JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2008**

Vol. 7, No.7 , September-October 2008

Methodological Proposal for Business Process Management sustained in the use of Patterns

Pedro Bonillo, Nancy Zambrano and Eleonora Acosta
Central University of Venezuela, Caracas, Venezuela

Abstract

At the moment, enterprises require complex business models with an organizational structures, processes and systems that must be explicitly designed. The work designed by these business models is clearly interdisciplinary, since it requires business development knowledge, different processes enterprises, management of these processes, and technological applications. In the scope of the software engineering would be desirable to obtain a system of methods, tools and techniques that allows the reuse of the best practices during the process of software development according to each one of the processes that are implemented in each domain. This work describes a theoretical methodology proposal framework. The methodology includes from the analysis of the requirements to the monitoring of the processes, supporting the analysis stages, design, model and configuration, through the use of patterns. The methodological proposal is conformed by two macro-processes: one related to the creation of the process itself and other that corresponds to the administration, and it includes: the maintenance, administration of the process in production and the monitoring through management indicators.

1 INTRODUCTION

The main purpose of this work is to propose a methodology that allows the Management of the Business Processes (BPM), based in the use of patterns [Alexander et. Al.77] [Gamma et al.95] [Buschmann et. Al 96]. This Work propose a taxonomy of patterns and its representation through an Architecture Definition Laguages (ADL) into an architecture of processes, services, and canon objects in the BPM domain. In addition, it opens out the pattern especifications[Acosta, Zambrano04], in order to be able to measure its quality through Attribute-Based into Architectural Styles (ABAS) [Kazman. Klein04]. ISO14-598 and ISO-9126 are used as a quality models of procesess and as the product. Considering this combination of methods, tools and techniques, it proposes a group of steps that in the BPM scope, allows to identify the key processes, to model them and to analyze them, to simulate them, to implant them in an assisted way (Such as new processes as their versions); to evaluate them, monitore them and improve them.

Construction of Software Based on Components

In literature, exists several definitions for "component" In [D' Souza, Wills99] it is understood by "component" as a coherent package of code that: (i) can be developed and distributed independently, (ii) has explicit and right specified interfaces to the service that it offers, (iii) has explicit and right specified interfaces to the services that are reliable from other components; and (iv) can be formed by other components, maybe opens out some of its properties, but without modifying the component itself [D'Souza, Wills99]. A more general definition is given by Jacobson, Griss and Jonsson[Jacobson et.al97] who defined it as an device that has been developed specifically to be reuse (this definition will be the one which this article will adopt). In this case a component could be as much as a case of use as any other reusable element that will rise during the development processes and it will be used in any activity, whenever it does not require knowledge of the software that uses it.

In the Objects Oriented (OO) approach, the objects can be seen as components, unless they satisfy additional guides directed to make auto-contents for them, the use of another components by means of aggregation and generally they interact with other components through the events.

The component used in the development process of software has three main targets: the reusability, the adaptation and the extension (the reusability can imply the adaptation and extension), understanding that:

- A component is reusable since its services can be used by another application.
- A component is adaptable if its supplier has anticipated the possible changes that can suffer this component, and can make it compatible with other hardware and software platforms.
- A component is extending whether its supplier gives the mechanism to modify or extend the services that the component offers.

Last year, we studied among others, two ways of the processes for software development: the agile and heavy methods. The fundamental difference between both of them is that while heavy methods try to obtain the common objective by means of order and documentation, the agile methods do it improving the direct and immediate communication processes between the people who are within. The development processes to consider in this proposal are: Unified Process (UP) as a method of heavy development, XP (eXtreme Programming Project) and FDD (Feature Driven Development) in representation of the agile methods.

If the project is sufficiently large to suggest the adaptation of components from previous developments, it is possible to say that UP is appropriate for the process of software development because it allows to obtain a better structure and discipline. A good possibility of reducing the work is with the reusability of models and processes already defined in a previous implementations of UP in different scopes. Referring to the architecture, XP with the system metaphors tries to determine an optimal architecture in the early stages of its development.Even though FDD is centered in the quality, it leaves all the weight of the architectural decisions to the main architect, but it does not specify how these decisions are related with the quality of the developing system.

On the other hand there is a generalized problem for the development processes UP, XP and FDD: the selection of the suitable architecture or combination of different architectural styles for a software system, is a problem that has not been solved yet and that it has been treated widely in literature. In relation to this problem, the use of components in these software development processes allows:

- In relation to obtain requirements: (i) the vertical analysis, focus in a domain or in a specific area of business; which its objective is that the resulting components can later become standards for any developed application in that domain and points out towards its reusability; (ii) the horizontal analysis, made in a generic way to give service to an ample rank of applications, without restricting itself to a domain of a given business; and (iii) specific analysis, made in a concrete domain, to obtain components "ad hoc" where the emphasis does not make so much in the reusability but in the extension.
- In relation to Design: during the gathering of requirements all the functions that had been identified and must be supported, but the distribution of these functions can occur immediately or can be constructed adapting the existing components. The following aspect corresponds to the partition of components, which can be made through: the use of cases, the patterns of design, the organizations of the domain, the anticipated evolution of the system, and the already existing components.
- Regarding the interaction of components: it is said it is direct (simple interaction) when the offered service adjusts to the forms and necessities of the required service. If the forms are not the suitable ones, it is necessary to previously make a packing process ("wrapper").

The growth of the systems complexity, usually constructed through the integration of components, increases the necessity to obtain more rigorous approaches than lead this process of decision. UP, XP and FDD have absence of a clear relation between the components (between these, the patterns), the architecture and the characteristics of the quality associated to the architecture. UP on the other hand, doesn't have an association of the nonfunctional requirements with the use of cases; a bad selection of the main use cases affects the architecture of the system; the model used test to evaluate the architecture doesn't have guides precise to determine this relation. In the next section, there will be a presentation of a new form to relate these concepts, to solve this deficiency.

Establishing the relationship between Patterns, Components, Applications, Processes, Methodology and Quality

In the Fowler book[Fowler97] there is an interesting generic definition: “a pattern is an idea that has been used in a practical context and probably it will be useful in others”. The main idea expresses that a pattern can be any “thing”. The expression in the practical context reflects the fact that it is developed (some authors prefer: discover) due to the practical experience of real projects. Considering this general definition of patterns, it is possible to affirmed that such can be expressed through components and these, as well as functions that are implemented in different applications (Figure 1).

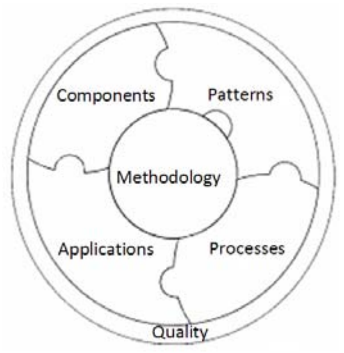

Fig 1: Relation Patterns, Components, Applications, Processes, Methodology and Quality.

In figure 1, the notches represents a “through” expression, which means that the patterns express themselves through components, the applications through components and patterns, the processes through applications and patterns, all of these within the framework of a methodology (a system that joins the concepts) and where each element must have associated with quality.

The concept of patterns in software engineering [Gamma et al.95], was introduced with the design patterns. At the moment, the software patterns constitute a more general concept, representing applicable conceptual structures in the diverse phases of the development process. In the following section, a new taxonomy of software patterns are set out.

Taxonomy of Patterns

A pattern is a solution to a problem, accepted as correct, which has been received a name and that can be applied in other contexts [Fowler97]. A pattern captures the experience and knowledge of experts, who have produced successful solutions to problems, giving the disposition of a soclution to those with less experience; nevertheless, the patterns do not always provide the definitive solutions, sometimes, the users of patterns must have creativity to use or implement a pattern [Acosta, Zambrano04].

In todays the Software engineering, the patterns can be applied at the level of: analysis of requirements, architecture designs, detailed designs, the interaction with the users and codes. With these the following classification or taxonomy can be set out, according to the abstraction level:

- Patterns Analysis: They are groups of concepts that represent a common construction of the conceptual model in the world. Can be relevant to a domain or adapted to many domains. The main idea is the construction of scenes using patterns. It is essential to try to have one more conceptual and structural vision of the situations, with the purpose of identify the intrinsic nature of the same ones. With that vision, it is possible to determine the type of scene corresponding to each situation and choose a pattern of a catalogue, reusing his structure with the purpose of deriving the easiest and directly scene. They consist in a text guide which for each scene component it includes guidelines about the content that must have the same one. They are presented as described scenes which a reduced number of conformation rules has been added as scene meta-components. Each component of the scene, according to the structure defined in [Leite00], has been completed with a nominal text that is expected to be replaced in the real scene generated when the pattern is used, but it as well guides the writing of the component [Ridao01]. For example, the analysis pattern to make a productive activity is applied to the scene to design a meeting agenda such as a form to reuse the characteristics of a productive activity to the specific domain of a meetings agenda.
- Patterns Architecture: They are fundamental organization schemes in a software system. They specify a series of subsystems and its respective responsibilities. It includes the rules and criteria's to organize the existing relations among them [Klein, Kazman99]. Some examples are: layers, filters and connections; models, views and controls (MVC). The architectonic styles are a generalization of the architecture patterns because they express the components and the relations of the structural and general skeleton of an independent application of the context and other styles; they are the categorization systems [Klein, Kazman99]. Some typical styles are the architectures based on data flow, those of implicit invocation, hierarchic and centered in data or those of virtual interpreter-machine.
- Patterns design: They are patterns of a level of abstraction smaller than the architecture patterns. They are therefore, next to which would be the final source code. For example: "abstract factory", "constructor" and "chain responsibility" [Buschmann et al.96].
- Patterns Interaction: also known as Patterns Interface describes a successful solution to a recurrent problem concerning the user's interface in a given context. A Pattern of Interaction is a way communication that is expressed in a simple annotation, in order to be understood by the people of the interactive design team that is generally multidisciplinary [Mahemoff, Johnston98]. Some examples are: Data formats of dates, hierarchic visual representations of the systems state [Van00].

- Patterns Implementation: They talk about the form to program or to implement a solution in a specific language, it is associated with the term *kit* or *idiom*.

From this hierarchical structuring, it is precise to notice the differences that exists between the types of patterns themselves and from the architectural styles, it is related to its level of abstraction (isolated patterns versus families - languages or catalogues - of patterns).

In general, the patterns have a limitation: they are difficult to specify and to evaluate from a model of a particular quality. For the specifying problem and the taxonomic relation, the use of ADL (Architecture Definition Language), according to the suggested hierarchic proposed and as far as the evaluation of the quality, proposes to use one of the first initiatives of quality measurement in the patterns: ABAS (Atribute-Based Architecture Styles) [Kazman, Klein04], as structures that extend the representation of the patterns, with the purpose to specify the information on themselves and the relative characteristics of quality. These subjects are considered in the next section.

Representation of Patterns and its Taxonomy, through a Language of Definition of Architectures

ADL (*Architecture Definition Language)* is a descriptive language that concentrates in the structure of high level application before the details of implementation of its concrete modules [Cod, May99]. A consensual definition of ADL doesn't exist until today, but commonly it is accepted that an ADL must provide an explicit model of components, connectors and their respective configurations. It is considered desirable, in addition, that an ADL provides the support of tools for the development of solutions based on architecture and its later evolution.

A Shaw study [Shaw,Clements96] analyzes the complex influence of the theory and the practice of the patterns on the ADL. This author considers that the ADL have been own of the software architects community, while the patterns and their respective languages have prospered between the software designers, particularly among the groups joined to objects oriented in designs. Naturally, both communities overcome themselves. Concerning to the relation between architecture and design, the maintained discussions have reached a consensus in that these designers operate at abstraction levels lower than the architects, but over the programmers levels[Shaw, Clements96]. On the other hand, Buschmann has documented patterns that are used as architectonic styles governed by ADL [Buschmann et al.96]. Shaw and Clements concluded their analysis alleging that the ADL can benefit incorporating analogous type of elements to the patterns in the sections that refer about styles, groups and rules of design [Shaw, Clements96].

The representation of an architecture of processes, canonical services and objects for its subsequent administration, implies the use of a ADL or a modeled generic language as it could be UML, this would allow to make the association that is explained in figure 1: relating the taxonomy of patterns before described, with the components and the applications, through the process, according to a methodology. In this association, it is necessary to establish measures that allow studying the quality and the audit process of the software construction and its product obtained through it, which is the reason why the next section is the ABAS subject [Kazman, Klein04].

Evaluation of the Quality in the Patterns of Software

An Attribute-Based Architecture Styles, (ABAS) is an information structure, a group that contemplates the pattern descriptions with the quality attributes; it was propose by Kazman, Klein, Barbacci, Longstaff, Lipson and Carriere [Kazman et al.98]. It is defined with a three elements base : (1) the topology of the types of components, and a description of the patterns of data and control, forthe interaction between the components (as in the standard definition); (2) a specific attribute of quality in a quality model, and; (3) the reasoning resulting after applying the attributes of an specific model of quality in the interaction of the types of components.

With the purpose of incorporating the concept of ABAS in the taxonomy of patterns exposed previously on this article and taking as a reference the patterns representations from interaction on the base of meta-pattern proposed by Acosta and Zambrano [Acosta, Zambrano04], the following extended representation is obtained:

Name (title), author classification, domain (well-known uses) and rank.	Name: Central idea by which the pattern is identified. Author: Name of the person who created the pattern. Classification: Type of pattern according to the taxonomy before mentioned. Dominion: Indicates the domain or the domains in which the pattern has been implemented. Rank: it is the degree of trustworthiness of this pattern with respect to the domain in which it has been implemented.
Problem	Describes the problem that will be solved, from the point of view of the user.
Solution	Describes, in a narrative and graphical way, the solution to the problem. With a base in: Objective, Intention, Motivation, Actors or Participants, Resources, Structures, Episodes, Collaborations and Implementations.
Attributes of Quality	Attributes of quality of interest, the context of use, contrasts and relevant attributes (specific requirements).
Measures of attributes of Quality	A summary about what is discussed in the description of the problem, using specific terms related to measurable aspects of the attributes in the quality model. This includes a discussion of events that could cause that the architecture responds or changes.

Parameters of attributes of Quality	A summary about what it will be discussed in the solution section, but specifying relevant terms to the parameters of the specified attributes in the quality model.
Analysis of the Model of Quality	A description of how the attributes of the quality model will formally be related to the elements of the pattern and the conclusions on the architectural behavior that it is obtained with the model.
Context	Presents the conditions under the used pattern.
Force	Conflicts that could restrict the solution. Including the exceptions.
Consequences	Describes the result to apply the pattern.
Examples/Against-Examples	Shows examples and against-examples of the proposed solution.
Related Pattern	Other patterns that are related to the pattern described.

Table 1 Meta-pattern adapted from [Acosta, Zambrano04] and [Kazman, Klein04]

It is important to mention that the usability of the pattern indicated in the representation of Acosta and Zambrano [Acosta, Zambrano04] is, in this case, a quality attribute. In addition, a pattern does not require, generally, all the elements mentioned before. In order to fulfill its objective, a pattern must have as a minimum the following elements: Name, Problem, Solution and Context.

Business Process Management (BPM)

Business Process Management is an ancient problem. This article tries to study it according to the new technologies, being considered to the potentialities of a model that includes re-usable components.

In the scope of the business processes, the technological solution by excellence talks about the term “workflow”, which is the process through the individual tasks that is coordinated to complete a transaction (using the defined processes of the business) within an organization. Workflow is a set of mechanisms that automate the work processes. These mechanisms, related to each other aspects of the administration, establishes priorities between the diverse tasks of each employee and optimize the communications between the different operative units. To obtain this, it is necessary to define which are the different tasks that are built in an organization; who participate in their execution; who are responsible for the same ones; which is the sequence of tasks of each processes and which are the actions that initiate each process. Although the contribution of the traditional workflow (modeled by the WFMC Workflow Management Coalition), it is still remarkable that there is a new generation which perhaps a hybrid that reunites the better things of all the “workflow” systems and other technologies: Business Process Management Systems (BPMS).

The BMPS incorporates ample capacities of integration with modern Java architectures, Net and XML. Additionally, they add other technologies such as Web Services, Business Rules Motors, Business Activity Monitoring (BAM) and Business Process Optimization (BPO). In agreement with Howard Smith and Peter Fingar [Bell03], guaranteed by BPMI (Business Process Management Initiative) and the WFMC, nowadays it can be affirmed that “the BPMS allows the companies to model, implement and manage the business processes, including enterprise applications, departments, and suppliers (“partners”), but without a referential frame integrated” [Bell03].

The BPMS have evolved since the integration of business architectures, where it is contemplated to the transformation and routings of data, administration of events, automatization of processes and the use of adapters in the 90’s; to the integration in the 2000 of the business processes through the basic model of the processes, the management of the suppliers, connectivity between companies through the ecommerce and formation of certain groups of processes for vertical industries; until arriving at the actual concept (since 2004) involved: applications of workflow, sophisticated model of processes, monitoring of the activities associated to the business processes (BAM), exhibition of the functions of the applications through web services, use of business rules management, support to multiple devices of access to the information in any place and from any position (“aware-contex”) through the use of portals, use of management tools of the life cycle of the software application development that supports the processes, mobile support of the processes and the interfaces, extraction, transformation and load (ETL) of data that are used by the processes and the capacity of simulation on the processes and versioning of them (Bussines Process Optimization BPO).

In spite of all the characteristics before mentioned, the BPMS lacks from a global and integral framework that establishes a methodology for its implementation and use. According to its implementation, most of the software patterns are not supported in a precise form. The market of the BPM architectures tends to concentrate in system flows to system and it is emerging slowly as far as the human-human flow attended by the computer [Bell03]. Based on this, BPMI is the organization which assumes the elaboration of standards (BPA, BPMN and BPMS; analysis, notation and semantics respectively) that sustains the BPM concept focusing on the business process such as the start point between the environment of it and putting it in practice through the technology. At the moment Workflow Management Coalition is being unified with the BPMI and with OMG (Object Management Group).

All the aspects discussed and the deficiencies shown above,lead to the elaboration of the methodological proposal.

2 METHODOLOGICAL PROPOSAL

In this section, there will be a description of the methodological proposal for business process management sustained in the use of patterns, as a form to solve the problems before they appear. Initially, appears an integral theoretical framework; following it with the appearance of the methodological proposal making aspecial emphasis in the aspects of the generation of applications, through the software engineering.

Integral Theoretical Framework

In this integral theoretical framework is outlined a possible solution to the problems shown in this investigation.

As shown in the taxonomy of patterns in figure 2, there is a relation between the abstraction level, the type of patterns, and the standard suggestions by BPMI, which obtains a direct relation of taxonomy of patterns with the domain where the methodology sets out (BPM). In addition, the taxonomy of patterns is also associated with its formal definition through an ADL establishing the relation between patterns, components, applications and processes through a methodology. Finally, this taxonomy of patterns is associated with the necessity to measure the quality of each pattern and to establish a mechanism of auditing in the use of the patterns, providing them of specifications of quality with the ABAS use, within the framework of a quality model (ISO9126 for the quality of the product and ISO14598 for the quality of the process).

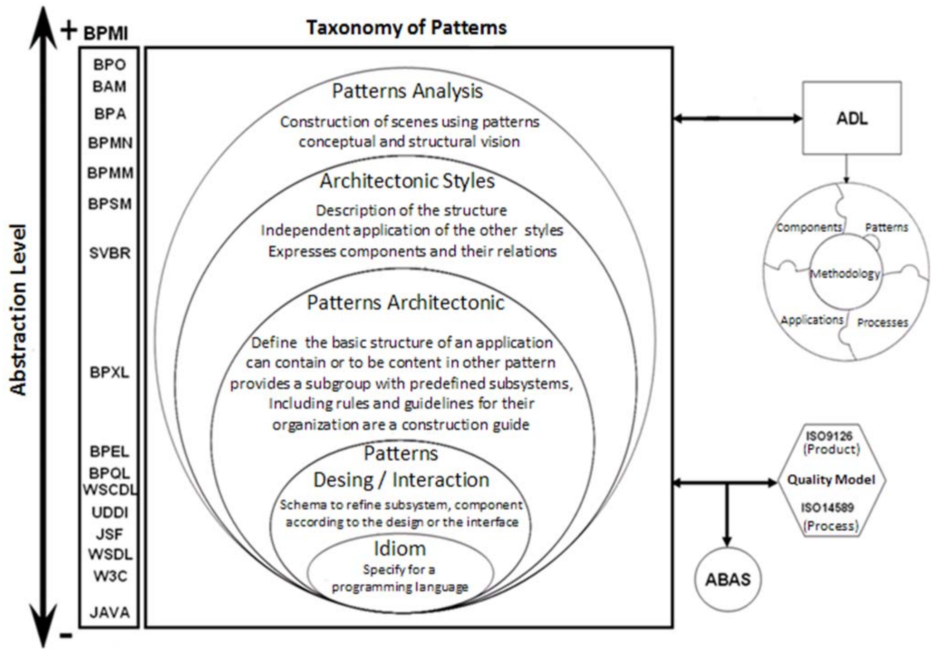
+ BPMI
Taxonomy of Patterns
BPO
BAM
BPA
BPMN
BPMM
BPSM
SVBR
BPXL
BPEL
BPQL
WSCDL
UDDI
JSF
WSDL
W3C
JAVA
-
Abstraction Level
Patterns Analysis
Construction of scenes using patterns conceptual and structural vision
Architectonic Styles
Description of the structure
Independent application of the other styles
Expresses components and their relations
Patterns Architectonic
Define the basic structure of an application
can contain or to be content in other pattern
provides a subgroup with predefined subsystems,
Including rules and guidelines for their
organization are a construction guide
Patterns
Desing / Interaction
Schema to refine subsystem, component according to the design or the interface
Idiom
Specify for a programming language
ADL
Components
Patterns
Methodology
Applications
Processes
ISO9126 (Product)
Quality Model
ISO14589 (Process)
ABAS

Fig 2: Integral theoretical framework of the Methodological proposal

The specification, through an ADL, of the levels of abstraction of the BPM architecture that sustains the propose methodology, allows the representation in four layers of the same one: layer of processes where the orchestation is made of itself, layer of services (represent the canonical objects and the services as functions of the applications), application layer (applications, components and software) and layer of technology (hardware). These layers allow the identification of three specific architectures: services, components and applications architecture.

In special, the architecture of services contains the services of: infrastructure management, administration of suppliers or associate, own business applications, applications of legacy, interaction, processes of business, information and connectivity (which allow the communication between all the services mentioned). In a level of abstraction superior to these services, there are services of management indicators and software development, which had the rolls such as business analyst (modeler of the process), architect, specialist of integration, developer and analyst of tests.

The process associated to the software development services has a direct relationship with the process since it manages other one; this process, as well, is defined in the services of process and is the conductor that will allow to administer of integral form the BPMS and that defines a cycle for the management of the business processes that consist of the identification of the key tasks, the model and analysis of the tasks, the simulation and implantation of new processes, and their evaluation and monitoring.

Description of the Methodological Proposal

1. The methodological proposal for the business processes management sustained in the use of patterns is conformed by two macro-processes: (1) Creation of the business processes; and (2) Administration of the business processes in execution.
2. The first macro-process includes the following sub-processes:
3. Analysis and evaluation of the requirements considering the type of priority and the base of the elicitation practices requirements of the software engineering. In this sub-process, the user asks for a requirement on an existing process or a new one, the analyst of the business identifies the requirement according to the practices of elicitation of requirements of the software engineering (it describes the requirement in terms of the functional requirements and nonfunctional, it assigns a priority for it and evaluates a billboard of the possible activities to make in the time). Next the analyst of the business consults the patterns of analysis available in domain and relates them to the requirement. Finally, for any other necessary information that it rises, it invokes a process of change management in order to introduce the new requirement in the existing architecture of processes;
4. Design of a standardized model of the process according to the patterns of analysis in the domain (better practices). In this sub-process, the analyst of the business compares it according to the requirements of the user, with the analysis patterns (better practices domain), in order to identify the existing breaches and the organizational risks to assume. Later, it negotiates the risks with the client; the

analyst of the business makes a design of the process model according to the architectural styles and the architectural patterns that are associated to the selected patterns analysis or the specific requirements of the user. In order to finalize this sub-process, the analyst of the business defines a definitive list of activities in the time and invokes to the process of software development, which the architect selects a development process (UP, XP or FDD) and configures a development environment (hardware and software).

5. Modeled, diagram and simulation. In this sub-process, the analyst of the business makes a diagram of the process designed in a BPA tool (selecting the already existing diagrams for the patterns of analysis, architectural styles and chosen architectural patterns in the sub-process of design). Next, the analyst of the business makes a simulation in the BPA tool in order to detect tendencies (necks of bottles, cycles, etc.) and asks the user for the approval of the process simulation. Then, the analyst of the business joins the actual diagram process with the existing general diagram for the architecture of all the processes (relating the present process to the existing processes) and makes a simulation again in order to verify the tendencies with respect to the general architecture of processes, asks to the user for the approval, makes a closing report of the design and the modeled one of the process, and indicates the architect whom exports the BPEL (Language of Execution of Processes proposed by the BPMI) from the BPA tool and the UML (Unified Language of Modeling) who will be input for the software development process.
6. Configuration and implementation of the logic of integration, businesses and presentation of the process through the orchestration of services, objects and the use of patterns of design and interface with base in the modeled phase. In this sub-process, the architect takes the UML diagrams created by BPA tool, and the functional and nonfunctional requirements from the analysis of process requirements, and it completes the UML diagrams according to the process of the selected development (UP, XP or FDD). Later, from the UML diagrams, the BPEL and the patterns of design related to the analysis patterns, architectural styles and selected architectural patterns in the previous sub-processes, the architect asks for the developer that it constructs the logic of business of the application that will sustain to the process (the developer besides to select the patterns design, also selects the idiom in which it made the development of the business logic); similarly according to the patterns interaction related to the analysis patterns, the architectural styles and selected patterns architectural in the previous sub-processes, the BPEL and the associate interfaces automatically with the BPEL through the BPA tools the architect asks for the developer that constructs the interface of the tool that supported the process (the developer besides to select the patterns interaction also selected idiom in which it made the development of the interface). Then, the architect asks for the integration of a specialist who makes integrations between the BPEL, the interface and the logic of business. Having the BPEL, the logic of business and the interface integrated, the architect proceeds to place in the BPEL the indicators management that will

be measure the process, for this it selects of a set previously defined according to the analysis pattern, the necessary indicators and it as well asks for the developer that constructs the reports necessary to show the indicators (the BPMS contains a modulate of BAM which store these indicators in the time according to structures of business intelligence). Finally the architect asks for the tests analyst the software certification product that sustains the process, soon to finalize the process of change management of the architecture of processes and to solicit collocates it to production of the process (it is precise to mention that different versions from a same process can be handled).

7. The other macro-process corresponds to the administration of the processes in execution and includes: the maintenance, administration of the process in production; and the monitoring, validation of the technical-functional data of the processes implemented through indicators management.

3 CONCLUSION

It can be concluded that at the moment it doesn't exist a treatment sufficiently exhaustive and consolidated of the subjects of business process management sustained in the use of patterns, according to the audit and the quality of the process and the product of the software. By means of this investigation these deficiencies are attacked, and propose the construction of:

8. A language of patterns for the methodology of business process management with a base in the construction sustained in software components: a study of the existing alternatives for the selection, application and verification of patterns and the composition of components of software from the point of view of the conditioners raised here (audit, verification static, automatic and reasonable) contemplating to the model object and the architecture of services for the processes;
9. An attended modeled method of applications of software that allows the audit in diverse stages of the software life cycle. This method of modeled and verification have in addition special characteristic: (a) it can be used by a development organization, without knowledge on formal methods; (b) it foments the collection and uses a knowledge that is habitually lost; (c) it allows to manage this knowledge with the necessity to integrate it in the source code of the developed programs, and (d) it is not joined to any language of specific development nor any specific intention.

- A system of practical viable verification of components: (a) the tools can be developed with base in network technologies and (b) the necessary basic knowledge fits in the typical profile of the professional software development.

The future pieces will orient the construction of one wikipedia for the administration of taxonomy of patterns and its quality in the BPM domain, along with the specification of the ADL and a prototype of low level that supports the implementation of the methodological proposal through a case of study in the BPM domain.

REFERENCES

[Acosta, Zambrano04] Acosta, E. y Zambrano N.: Patterns and Objects for User Interface Construction, in Journal of Object Technology, vol. 3 no. 3 March-April 2004 pp. 75-90

[Alexander et al.77] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fidsdahl-King, I., Angel, Sh.: A Pattern Language. Oxford University Press, New York, 1977.

[Bell03] Bell, T.: A BPM Taxonomy: Creating Clarity in a Confusing Market. Gartner Research Note, Technology, T-18-9669. 2003.

[Buschmann et al.96] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., y Stal, M.: Pattern-Oriented Software Architecture: A System of Patterns. Wiley & Sons. 1996.

[Cod, May99] Coad, P., May, M.: JAVA Design: Building Better Apps and Applets. 2nd Edition. Yourdon Press, Upper Saddle River, 1999.

[D’Souza,Wills99] D’Souza, D., y Wills, A.: Objects, Components and Frameworks with UML. The Catalysis approach. Addison-Wesley. 1999.

[Fowler97] Fowler, M.: Analysis Patterns: Reusable Object Models. Addison-Wesley, 1997.

[Gamma et al.95] Gamma, E., Helm, R., Johnson, R., y Vlissides, J.: Design Patterns. Addison Wesley. 1995.

[Jacobson et al.97] Jacobson, I., Griss, M., y Jonsson, P.: Software Reuse. Architecture, Process and Organization for Business Success. Addison-Wesley. 1997.

[Kazman, Klein04] Kazman R., y Klein, M.: Attribute-Based Architectural Styles. Carnegie Mellon Software Engineering Institute Technical Report CMU/SEI-99-TR-022, 2004.

[Kazman et al.98] Kazman R., Klein M., Barbacci M., Longstaff T., Lipson H., Carriere J.:The Architecture Tradeoff Analysis Method. CMU/SEI-98-TR-008, ESC-TR-98-008. 1998.

[Klein, Kazman99] Klein M. y Kazman R.: Attribute-Based Architectural Styles, CMU/SEI-99-TR-022, ESC-TR-99-022: 1-69. 1999.

[Leite00] Leite, J., Hadad, G., Doorn, J., Kaplan, G.: A Scenario Construction Process, Requirements Engineering Journal,Vol.5, N° 1 2000 pp. 38-61.

[Mahemoff, Johnston98] Mahemoff, M. y Johnston, L.: Pattern Language for Usability : An Investigation of Alternative Approaches. Asia-Pacific Conference on Human-Computer Interaction (APCHI’98) Proceedings, 25-31. Tanaka, 1998.

[Ridao01]	Ridao, M.: Uso de Patrones en el Proceso de Construcción de Escenarios. Tesis Maestría en Ingeniería de Software, Univ. Nacional de La Plata, Noviembre 2001.
[Shaw, Clements96]	Shaw M., Clements P.: How Should Patterns Influence Architectural Description Languages? A Call for Discussion. Computer Science Department and Software Engineering Institute Carnegie Mellon University. 1996.
[Van00]	Van Welie, M. y Troetteberg, H. Interaction Patterns in User Interfaces. 7th Pattern Language of Programs Conference, August 2000, Illinois, USA, 2000. URL: http://www.cs.vu.nl/~martijn/patterns/index.html

About the authors

Pedro Bonillo Ramos is a teacher and researcher at the Central University of Venezuela. He obtained his Master in Computer Sciences at the Simon Bolivar University in Venezuela in 2002 and his Master in Business Administration at the Yacambú University in Venezuela in 2004. Currently, he is working on his PhD in Computing Sciences doing research about Software Engineering mainly focusing in Business Process Management. He is also working on his other PhD in Management Sciences doing research about Value Data Maps. He can be reached at pbonillo@cantv.net

Nancy Zambrano Rivas is a teacher and researcher at the Central University in Venezuela. She obtained her Master in Computer Sciences at the Central University of Venezuela in 1989. She obtained her PhD. in Computing Sciences, Informatique at the Laboratoire de Recherche en Informatique (LRI), Université Paris-Sud XI, in 1995. She is researching about Software Engineering (light methods, heavy methods, object-oriented methods, Unified Process, software development methods based on transformations) and Human-Computer Interaction (interaction patterns, tendencies in interface). She can be reached at nzambran@ciens.ucv.ve

Alecia Eleonora Acosta is a teacher and researcher at the Central University of Venezuela. She did a D.E.A d'Informatique at the Université Paris-Sud XI, France, in September 1992. Later, she obtained her Master in Computer Sciences at the Central University of Venezuela in 1993. She obtained her PhD. in Computing Sciences at the Central University of Venezuela in 2004. She is researching about user interface designs, patterns, usability and socialization of the computation. She can be reached at eacosta@ciens.ucv.ve