattern, until having explored all of them. The result is a specialized process within the operative context having the highest possible level of detail according to the knowledge stored in the experience base.

4 Case Study

The framework presented in the previous sections has been applied, through a case study, to formalize the experience matured by industrial projects with the aim of clarifying how the framework can be used in practice. In particular, acquired knowledge in previous years within the Software Engineering Research Laboratory (SERLAB) projects is related to extraordinary maintenance of software systems. For further details see [14][15][16].

Legacy system rejuvenation is straightforward according to various factors: goals, budget, resources, economical value and quality of the legacy, and so on. When rejuvenating an aging system, one or more types of renewal processes are used. For instance: *reverse engineering* analyses a subject system to identify its components with their inter-relationships, and to create a representation of the system in another form or at a higher level of abstraction; *restructuring* improves the structure of a program automatically, without taking the program purpose into account; *restoration* improves the structure of programs, and of data according to their meaning in the programs; *reengineering* examines and alters a subject system to reconstitute it in a new form with improved quality. This may include modifications with respect to new requirements not met by the original system; *rehosting* refers to migration of the system to a different architecture; *migration* involves changing the software environment the legacy system runs on Chifosky (1990).

Before deciding the most appropriate combination of renewal processes to use, some preliminary activities must be carried out:

- **Portfolio analysis:** consists of the analysis of system's capabilities to be taken into account for the maintenance process.
- **Quality Assessment:** this activity identifies the quality level of the existing system and that of the modified one.
- **Economy Evaluation:** this assessment helps understand the cost and return of modifications.
- **Risks Assessment:** this assessment is headed to identify and mitigate risks.

These activities are necessary for deciding the renewal process to adopt according to the improvement goals of the process and of the constraints of the project.

The project goals considered within the case study are described as follows:

Diminishing the cost of application administration. There is a vast variety of cost sources in maintenance process; it is not possible to consider all of them, because some are specific to the Organization; others are hidden or difficult to identify, such as: transfer of knowledge, distance between the supplier of management services of the software system and the organization using the system.

Part of the cost taken into account are: Cost price of the software System; Cost of maintenance and operation of the system:

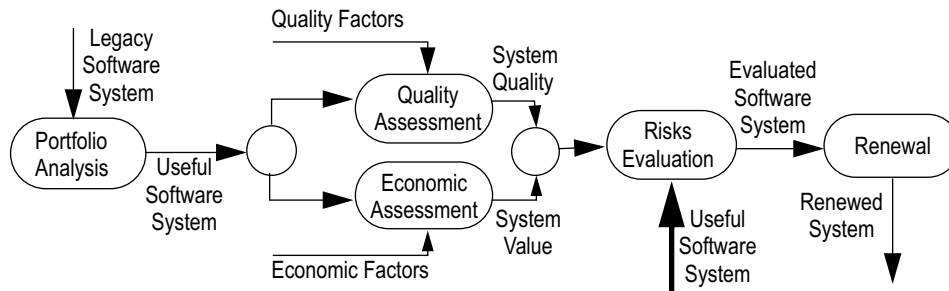


Figure 2: Root Process.

this includes also the costs for human resources and the relative logistics involved in the activity; Cost of assistance, basically refers to the cost of the experts; Cost of diffusion of knowledge of the package to maintainers, managers, assistants and users; Cost of the hardware and software platforms necessary for using, managing and maintaining the software system; Cost of coping with bad or incomplete functioning of the software system.

Relying on experiments developed in industrial settings it results that each cost of those listed before can be reduced by activating specific maintenance activities (re-engineering, rehosting, reverse engineering, restructuring).

Improve engineering features of application. When analysing a system it is possible to consider different detail levels. The two main ones are the code's structure and the engineering features. The latter refers to the general organization of the system, including the division in modules, the relationships between them, the data access and updating. The former regards more directly the intrinsic quality's aspects of the code. This goal often requires the legacy to be reengineered. Note that reengineering may also include some types of reverse engineering followed by restoration or restructuring in order to make the system more evolvable in future.

Change the application deployment. A maintenance process can require new capabilities of the development environment, in order to be executed properly and successfully. Such improvements can refer to, e.g., configuration management, development or maintenance environment, hardware settings, print management, and others. This often requires rehosting.

Change the middleware in use. Sometimes, in order to rejuvenate the legacy system (for example to be able to adapt to new technologies, peripherals, services, ...), the modification of the middleware of software system is required and thus migration is needed.

Improve code's structure. This activity aims at improving the quality factors of the code itself, such as coupling, cohesion, pollution, lexicon, and modularization. This goal is often reached by using restructuring.

Improve readability and understandability. In order to achieve a high quality maintenance task, programmers must be able to understand the code directly from the listing. Consider that the programmer works basically on it, also if he could use

further project documentation. In order to reach this objective a restoration process is used.

Update system's documentation. When modifications are brought on the code, the system documentation must be properly modified, in order to keep it aligned with the code. We mean all the documentation: requirements specification, analysis and design documentation, user manuals, test beds. In this case reverse engineering is needed.

Finally, when the legacy has a poor documentation, a low quality level, a low economic value and the previous improvements are all required, it is probably more convenient to *Write the system from scratch*. When Rehosting, Migration, Restructuring, Restore, Reengineering or Write from Scratch are executed, the Equivalence test is required in order to assure the equivalence between the legacy system capabilities and renewed system capabilities.

In Figure 4 the process pattern of extraordinary maintenance, is illustrated. In the following a brief description is given:

- **Name:** Renewal of legacy system
- **Process Model (PM_x):** it describes the core activities of a renewal process and the artifacts exchanged between process activities. There is also an activity named "renewal" that is further refined by using decision model. The root process is depicted in Figure 2.
- **Problem:** evaluate degradation and rejuvenate an aging system.
- **Decision Model:** the decision model defines the path of actions for specializing the PM_x according to a predefined CP_j. The decision model is structured as follows: in the gray part there are 7 diversity factors DF_i that represent the possible goals of the Extraordinary Maintenance (*Diminishing the cost of application administration; Improve engineering features of application; Change the application deployment; Change the middleware in use; Improve code's structure; Improve readability and understandability; Update system's documentation*). The factor instances FI_{ik} correspond to the possible values that a DF_i can assume. In this case they consist in "Y" or "N" that indicate the need to reach the correspondent maintenance goal. In the white part there are the 12 sub-processes that can be selected for properly specializing the "renewal" activity of the root process. The symbol "x" indicates that the extraordinary maintenance process must include the correspondent sub process

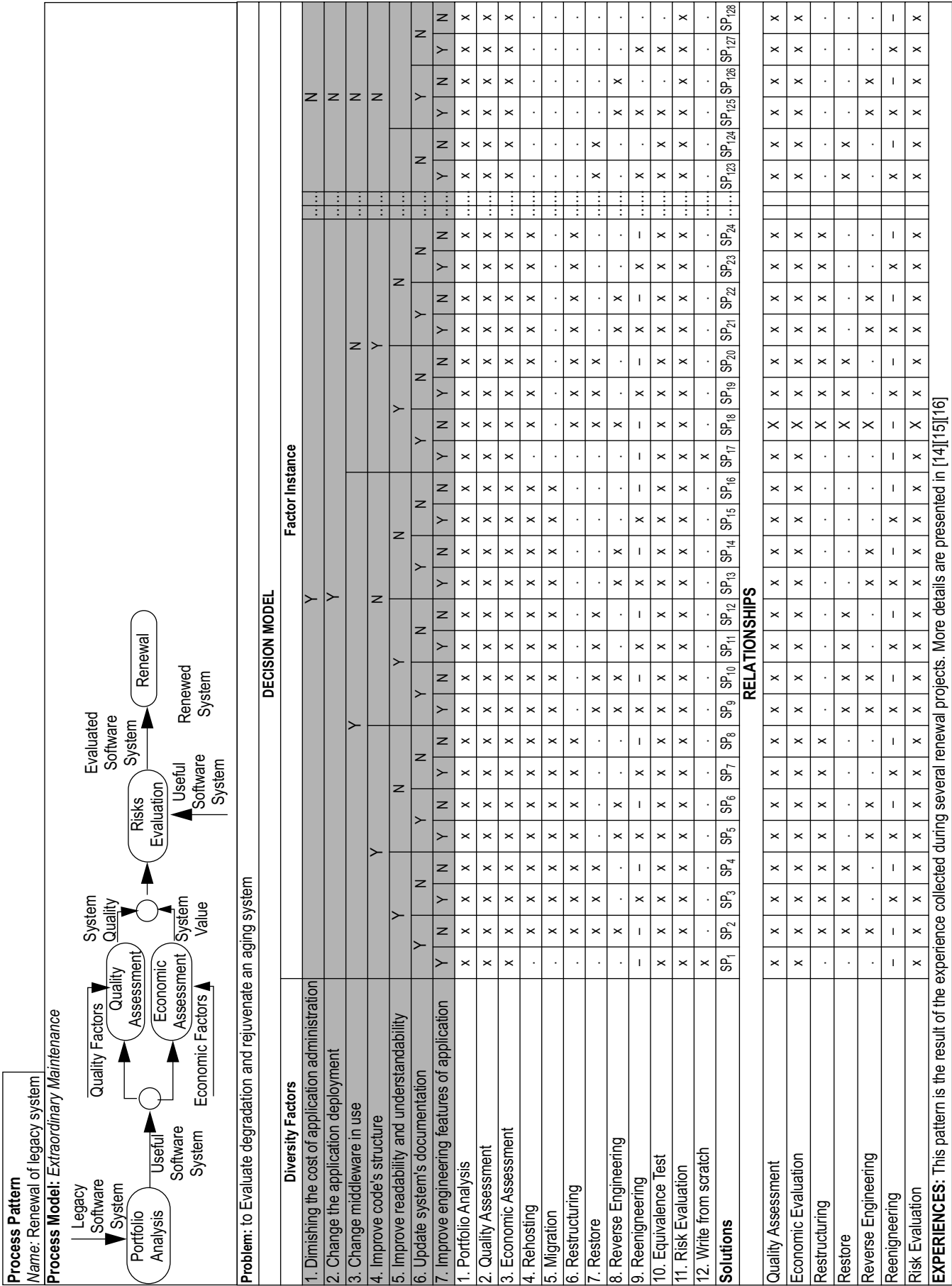


Figure 4: Extraordinary Maintenance Process Pattern. (b)

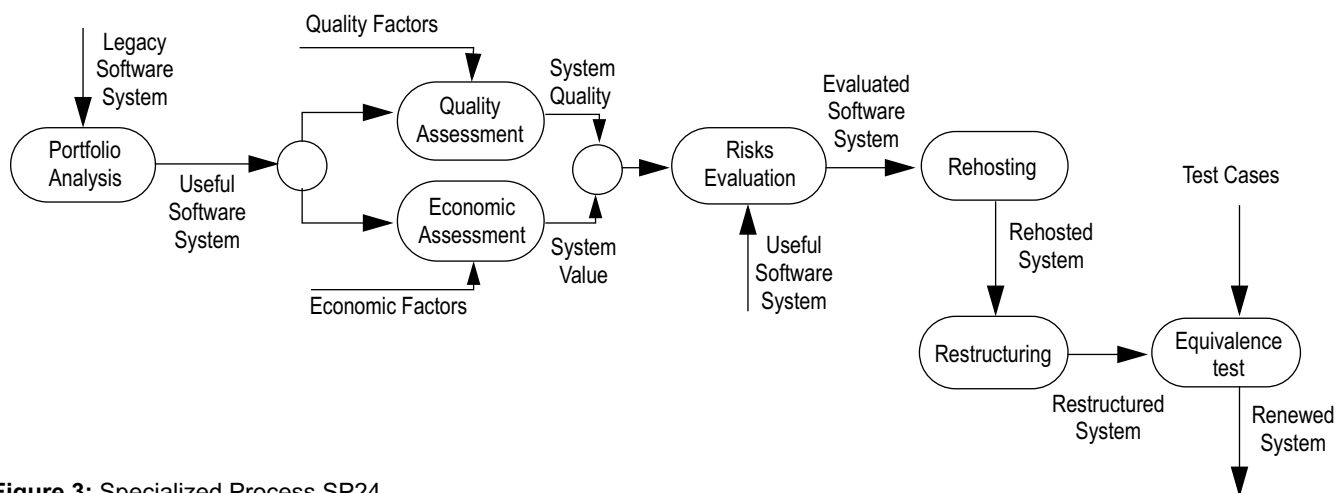


Figure 3: Specialized Process SP24.

reported on the same row. Thus, each column contains the specialization rules to obtain a specialized version SP_{xj} of the PM_x on the CP_j . In Figure 3, only a part of the decision model is presented. It includes a total of $2^7=128$ rows that correspond to the different CP_s considered.

- **Solutions:** The decision model individuates 128 different SP_{xj} of the root process PM_x . As an example, let's consider the case of the column 24: the context profile is $CP_{24}=\{Y,Y,N,Y,N,N,N\}$, i.e. the goals of the correspondent specialized process are:
 - *Diminishing the cost of application administration,*
 - *change the application deployment,*
 - *Improve code's structure*

The rule on the column 24 dictates that the SP_{24} must include the following activities:

- *Portfolio Analysis,*
- *Quality Assessment,*
- *Economic Evaluation,*
- *Rehosting,*
- *Restructuring,*
- *Equivalence Test,*
- *Risk Evaluation.*

The resulting specialized process SP_{24} is showed in Figure 3. The rationale for specialization was elicited from [14][15][16].

- **Relationships:** this section specifies other process patterns (PP_i) that specialize the current one. These patterns refer to sub processes that detail an SP_{xj} phase. In the case of the SP_{24} of the pattern in Figure 4, in order to obtain a fully specialized process the following PP need to be further considered:
 - *Quality Assessment,*
 - *Economic Evaluation,*
 - *Restructuring,*
 - *Risk Evaluation.*

- **Experiences:** indicates the source of experiences used to define the pattern or projects executed by using the pattern considered. For what concerns pattern in Figure 4, the

source of experience is represented by the references [14][15][16] included in the bibliography.

5 Discussion and Conclusion

The innovative part of ProMisE consists in keeping an experience base updated by recording the continuous changes in the real world. The major advantages in using ProMisE can be synthesized in the next two points:

1. *To improve the comprehension of the characterization.* Over the time, the Organization implementing the process can better detail the set of operative contexts, by adding (or deleting) some DF or by adding or deleting some FI. It is easy to imagine which sort of chain effects this has on the experience base: all the patterns and the decision models involved must be appropriately changed.
2. *To improve the comprehension of the relationship between the characterization and the specialization.* Some evolutions in the actual operative context can lead to choosing another specialized process segment for the context profile CP rather than the current one recorded in the experience base.

Processes are adopted in frequently changing environments; this condition leads to expensive and complicated process tailoring, deploying money and consuming time to ensure stability and avoiding lacks of capability.

Given a problem and a first solution for it, thanks to the concept of patterns, ProMisE allows to specialize the initial solution according to the context in which it will be used. Furthermore, it points out all the changes made throughout the specialization process. In this way it addresses for reusing experience.

ProMisE aims to extend the well known concept of pattern and more precisely it wishes:

1. To create a family of solutions for a family of problems just starting from specific pairs problem-solution. This allows the process engineer to apply the work and the knowledge developed in specific experiences to many other different ones, but placed in the same family of process-solution pair.

2. To continuously enrich the experience base by properly modifying the decision models and the process patterns as real world evolutions occur and consequently to make the organization more capable to face process diversity problems.

We are realizing the system in collaboration with a company operating in the field of Enterprise Resource management.

Some problems have been highlighted and are discussed in the following.

First of all the management of experience is a central concern. The knowledge base of the system grows as the captured experience increases. The experience needs to be elicited in the appropriate way by the field. This is not a trivial issue, if considered scalability and competence of the process engineer. The competence of the process engineer regards basically the experience to be stored (too much can create pollution in the base, and if it is not enough it can make usefulness the effort of maintaining the system).

Scalability is another problem to be faced; as matter of fact, decision models can grow very fast. An excessive growth can affect usability of the system (navigation between the decision tables, comprehension of the content of decision table) and the maintainability of the system (validate consistence of the data and identify the impact of the changes).

Finally, packages of integration should be properly defined in order to create an appropriate process by the combination of the single components. Parameters to be used in the packages must be identified properly, values must be defined and the couples parameter-value have to be updated when needed. From the realization of the system we aim at elaborating solutions for these problems and validating them by their application.

References

- [1] M. Lindvall and I. Rus, "Process Diversity in software Development", IEEE Software Vol.17 N° 4, IEEE Computer Society, Los Alamitos, July/August 2000, pp. 14–18.
- [2] K. M.Dymond. A guide to the CMM-understanding the Capability Maturity Model for Software, Press Inc.
- [3] M. T. Baldassarre, D. Caivano, C. A. Visaggio, and G. Visaggio, "promise: a framework for Process Models customization to the operative context", proc. of IEEE International Symposium on Empirical Software Engineering, 2002, IEEE Computer Society, pp. 103–111.
- [4] D. C.Schmidt, M. Fayad, and R.E. Johnson, "Software pattern", Communications of the ACM Vol.39, No.10, ACM, New York, October 1996, pp. 37–39.
- [5] B. Appleton, "Patterns and software: Essential concepts and terminology", 2000. <<http://www.enteract.com/~bradapp/docs/patterns-intro.html>>.
- [6] T. Winn, P. Calder, "Is this a pattern?", IEEE Software Vol.19, N°1, IEEE Computer Society, Los Alamitos, January/February 2002, pp. 59–66.
- [7] F. Houdek and H. Kempter, "Quality Patterns – An approach to packaging software engineering experience", Proceedings of the 1997 Symposium on Software Reusability, ACM Software Engineering Notes Vol. 22, Num. 3, ACM, New York, Mai 1997, pp. 81–88.
- [8] P. W. Fach, "Design Reuse through Frameworks and patterns", IEEE Software Vol.18, N° 5, IEEE Computer Society, Los Alamitos, September/October 2001, pp. 71–76.
- [9] L. Rising, "Patterns in postmortems", Twenty-Third Annual International Computer Software and Applications Conference, IEEE Computer Society, Phoenix, Arizona, October 1999, pp. 314–315.
- [10] M. Fredj and O. Roudies, "A pattern based approach for requirements engineering", 10th International Workshop on database & Expert Systems Applications, IEEE Computer Society, Florence, Italy, September 1999, pp. 310–314.
- [11] S. M. Sutton, Jr., "The role of process in a Software Start-up" IEEE Software Vol.17 N° 4, IEEE Computer Society, Los Alamitos, July/August 2000, pp. 33–39.
- [12] M. Morisio, C. Tully, and M. Ezran, "Diversity in Reuse Processes", IEEE Software Vol.17 N° 4, IEEE Computer Society, Los Alamitos July/August 2000, pp. 56–63.
- [13] V. R. Basili, G. Caldiera, and D. Rombach, "The Experience Factory", Encyclopedia of Software Engineering – 2 Volume Set, 1994, pp. 469–476.
- [14] G. Visaggio, "Value-based decision model for renewal processes in software maintenance", Annals of Software Engineering, 9 (2000), Kluwer Academic Publishers, pp. 215–233.
- [15] G. Visaggio, "Ageing of a data –intensive Legacy System: symptoms and remedies", Journal of Software Maintenance and Evolution: Research and Practice 13 (2001), John Wiley, pp. 281–308.
- [16] A. Bianchi, D. Caivano, G. Visaggio. "Quality Models Reuse: Experimentation on Field", Proceedings of the International Computer Software and Applications Conference (COMPSAC) – IEEE Computer Society, Oxford, England, August 2002.
- [17] I. Rus, C. Seaman, M. Lindvall, "Process Diversity", Journal of Software Maintenance and Evolution: Research and Practice, 15(2003) John Wiley, pp. 1–8.
- [18] J. Munch, "Transformation-based Creation of Custom-tailored Software Process Models", Proceedings of the 5th International Workshop on Software Process Simulation and Modelling 2004, ProSim 2004 May 2004