Process Reuse Architecture

Soeli T. Fiorini, Julio Cesar Sampaio do Prado Leite,
and Carlos José Pereira de Lucena

Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio
Rua Marquês de São Vicente 255, 22451-041 Gávea-RJ
Rio de Janeiro, Brasil
{soeli,julio,lucena}@inf.puc-rio.br

Abstract. This paper proposes a systematic way to organize and describe processes, in order to reuse them. To achieve that, a process reuse architecture has been developed. This architecture is based on processes and their types (standard, pattern, usual and solution), on process frameworks, based on the theory of application framework and on different kinds of process modeling languages, which are specified in XML, to describe each type of process. In order to facilitate the reuse and retrieval of information, we use facets, reuse guidelines, as well a process patterns taxonomy. Some processes of requirements engineering have been analyzed so that it was possible to create a process framework and a web tool has been developed to enable a case study to validate the proposed architecture.

1. Introduction

The growing interest in Process Engineering is an evolution of studies focused on products which verified that the quality of the developed software has a strong relation with the process which is used to elaborate it. In 1984, at the First International Workshop on Software Processes - ISPW a group of researchers learned about the new area of process technology, which emerged at the time. Afterwards, more than twenty workshops and conferences have been held (ICSP, IPTW, EWSPT, FEAST e PROFES)

In the last few years research on software reuse has focused on process related aspects, such as: software process programming [1] [2], software engineering environments based on processes [3], user interface guided by defined processes [4], software processes improvement [5] [6] [7] [8] and definition of process patterns [9] [10] [11] [12] [13] [14]. Although some problems, directly or indirectly related, have already been solved, others remain waiting for better proposals.

Nowadays, while attempting to improve processes, many companies try to reach levels of process maturity [5], based on their improvement and definition. However, those improvements are expensive, and they rely on the involvement of managers and teams, process data, of people who know process modeling, training, and cultural change. Several factors, imposing difficulties, make companies spend long periods of time to define some processes [15], and some of them give up in the middle of the maturity process. A frequently mentioned process to accelerate this within a company is to replicate one organizational process in other projects. At this point, the process

K.R. Dittrich, A. Geppert, M.C. Norrie (Eds.): CAiSE 2001, LNCS 2068, pp. 284–298, 2001.

descriptions are very important because they allow the knowledge to be reused. However, the creation of a process, which is used in several other projects, is a hard task [19].
Models such as CMM [5] or the paradigm of the software programming process [1] aim at following one defined process and at having one standard; nevertheless, they rarely have the concept of reuse of process and artifacts, as a central objective [16]. Many organizations have to begin the definition of their processes based on some kind of existent tacit knowledge or solely on bibliographies which demand a lot of time to be understood, because they often contain inconsistencies.
Another possibility is to begin the definition based on models or norms, such as ISO and CMM; these however, are either too generic or need interpretation and are not always organized in the form of a process. Efforts made to measure the definition of processes to projects, report that it is necessary, for a key area (CMM), between 800 and 1000 hours/people or more, depending on the amplitude and depth of the process [17].

In order for the processes to be reused, the companies need to express common elements and variables within one process. Frameworks [18], [19] provide a mechanism to obtain such reutilization [17] and are very appropriate to domains where various similar applications are built several times, from scratch. Researches on patterns [20] [21] also have shown that they are effective tools for reutilization.
The use of application framework and patterns has taken place in many areas, such as design, interface, code, organizational and analysis; however, it seldom happens in processes. The utilization in processes is still isolated [22], [23] and [9], with a not very clear definition and needs to define what processes and patterns frameworks really are, and they do not have specific techniques for its development. Therefore, by integrating many concepts, we present a process reuse architecture, based on kinds of processes (standard, pattern, usual and solution), on process frameworks, based on the application frameworks theory, and on different types of process modeling languages, which are specified in XML, to describe each type of process. A tool, which implements this architecture and its elements, is presently in its late stage of development.

Section 2 presents the process reuse architecture and its elements. Section 3 presents how to use the architecture (processes and frameworks) to generate a solution process – a process instance. Section 4 presents related works and we conclude in section 5.

2. Process Reuse Architecture

Architecture in its strict meaning is the art of building. Thus, the process reuse architecture proposed here is an organization of processes and related elements. The objective of the architecture is to organize processes to enable their reutilization. By using the organization and structuring of information together, we try to facilitate the access and reutilization of the processes. Because it is reusing processes, it has all the benefits known in the technology of reuse, making it possible to use acquired knowledge and experience, which have been already established by other organizations or researchers [28]. A process is defined here as a set of interrelated activities that receive inputs and produce outputs.

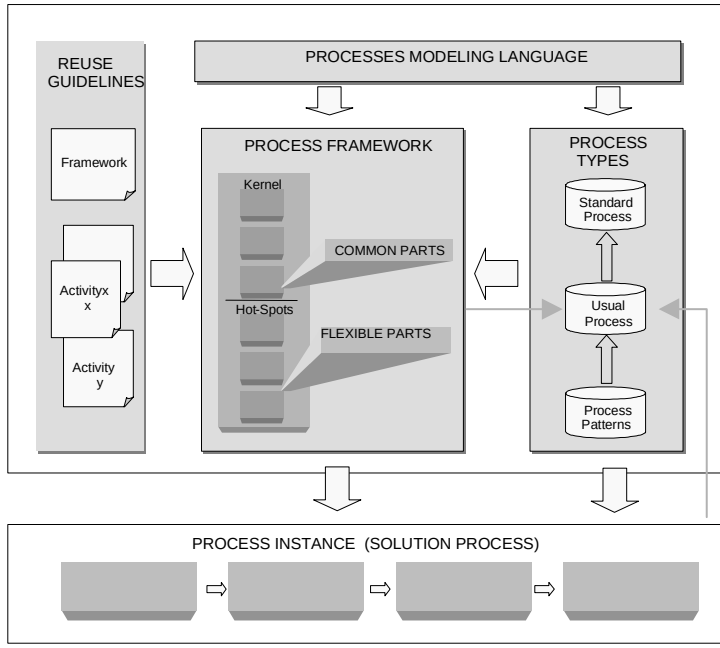

Fig. 1. Process reuse architecture.

Figure 1 illustrates the proposed architecture. The architecture elements are the process modeling language, types of processes (standard, pattern, usual e solution), process framework and reuse guidelines.

Process Modeling Language

A process modeling language is a formal notation used to express process models. A process model is the description of a process, expressed in a process modeling language [24]. To express the process model we have chosen XML [25]. XML specifies a marking meta-language that allows the structured representation of several types of information. This information can be sent/received and processed on the Web in an even way. Because the processes naturally have a hierarchical structure, we adopted the XML, taking the advantage of its benefits to work with documents structuring. The XML's DTD – Document Type Definition – contains or points to the declarative marks, which define a grammar or set of rules to a certain class of documents. This way, in a DTD we define languages to model the processes. It is important to emphasize that each one of the process types has a representation and therefore they can contain different attributes in their description, although many attributes appear more than once. The languages for the process patterns are both an evolution of design patterns templates [20], [21] e [26] and research on software reuse. For the other kinds of processes, the base was previous results of process modeling [27], [28] and [29], including the framework idea. All kinds of processes

have a great deal in common as far as the modeling is concerned, and that allows the reuse to be carried out more amply. For example, all of them have activities

Process Types

Processes will form the base of the architecture. The types have the purpose of distinguishing processes, characterizing different sources of information used in the process definition. All the processes are stored in a database. When the reuse takes place, a process instance is stored in the usual process database and an XML file can be generated, enabling the process engineer to store one copy in his/her directory. The types of process defined are:

- **Standard Process:** it is a standard (for example, CMM or ISO) form of process. It is a base for process definition, according to specifics process improvement and quality assurance standards. It has a normative purpose.
- **Process Pattern:** it is a process that deals with problems related to processes. According to the definition of patterns, it can not be new or hypothetical [20] [21].
- **Usual Process:** is any existing process that is neither standard nor a pattern. It is not standard because it does not have the normative purpose and it is not a pattern because it has not been tested (applied) a considerable number of times to solve one recurrent problem.
- **Solution Process**: it is an instance of either the framework or of any other process (pattern, usual or standard), or of the combination of those.

Process Framework

A process framework models the behavior of the processes in a given domain [41]. A process framework is defined as a reusable process. Its kernel models the common activities among the processes of the domain. Hot -spots model the specific and flexible parts of the processes of the domain and they are reified during the instantiation of the framework. The hotspots are activities or other elements of the process (techniques, tools...), that define characteristics or specific paths of a solution process (process instance). The hotspots are instantiated by the process engineer, redefining the description of activities or elements, both on the macro and detailed levels. The framework itself will have to be represented as a process, identifying the parts that are hot spots and common activities. The instantiation of one processes framework generates a solution process. In the next Section we detail the use of the framework

Reuse Guidelines

The objective of the reuse guidelines is help reuse and adapt the framework and it also shows both flexibilization and common points. The reuse guidelines are available for the framework, given an overview of whole process, and for the macro activities. On the upper side of the guide is the process or the activity name. The guide design (Figure 2), follows a structure based on hot-spot cards [18], features model [30] and components reuse concepts – 3C – [31], whose details follow:

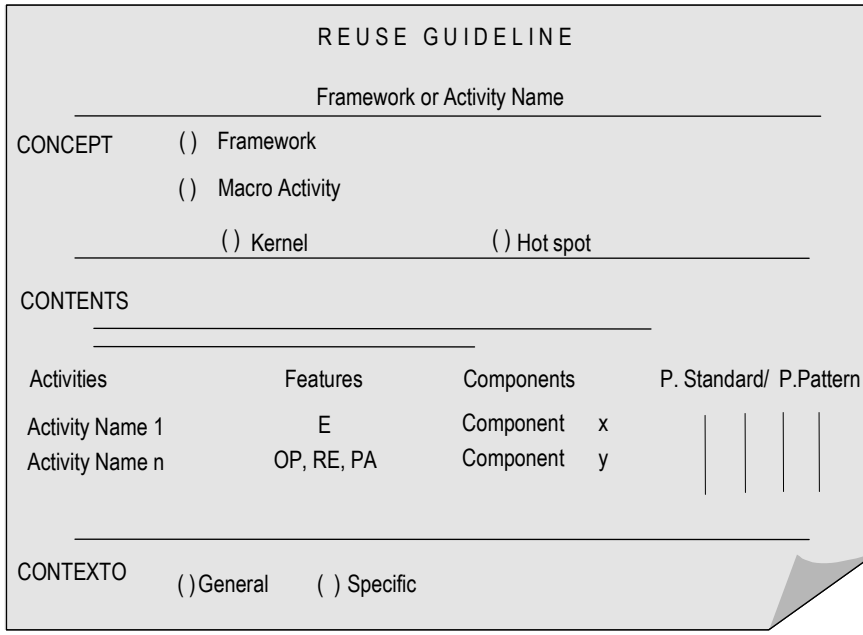

Fig. 2. Reuse guidelines

The 3Cs Model

3Cs Model of reuse design provides a structure, which has been effective in the design of reusable assets [22]. The model indicates three aspects of a reusable component – its concept, its contents and its context. The concept indicates the abstract semantics of the component; the contents specify its implementation and the context specifies the environment, which is required to use the component.

The "Concept" section of the reuse guidelines provides information as to whether the guide is one of framework as a whole (giving a general view of the process), or a specific activity, on macro level. It also provides information to identify, in the case of activities, whether they are hot spots or common point. (kernel).

The "Contents" section identifies activities/ components. Activities are elements of the framework and components are other activities (from database) that can complement process description or fill the framework hot spots - when reusing the framework. For a framework, the macro-activities are the components themselves. For the macro-activities, the detailed activities are the components. It also provides information to identify the features of each activity (see next item).

For each activity it should be possible to access its basic data and the components related. For example, "Elicit Requirements" is related to "Collect facts". In the same section, available process patterns and process standard items, such as practice in the CMM, for the macro activities are listed.

The "Context" section defines whether the framework is generic or specific. As far as the activities are concerned, the section describes the applicable techniques and when they should be applied.

Features Model

The features are present in their original proposal, the FODA (Feature Oriented Domain Analysis) [30] project, aiming at "through the model, capturing the final user's view of the applications requirements to a determined domain". In our proposal we use the characteristics as a semantic model, where they are the process activities attributes. They are useful to organize the framework, allowing it to identify the flexibilization and common points. The domain engineer captures, from the analyzed processes, certain relevant features, so that the process engineer can have a reference about the process activities in that domain when he reuses processes.

Some examples of features are:

[CO]mmon: the characteristic indicates that the activity is common, in the set of processes, which are analyzed in the domain;

[OP]tional: the characteristic indicates that the activities are not mandatory in the domain;

[S]pecific: when the activity is specific of few processes in the domain. They represent particularities of certain processes;

[IM]plicit: the characteristic indicates that the activity in the analyzed processes in the domain is implicitly present in the process descriptions, although it does not appear clearly as an activity.

Hot-Spots Card

According to Pree [18] a hot-spot card describes hot spots. Comparing what the hot-spot card specifies and what the reuse guidelines provides, we have, respectively:

- The hot spot name – in the heading of the guide it is possible to visualize the name of the process or activity. However, it is only in the "Concept" section of the guide that we find information on the flexibilization points (common point or hot spot).
- A concise term describing the functionality – In the "Contents" section of the guide the component names and the components lead to the objective/description through links.
- Degree of flexibilization – the "Context" section of the guide defines the flexibilization.

Operations

The process engineer can carry out the basic operations: inclusion, exclusion, searches and update associated with the process database. During the research he can choose a process, which might be reused as it is, or not. If the process does not meet the engineer's expectations, he or she can adapt the chosen process or create a new one, by reusing activities from several processes available in the database. The framework tailoring is not manual. We instantiated processes originally from the databases, from where they are retrieved by the use of facets and other resources. Our proposal is not to enact processes, but to define them.

Process Classification and Retrieval

There ought to be a mechanism to classify the content of the databases and to assist the user in the queries of processes, to find the process that better matches his or her needs. We define several process elements that are used in the process classification and retrieval. For example, we can retrieve using the usual process classification (fundamental, organizational or support) or using the taxonomy created for process pattern based on problems addressed (requirements elicitation, requirements analysis, ...). In this section we will describe two classification methods: facets [32] for all process types and Pattern Society [28] for process pattern.

Facets

Prieto-Díaz [32] proposes a scheme of classification through facets to catalog software components where it is believed that the components can be described according to their functions, how they perform such functions and details of implementation. Facets are considered as perspectives, points of view or dimensions of a particular domain. The elements or classes that comprise a facet are called Terms. It has been demonstrated that facets help in organizing and retrieving in component libraries.

The facets are used as describers of activities, focusing on the action that they carry out and on objects, which are manipulated by the activity. For example, the activity "elicit requirements" has the action "to elicit" and as object "Requirements". Both the classification and the retrieval specify the actions and/or objects when carried out. All the actions may have a list of synonyms so that they improve the retrieval.

Process Patterns Society

One of the problems that we face when we search for a pattern is to locate the appropriate pattern and to understand its solution. Many patterns are proposed individually, and others come organized in some languages for a specific purpose. Therefore, it is essential that there be a form to classify patterns in order to retrieve and use them. The term "Process Patterns Society" expresses taxonomy to organize the process patterns. In the textual description of the pattern solution, many times several links appear, for example, reference to other patterns. Our approach gives more structure for the pattern solution. The society is a way to organize them and make it look like a "patterns network map", which facilitates the search and understanding of the presented solution. It is something similar to a Catalog used in Design Patterns or Patterns System [33]. In our work, with the taxonomy of the proposed society, it is possible to organize the process patterns as: individual, family or community (see figure 3).

- Individual Process Pattern: is a process pattern, which presents a solution to a problem, without making reference to other patterns.
- Family Process Pattern: is a process pattern whose solution comprises one or more patterns.
- Community Process Pattern: is a set of process patterns, individuals and families of a certain domain.

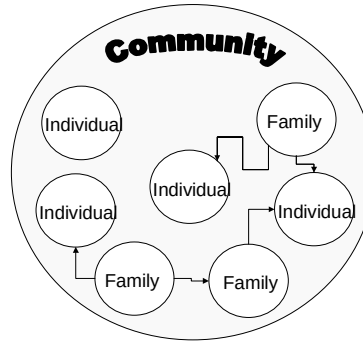

Fig. 3. Process pattern society

RAPPeL – A Requirements Analysis Process Pattern Language for Object Oriented Development [23] is a pattern language with several patterns. One of them is the Requirements Validation pattern and the other one is the Prototypes pattern. Using RAPPeL´s Requirements Validation pattern as an example, we will explain our approach. This pattern has the following problem and solution description:

Problem: "*How to verify that the specified behavioral requirements are correct and complete?*"

Solution: "*Have all interested parties thoroughly read the requirements specification. Conduct review meetings on sections of the requirements specification. Have a secretary take note of every issue raised during the reviews in the Issues List. Follow up on all issues raised.*

Build prototypes and review them with users. Again, record every issue raised in the Issues List and have follow up on each issue. Continue verification of requirements during system development through each iteration.

If needed, establish an arbitration group to reconcile disagreements on requirements.

Distribute prototypes to customers and conduct surveys and usability studies."

Using our approach we are looking for activities (components) at the solution and emphasize them. As a result the Requirements Validation pattern, described below, has a solution composed by two components (we have describe only the first component in our structured language):

1) **Component Name:** Conduct (action) reviews meeting (object)
   Synonym: Realize, execute
   Pre-condition: Have a documented requirements specification
   Input: Requirements specification
   Recommendation: Conduct review meetings on sections of the requirements specification. Have a secretary take note of every issue raised during the reviews in the Issues List. Follow up on all issues raised. Use cases are excellent in structuring the specification of the behavioral requirements for study and validation. Customers can 'role play' the various use cases using the domain objects to get a better feel on whether the system is doing what is expected.

Restriction: it is a human process with no automated support
Post-Condition: requirements specification reviewed.
Output: Issues List

2) **Component Name:** Build (action) prototypes (object) – individual pattern

The Requirements Validation pattern solution makes a reference to the Prototypes pattern (build prototypes), so it is a process pattern family. Component names are organized using *action* and *object* facets, which are used to retrieve information. Sometimes we complement the recommendation with other information described in patterns sections other than solution section (”Use cases are ....” is not present into the original solution).

3. Generating the Solution Process

In order to generate the solution process from the proposed architecture, we can reuse processes by using the framework, or we can surf throughout the database of the process (standard, usual and patterns), looking for one which is adequate to our needs. The solution process, which is generated, is saved as a usual process and can be visualized and used in the XML version (Figure 4).

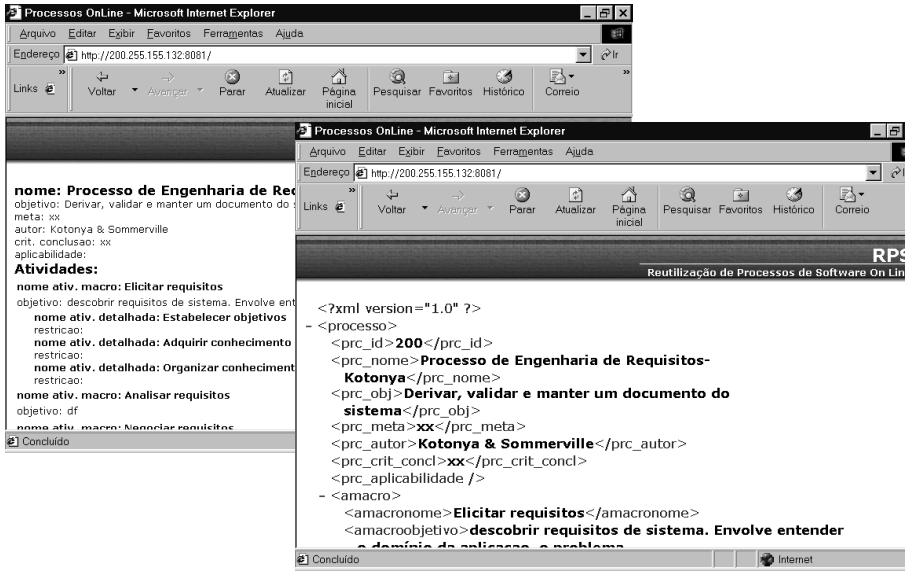

Fig. 4 XML view

In order to reuse from the database, we use the search resource, throughout which we can choose any element of the process to carry the search, such as tools, techniques, and templates. Also facets can be used and in order to locate patterns we can base our search on process patterns society. A web tool, which implements the

proposed architecture, allows searching and the results can be compared. For example, we can compare a process made by an author with another one, activity by activity. The processes can be partially or totally reused (some activities). The solution process for example, may contain activities from any kind of process: standard, usual and patterns. This is possible because all process types have the same kernel – the "activities detailed" – that are called "items" in process standards and "components" in process patterns. The kernel is the activity's name, description, inputs, outputs, pre-conditions and post-conditions.

To generate the solution process, based on framework, we must utilize the reuse guidelines. The following table illustrates the initial part of a guide, for the requirements engineering process – giving the general view of the process (framework). The guide partially shows two macro activities: optional [OP] – hotspot – and a common [CO] and essential [E] – kernel. What follows is a description of how it is carried out and reused, based on the guide.

REUSE GUIDELINES						
REQUIREMENT PROCESS ENGINEERING						
CONCEPT						
(x) Framework		( ) Macro Activity		( ) Detailed Activity		
		[ ] kernel [ ] Hotspot				
CONTENTS						
Objective: derive, validate and maintain a requirement document.						
Activities	Features	Components	Process Standard/ Pattern			
			CMM N2 ( )All	*CMM N3 ( ) All*	*CMMI Cont.*	*Pattern*
Analyze the Problem	*OP,AN, FO*	Analyze the Problem – Rational Define Scope – Alister FO - Context Analysis– Bracket (outside the requirement process)			*GP1.1*	
Elicit Requirements	*CO, E*	Elicit requirements – Kotonya/ Julio/ Loucopoulos/ Sommerville Collect facts – Alister Capture Requirements – Alcazar Identify Requirements – Brackett Understand Necessities of Stakeholders (it has Elicit requests of the Stakeholders)/ Analyze the Problem – Rational			*GP2.7 and SG1*	*Define initial requirement*

Reusing Activities

Whenever a process engineer selects a framework, its activities are showed together with its reuse guideline (table below). The instance that will store the new process, based on the framework, needs to be selected (from usual process combo box). Using

the features into the guide it is possible to know what activities are common (CO) and essential or optional (OP). The common macro activity description can be complemented with component description. The optional activities (hot spots) will be deleted or will be filled. We can fill the activities looking to the reuse guidelines for one component or process pattern that matches the process engineer needs. When a component or process pattern is reused to fill an optional activity its detailed activities are reused too. In the case of common macro activities it needs detailing. That means that it is necessary to inform which detailed activities are part of the macro activity. To do so, we must use the macro activity reuse guidelines. In this example, it would be the guide for the reuse of the "Elicit Requirements" activity. The "Context" section of the reuse guidelines (not shown in the example) provides the techniques that can be used and when they should be used.

Reusing standards

We try to use standards in our work, such as CMM, ISO/IEC 12207 or CMMI in a different way. Usually, a standard is chosen and processes are defined from it. On the framework, each activity can have one or more items from the associated standard; for example, "Elicit Requirement" has two associated CMMI standard items. In this proposal, we start from a process (framework) and we verify if the existing description complies with the standard.

Obviously, some standard items are not totally covered by the framework, which makes it necessary to act in the usual form. However, we provide support to carry it out in this form.

The standards usually have items that refer to activities themselves and others that refer to institutionalization matters, resources and organization. Those standard items that refer to activities and do not have references on the framework, are automatically created as activities in the solution process, when the option is "all" the standard. For those standard items, which have a reference activity on the framework, the process engineer should refer to two standard aids: interpretation and recommendation. The interpretation is a specialist's knowledge about the standard and it aims simply at facilitating the understanding, minimizing doubts about how to use the standard item. The recommendation is a tip on how the item can be applied in the activity, making it comply with the standard. The standard items that do not refer to activities are linked to the definition in other elements of the process, such as tools, method, technique and training, when necessary.

Reusing patterns

Patterns work as an aid for defining the process activities. Patters related to the activities are listed beside and can be consulted. For example, "Elicit Requirements" has an associated pattern, which is "Defining initial requirements". When we wish to use a pattern in the activity definition, we can chose to reuse the pattern solution as it is, or we can create a new one, based on the pattern solution.

4. Related Work

In the literature little has been found regarding the description of process patterns. Coplien [34] defines process and organization patterns as managerial practices,

Ambler [9] shows process patterns in a particular way, classifying process patterns into three levels, but recognizes that his descriptions does not follow any established form. Landes at. all [35] use quality patterns as a "primitive representation to describe experiences in a structured manner", which is our goal too. The experience is classified, in more details than in our approach, using some facets and some keywords. But they do not have a pattern organization (pattern society) similar to ours and a structured way to describe solution inside the pattern. On the other hand, they have subcategories for patterns (theory patterns, practice patterns and lessons patterns) to model the evolution of experience over time. Vasconcelos [12] describes an environment and forms for process patterns, not clarifying, however, the difference between solution process and process pattern. Vasconcelos also uses the concept of activity pattern for what we call components, and he does not use a hypertext to improve the description of the patterns. In [36] we can see patterns "related to the system testing process", but the description form is traditional. In the article "A Pattern Language for Pattern Writing" [37] a lot of patterns related to pattern structure are shown, pattern naming and referencing, patterns for making patterns understandable and language structure patterns, which coincide with our main idea.

Related to our general idea about the architecture, some works are similar in some aspects, without process types and guidelines and with focus on tailoring process to project. Henninger [38] [39] has an environment for reuse process that tries to reduce the gap between the process that is defined and the one that is in use, by combining an organizational learning meta-process with a rule-based process engine. With a similar objective Rolland et. all [40] use a contextual model approach constructing process models dynamically. Hollenbach and Frakes [22] have a framework organized in several sections for process definition. Each section has some attributes similar to and different from ours. They have a tailoring methodology.

Concerning our Patterns Society schema for patterns classification, it is somehow similar to a Pattern Catalog (collection of related patterns) or a Pattern Systems (cohesive set of related patterns which work together to support the construction and evolution of whole architectures) [33].

5. Conclusion

The result of Rollenbach and Frakes's research [22] shows that at least a tenfold improvement in time and effort to create a project or business unit process description occurs when we instantiate a reusable process instead of building the process from scratch. Our approach involves to building processes from the process data base and frameworks. We use processes as a way to store and reuse knowledge in organizations.

In this article we define an architecture based on the concepts of Standard Process, Process Pattern, Process and Solution Process. It is important to emphasize that each one of them has a representation and therefore they can contain different attributes in their description, although many attributes appear more than once. Process patterns are both an evolution of design patterns templates and research on software reuse. The other approaches are based on current research on process modeling. We achieve more expressiveness through a powerful process data base that integrates this kind of processes, because we can establish relationships between, for example, problems

(described in process pattern) and usual process of the same domain. Also, we propose a taxonomy for the classification and retrieval in which patterns can be perceived individually, as a family and as a community and activities can be organized using facets. It is very important for process engineers and SEPG managers when defining process (in software process improvement programs) because they will have one central information source together with one schema of classification and retrieval to help them. This reduces process definition time.

In short, the contributions are:

- An architecture of processes reuse gathering different concepts (patterns, standards and frameworks), which enables the reuse in a systematic and practical way.
- A more structured language, specified in XML, to define processes, patterns and standards.
- By using XML (Extensible Markup Language) and XSL (Extensible Stylesheet Language), it is possible to achieve reuse, of the process content, structure and presentation.
- A processes organization that allows for the creation of a process which can be the base for the generation of a standard process and whose definition can comply with certain models and norms, which are useful in certification programs.
- A framework for Requirements Engineering and its use definition.
- A systematic scheme of process database organization and access to search and selection.

Our future work involves the development of case studies to validate the architecture. To exemplify the proposed architecture we are finishing a WEB tool. The tool will make possible that a collection of patterns be available for use. We will start with process patterns, process standards and usual process for requirements engineering and will use a questionnaire driven method to measure the level of reuse that potential users of the tool may have achieved [41].

References

[1] Osterweil, L.: Software Process are Software too, Proceedings of the 9th International Conference on Software Engineering (1987)

[2] Ambriloa, V., Conradi, R., Fuggetta, A.: Assessing Process-Centered Software Engineering Environments, ACM Transactions of Software Engineering and Methodology, 6 (3), (1997) 283-328

[3] Karrer, A S. and Scacchi, W.: Meta-Environment for Software Production, Information and Operation Management Departament, University of Southern California (1994)

[4] Guimenes, I. M. S.: Uma introdução ao Processo de Engenharia de Software, Universidade Estadual de Maringá, Paraná, Julho (1994)

[5] Paulk, C.M., Curtis, B., Chrissis, M. B., Weber, V.C.: Capability Maturity Model for Software, Ver. 1.1, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-93-TR-24, ESC-TR-93-177 (1993)

[6] Konrad, M. and Paulk, M.: An Overview of SPICE's Model for Process Management, Proceedings of the International Conference on Software Quality, October, Austin, TX, (1995) 291-301
[7] The Trilhum Model, http://www2.umassd.edu/swpi/BellCanada/trillium-html/trillium.html
[8] Haase, V. et al, Bootstrap: Fine-Tuning Process Assessment, IEEE Computer, Vol. 27, Number 17, July (1994) 25-35 – http://www.bootstrap-institute.com/
[9] Ambler, S. W.: Process Patterns – Building Large-Scale Systems Using Object Technology, Cambridge University Press/ SIGS Books (1998)
[10] Wills, A. C.: Process Patterns in Catalysis, http://www.trireme.com/catalysis/procPatt
[11] Berger, K. at all : A Componentware Development Methodology based on Process Patterns, OOPSLA 98 (1998)
[12] Vasconcelos, F. and Werner, C.M.L: Organizing the Software Development Process Knowledge: An Approach Based on Patterns, International Journal of Software Engineering and Knowledge Engineering, Vol. 8, Number 4, (1998) 461-482
[13] Fontoura, M. F. M. C.: A Um Ambiente para Modelagem e Execução de Processos (An Environment to Model and Enact Processes), Master Thesis, PUC/RJ (1997)
[14] Coplien, J. O.: A Generative Development Process Pattern Language, in PloP (1995)
[15] SEI, Process Maturity Profile of the Software Community 1999 Year End Update, SEMA 3.00, Carnegie Mellon University, March (2000)
[16] Henninger, S.: An Environment for Reusing Software Processes, International Conference on Software Reuse, (1998) 103-112
[17] Hollenbach, C., Frakes, W.: Software Process Reuse in an Industrial Setting, Fourth International Conference on Software Reuse, Orlando, Florida, IEEE Computer Society Press, Los Alamitos, CA (1996)
[18] Pree, W.: Framework Patterns: Sigs Management Briefings (1996)
[19] Buschmann, F., Meunier, R.: A System of Patterns, in PLoP (1995)
[20] Coplien, J. O.: Software Patterns, SIGS Books & Multimedia, USA (1996)
[21] Gamma, E., Helm, R., Johnson, R., e Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Design, Addison-Wesley (1995)
[22] Hollenbach, C., Frakes, W.: Software Process Reuse in an Industrial Setting, *Fourth International Conference on Software Reuse*, Orlando, Florida, IEEE Computer Society Press, Los Alamitos, CA (1996)
[23] Whitenack, B., RAPPeL: A Requirements-Analysis-Process Pattern Language for Object-Oriented Development, *in PloP* (1995)
[24] Finkelstein, A., Kamer, J., Nuseibech, B.: *Software Process Modelling and Tecnology*, Research Studies Press Ltda, 1994
[25] http://www.w3.org/TR/REC-xml
[26] Alexander, C.: The Timeless Way of Building, New York: Oxford University Press (1979)
[27] Fiorini, S.T., Leite, J.C.S.P., Lucena, C.J.P.: *Describing Process Patterns*, Monografias em Ciência da Computação no. 13/99, PUC-Rio, Agosto (1999)
[28] Fiorini, S.T., Leite, J.C.S.P, Lucena, C.J.P.: Reusing Process Patterns, *Proceedings of the Workshop on Learning Software Organizations*, Oulu 2000
[29] Fiorini, S.T., Leite, J.C.S.P, Macedo-Soares, T.D.L.: Integrating Business Processes with Requirements Elicitation, *Proceedings of the 5th Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WET ICE'96) (1996)
[30] Kang, K. et al., Features-Oriented Domain Analisys – Feasibility Study, *In SEI Technical Report* CMU/SEI-90-TR-21, November (1990)
[31] Lator, L.,Wheeler, T., Frakes, W.:Descriptive e Prescriptive Aspects of the 3C's Model: SETA1 Working Group Summary. *Ada Letters*, XI(3), pp 9-17 (1991)
[32] Prieto-Díaz, R.: Implementing Faceted Classification for Software Reuse, Software Engineering Notes, Vol. 34, no. 5, May , (1991) 88-97

[33] Appleton, B.,*Patterns and Software: Essential Concepts and Terminology*, http://www.enteract.com/~bradapp/
[34] Coplien, J, O, Schmidt, D. C.: Pattern Languages of Program Design, (First PLoP Conference), Addison-Wesley (1995)
[35] Landes, D. Schneider, K., Houdek, F.: Organizational Learning and Experience Documentation in Industrial Software Projects, *Proceedings of First Workshop OM-98* (Building, Maintaining, and Using Organizational Memories), Brighton, UK, August (1998)
[36] DeLano, D. Rising, L.: System Test Pattern Language, http://www.agcs.com/patterns/papers/systestp.html
[37] Meszaros, G. and Doble, J.: A Pattern Language for Pattern Writing, http://hillside.net/patterns/Writing/pattern_index.html
[38] Henninger, S.: An Environment for Reusing Software Processes, International Conference on Software Reuse, (1998) 103-112
[39] Henninger, S.: Using Software Process to Support Learning Organizations, Proceedings of the Workshop on Learning Software Organizations, Kaiserslaurten, Germany (1999)
[40] Rolland, C., Prakash and Bejamen, A.: A Multi-Model View of Process Modeling, Requirements Engineering, 4, (1999) 169-187
[41] Fiorini, S.T., Leite, J. C. S. P.: *Arquitetura para Reutilização de Processos* (Process Reuse Architecture), Tese de Doutorado (PhD Thesis), Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio, Departamento de Informática, (2001).