The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1463-7154.htm

Reflections on the modularity of business process models

The case for introducing the aspect-oriented paradigm

Claudia Cappelli and Flávia Maria Santoro
NP2Tec – Department of Applied Informatics, Federal University of the State of Rio de Janeiro, Rio de Janeiro, Brazil
Julio Cesar Sampaio do Prado Leite
Departamento de Informática, PUC-Rio, Rio de Janeiro, Brazil
Thais Batista and Ana Luisa Medeiros
Computer Science Department, UFRN, Lagoa Nova, Brazil, and
Clarissa S.C. Romeiro
NP2Tec – Department of Applied Informatics, Federal University of the State of Rio de Janeiro, Rio de Janeiro, Brazil

Abstract

Purpose – The aspect-oriented (AO) paradigm is first proposed to deal with programing modularity issues, but different researchers have been exploring AO concepts in the designing and definition of software systems. The goal of this paper is to discuss and present a proposal that addresses the application of AO concepts to the design of business processes (BPs) in order to improve usability and understandability of process models.

Design/methodology/approach – The paper departs from previous work on analyzing the application of AO for software design. The observations were backed by a case study, which was used to illustrate the issues by means of examples.

Findings – The paper presents findings on important issues related to the integration of AO paradigm and BP modeling, such as crosscutting representation, crosscutting composition, quantification, and join point exposure.

Originality/value – The paper explores a new frontier: the application of AO concepts to the design of BPs. As of now, few works have explored this new view on process modularity. The paper claims that application of AO concepts to the design of BPs is important in the consideration of usability and understandability. Its contributions are also backed by a prototype process editor, CrossOryx, a web-based editor for modeling process using AO concepts.

Keywords Process planning, Modelling, Computer software

Paper type Research paper

1. Introduction

Business process modeling (BPM) has been adopted by organizations in various contexts, including the support for best practices review and information system engineering and design. According to Aguilar-Savén (2004):

> [...] a business process is the combination of a set of activities within an enterprise with a structure describing their logical order and dependence, whose objective is to produce a desired result.

Business Process Management Journal
Vol. 16 No. 4, 2010
pp. 662-687

1463-7154
DOI 10.1108/14637151011065955

BPM enables common understanding and analysis of a BP and, as such, improves practice reviews and helps the design of information systems.

Notations and tools define the elements involved in process modeling and their interactions, such as process, activities, objectives, events, information, flows, actors, and groups. The BP model, the product of the process modeling, can be defined as a set of views that represent different perspectives of one or more specific aspects of the business (Melão and Pidd, 2000). Combined, these views provide broad understanding on the organization and its business, and serve as a basis for understandability about communication among internal areas and actors, or among the organization and clients, suppliers, government, and other organizations.

In general, most notations and languages presented in literature follow some basic functional decomposition principles and separation of concepts. According to the functional decomposition principle, the model must map the functions of the company in a hierarchical way in a top-down manner. The separation of concepts, in turn, emphasizes the need to model the process by composing it from the different kinds of entities related to processes, such as actors, events, decisions, and so forth.

Although these principles do help the organization of processes towards modularization, we argue that they are not enough to represent modularized concepts repeated in several parts of the same process or in several different activities overlapped in processes. These concepts are called crosscutting concerns (Filman *et al.*, 2005; Kiczales *et al.*, 1997), since they are interlaced with other basic concepts of a process and are scattered in various parts of the process model, interwoven with different elements which compose a process.

A crosscutting concern at the BPM level could be any concern which cannot be effectively modularized using the current abstractions of BP models. Crosscutting leads to a set of BPM elements representing a concern, being scattered and tangled all over the BP specification. This results in reduced understandability and reuse capability of those models and also in increased maintenance overhead. The introduction of a change in a specific part of a process, for instance, may impact several parts of the process. Therefore, the central research issue addressed in the paper is: how to build a modular process model that tackles the problem of scattered and tangled concepts?

Based on the use of the aspect-oriented (AO) paradigm (Kiczales *et al.*, 1997) to modularize crosscutting concerns scattered and tangled all over software specification and code, we propose AO-BPM to support better modularization of crosscutting concerns in BPM. AO-BPM separates the BP into:

- The basic BP (core process) which contains the essence of the BP.
- The aspect process which captures the crosscutting information cutting across the core process.

To the best of our knowledge, few researchers have considered the mating of AO concepts at the BP level, but only in preliminary form ([XXX]; Charfi, 2007; Wada *et al.*, 2008; Park *et al.*, 2007). Approaches addressing this issue at the software requirements level have in common the introduction of a whole new set of abstractions – usually derived directly from AO implementation techniques – thereby increasing the complexity of the conventional requirement abstractions. If the same type of approach is imported to BPM, this will place a burden on process designers. So, there is a lack of

knowledge on how, why, and which extensions are required for traditional process model elements, whether crosscutting concerns are to be considered.

Inspired on a previous paper ([KKK]) discussing the integration of AO and software design (architecture description languages – ADLs), this paper consolidates previous results by proposing a novel way of modularizing process models. We stress our proposal by presenting some essential issues associated to the integration of AO and BPM. Our main goal is to identify which extensions to conventional BP model abstractions are needed to modularly represent crosscutting concerns. We claim that those reflections can lead to a seamless integration of the two fields. We analyze six critical issues and use a running example about change management process to illustrate. We present also CrossOryx, a customization of the Oryx BP editor, to support crosscutting concern modeling and the composition of aspects and of the core process.

Our contribution lies on proposing a novel approach towards BPM modularity. We justify its needs, we make the rationale behind it explicit, we exemplify its use and we present an editor to support the approach. Our research answers the central question of how to provide modularity, which deals with extra functional concepts, that is, concepts spread over a process but which are accessory to the central process functionality. Providing the modularity to tackle these extra functional concepts is believed to improve understanding and evolution of these processes. Other researchers, Kueng and Kawalek (1997), Soffer and Wand (2005), and Wada *et al.* (2008), have mentioned the problem; however, our approach is novel as it proposes the inclusion of a new modularity mechanism at the level of process description. In a review by Recker *et al.* (2009), the issue of modularity is identified as "the cluster systems structured around things" and even BPM notation, the best-evaluated for the cluster, has problems. Our approach contributes to BPMN completeness regarding modularity.

The remainder of this paper is structured as follows: Section 2 provides background on modularity and discusses related work on BP and the aspect approach. Section 3 presents our research process. Section 4 presents our reflections on the integration of aspects and BPM, and a running example used to illustrate our claims. Section 5 presents the CrossOryx editor. Section 6 contains the final remarks.

2. On modularity

Modularization is a technique commonly used at software design and programing levels which separate a software in parts, the so-called modules, wherein each module represents a concern. According to Parnas (1972), the effectiveness of "modularization" is dependent upon the criteria used in dividing the system into modules.

Modularity is of fundamental importance in dealing with complexity, and, as such, it is very much related to abstraction (Prieto-Diaz and Neighbors, 1986). In this section, we provide a general overview of modularity using the perspectives of system thinking, software engineering, and process modeling.

As we reviewed the literature on modeling, we have found out that software engineering, systems theory, and BPM have several problems in dealing with modularity. In order to address these problems, we argue in favor of a new approach to modularity which has been an emerging field of study in software engineering. We refer to the AO ideas (Kiczales *et al.*, 1997) which explore the orthogonality of concepts across different sets of concepts. Others (Charfi, 2007; Wada *et al.*, 2008;

Park *et al.*, 2007) in the field of BPs have already considered using AO ideas, but, as we will discuss later in this section, in a restricted manner.

First, in Section 2.1, we discuss the modularity in the software field where the problem has been far addressed to help clarify the concept, and then, in Section 2.2, modularity in BPM context is argued. Section 2.3 introduces the formal concepts and ideas behind crossing concerns and aspect modeling. Section 2.4 discusses related work, pointing out specific proposals that come up with AO approaches for process modeling.

2.1 Software modularity

The idea of software modularity has been a central software engineering concept since its inception. Sub-systems, sub-programs, modules, components, procedures, functions, objects, and routines are some of the different names used in the programing and software design field to represent the idea of organizing a complex whole into different parts. Since partitioning is a central idea in dealing with complex artifacts, the initiative is present in different approaches towards software construction.

Modularity is very much related to what von Bertalanffy (1969) calls hierarchic order: "A similar hierarchy is found both in 'structures' and in 'functions'." In the last resort, structure (i.e. order of parts) and function (order of processes) may be the very same thing: in the physical world matter dissolves into a play of energies, and in the biological world structures are the expression of a flow of processes. Different computer programing languages use different strategies towards modularity, and these different strategies use different naming towards their constituent parts (Scott, 2009). In the realm of software design, a module denotes, according to Stevens (1991), a component which itself may contain subcomponents. The author states that this is a very imprecise term, also used to denote "not-monolithic," may be at a high- or low-level of detail, being used both as a basic tenet for large programs (Parnas, 1976), and as a central concept in architecture by means of module interconnection languages (Prieto-Diaz and Neighbors, 1986).

Achieving modularity is somehow a matter of how well organized the constituents of a whole are and how they relate. As well-noted by Bertalanffy (1969), the whole is more than the sum of its parts, since it must account for the relations among parts. The organization of a complex artifact will depend on several issues, such as the education of who is organizing (Abran *et al.*, 2001), their skills with dealing with abstractions (Kramer, 2007), and the purpose and the viewpoint of the organization (Ross, 1977).

However, achieving modularity is not an easy task. In particular, this task is difficult when one concept is spread across the whole. For instance, consider a simple system: we may say that it is divided in three parts – getting input data, processing the data, and displaying the data. Even in such a simple system, we may need to produce a log of what is being done in each part, so as to allow for future auditing. In such case, the log concept is the same which will be present in all three parts. So the log is a concept which crosses the input, process, and output parts of system. A possible solution will be just to make the log a different part, but, if it is considered as a part of the whole, it will be a very different part from the other three parts, since it somehow "belongs" to the other parts. As such, it is evident that its nature is distinct from the three other parts.

Bertalanffy (1969) and Abelson *et al.* (1985) highlight the limits/inadequacy of hierarchy as the cornerstone of modularity. Abelson *et al.* (1985) provides a simple example:

> [. . .] Triangles and quadrilaterals, for instance, are both subtypes of polygons. In addition, a type may have more than one supertype. For example, an isosceles right triangle may be regarded either as an isosceles triangle or as a right triangle. [. . .] Dealing with large numbers of interrelated types while still preserving the modularity in the design of large systems is very difficult and is an area of much current research.

In this case, the isosceles right triangle "crosses" both the isosceles triangle and the right triangle. The example shows the difficulties of dealing with parts that do fit in the whole in different manners.

Dealing with these difficulties is not only a matter of properly representing the design, but also identifying the parts of a whole which may be characterized as crosscutting concepts. Similar problems occur in the level of BPs.

2.2 Modularity on BPM

Despite its relevance in different contexts, modeling of BPs still presents challenges to organizations due to its complexity (Lin *et al.*, 2002; Melão and Pidd, 2000; Rosemann *et al.*, 2006; van der Aalst *et al.*, 2003; Recker *et al.*, 2009). Therefore, research has been made concerning aspects of quality to make process models more "understandable" ([XXX]). Mendling and Strembeck (2008) state that we know little about the act of process modeling, and about the factors contributing to a good process model, in terms of human understandability; besides the fact that, human modelers misplace the interrelations of large and complex models due to their limited cognitive capabilities. Within those works, we have noticed just a few which emphasize, specifically, the modularity factor.

Mendling and Strembeck (2008) discuss understandability of process models decoded in terms of model size as an enabler of quality and clarity, requiring checking whether models are correctly interpreted. However, they recognize that there are other factors beyond size that presumably affect the understandability of a process model, such as the degrees of sequentiality, concurrency, or structuredness. Although not explored by the authors one of the concepts associated to the structure is modularity.

Recker *et al.* (2009) argue that the process modeling techniques do not provide adequate support for process decomposition, which is very closely coupled to modularity. Their analysis indicates that overcoming deficits related to modeling classes of things could potentially lead to better representational support for decomposition principles and, for example, proper representational support could assist modelers in the design of multi-level process architectures, or process variants for different involved issues.

The research on visualization of process models is also closely related, since it is necessary to establish viewpoints about the process to be perceived by different stakeholders. In this context, Bobrik *et al.* (2007) present the process models visualization problem. Their approach deals with large organizations, in which the users have distinguished perspectives over the BP and the related data. They understand that personalized views to the managed process are needed and that existing tools do not provide adequate mechanisms for building and visualizing such views. Jablonski and Goetz (2008) talk about a very similar point and present the

visualization problem as a process modeler and user problem because they need visualize models in various ways depending on use. They identify five main perspectives for a basic process modeling language: functional perspective (identifies process steps), data flow perspective (defines data used in a process), operational perspective (specifies which operation is invoked in order to execute a process step), organizational perspective (defines agents responsible to perform process steps), and behavior perspective (define causal dependencies between elements).

Designing and modeling for flexibility are the way to handle complexity in process models; moreover, modularity is the key for managing complexity and flexibility (Bhat and Deshmukh, 2005). The authors summarize flexibility dimensions in process models (Table I) and the usual modeling methods that deal with each one. For those authors, it is important to represent the flow, rules, practices, and exception handling in a flexible manner such as independent parts (like modules) of the process model. For each dimension, there are specific methods that attempt to address the representation, but none is pointed out as being able to integrate all of them.

Vanderfeesten *et al.* (2006) argue that it is quite difficult, even for experts, understanding whether pairs of activities in a model with lots of parallelism and choices are exclusive or not. Furthermore, even if activities are on a directed path, it is not directly clear which other elements they depend on if there are lots of routing elements among them. Ould (2005) points out that, within an organization, there are many processes dealing with the different aspects of its existence, and all the activities are part of a continuously changing network of interacting processes. This author affirms that, if we wish to model an activity at the organizational level, consequently, an approach is necessary which captures those dynamic relationships, the network and the way it operates. For example, it would be important to represent two processes which operate independently, but interact at many points; also, where one process starts another, which, by its turn, operates independently. Thus, although it is a very popular modeling technique, hierarchical representation is inappropriate.

Kueng *et al.* (1996) argue that when modeling BPs, there are four key elements to be addressed:

- on a conceptual level: goals, activities, and roles; and
- on a pre-implementation level: objects.

The authors described modeling approaches supporting each of those elements, but concluded that all of these are still immature. One of the main reasons is that they do

Flexibility dimensions	Modeling methods
Flexibility of the process sequence	Process hierarchy and conditional elements
Flexible business rules	Externalization of rules
Flexible practices	Business tasks and norms
Flexible exception handling (anticipated)	Conditional elements, exception workflows, norms, and rules
Flexible exception handling (unanticipated)	Exception workflows

Source: Bhat and Deshmukh (2005)

Table I.
Application of modeling methods

not address the relationship between operational behavior and BPs managerial co-ordination, control, development, and policy. All these facets are different concerns of the process models, and, thus it would be necessary to represent them, besides indicating how they are interrelated.

We concluded that modularity is concentrated basically on levels of abstraction within process models, and, for lower granularity levels, definition of its atomic elements, which can be reused along the diverse diagrams. Crosscutting concerns which may occur among those diverse levels have not been addressed properly. As such, we understand the necessity of providing a careful analysis of the various issues related to BPM, including the crosscutting concerns. Any adaptation of existing modeling notations should be based on a clear understanding of new challenges at this level.

2.3 Identification of crossing concepts and the aspect approach

According to Silva (2006), it is better, instead of thinking of classes of concepts which usually cross other, to think that crosscutting is dependent on different situations and will usually be an option of the one who is designing a whole with parts. Silva (2006) recommends the following set of heuristics to classify a crosscutting concept:

- if the concept is repeated several times in different places;
- if the concept is used by different other concepts;
- if the concept reflects an integration of semantically distinct situations;
- if the concept represents a decision situation from which different options may be taken, and its absence does not interfere with the global objectives of the whole;
- if the concept can be reused in other domains; and
- if the concept is very much independent of the other concepts.

Notwithstanding the fact of characterizing crossing concepts as situation-dependent, the class of concepts concerning quality attributes bears the characteristic of being general, and, as such, will cross different other modularized parts. In the field of requirement engineering, this class of concept has been defined as non-functional requirements or soft-goals (Chung *et al.*, 2000; Chung and Leite, 2009). Usually, this type of concept is applied to different parts of a whole. In this case, identification of crossing concerns should make use of catalogues, in which quality attributes are operationalized (Chung *et al.*, 2000).

The literature on BP management has used the concept of soft-goals (Kueng and Kawalek, 1997; Soffer and Wand, 2005) to point out the lack of attention to general BPs characteristics. For example, Kueng and Kawalek (1997) affirm that:

> We need to be able to state what we want to achieve so that we are then able to define the necessary activities which a business process should encompass (i.e. goals are used to structure the design).

In the BP context, the general heuristic for finding these crosscutting concepts is bound to the ideas of strategic goals. As explained by Kueng and Kawalek (1997), their approach to find these crosscutting concepts is based on attempting to use the literate wisdom on the "goodness" of a designed process. Using this knowledge, the authors approach the "goodness" attributes from different users viewpoints to list pairs of

goals and means for a process. By following this process, the authors will be producing concepts to be applied to process so as to evaluate them.

There have been several works on understanding and identifying crosscutting concerns in software development since Kiczales *et al.*'s (1997) paper introduced the concept of AO programing (AOP). Based on the AO principles, AO software development (AOSD) aims to promote enhanced modularization and composition of crosscutting concerns through the software development stages.

In process modeling, for instance, the natural unit of modularity is a process, and a crosscutting concern is a concern that spanning multiple processes. Typical crosscutting concerns include logging, error handling, and security. As the crosscutting behavior is scattered across some processes and tangled with other concerns, it is difficult to find and modify it. As a consequence, process comprehension, evolution, and reuse are difficult even to business experts. An activity which represents logging, for instance, intertwines with other primary activities of a process. Using AO concepts, it is possible to modularize the crosscutting concerns, separately from the main process flow.

AOSD proposes the following abstractions (see Kiczales *et al.* (1997) for those definitions). Aspects are the elements designed to encapsulate crosscutting concerns and take them out of the core elements in a given specification or implementation. For instance, activities representing the logging concept can be modularized as an aspect. In order to specify the composition of an aspect with the main process flow, an aspect contains pointcuts and advices. Pointcuts are sets of join points. Join points are the core description elements which an aspect intercepts: thus, join points facilitate aspect composition. For instance, in the logging aspect, join points are main process flow elements where the aspect is applied. A pointcut defines an expression with quantification mechanisms to select one or more join points to be advised by an aspect. A pointcut language defines patterns to write pointcut expressions. Advices define the action which must be taken when a join point is reached. They act on a pointcut and can be configured to act before (before advice), after (after advice), or around (around advice) the joint point. In the logging aspect case, the advice is the set of activities to log the main process activity.

In summary, by decomposing a process into a main process and aspect processes, which modularize crosscutting concerns and which affect the main process, the result is a more understandable, reusable, and easier-to-change set of processes. In the next section, we highlight issues arising when relating AO and BPM concepts, to discuss whether the presence of crosscutting concerns requires extensions to conventional BP notations. We have chosen to focus on these issues because:

- They involve the particularly important modeling elements since they constitute the common modularity aspects addressed by notations and frameworks.
- They have recurrently been a hotspot in the existing AO solutions.

Since our context is BPs, we shall be using BPMN as the basic notation in which to discuss the issues of extensions, or not, for using aspects in modularization. A running example will be used in defense of our proposal as we address these issues.

A few other authors have considered AO approaches for dealing with processes. However, as we discuss in the next subsection, in different contexts and goals than our research.

2.4 Related work

Charfi (2007) proposes an AO workflow language, providing constructs typically found in AOP languages to address those problems. He observed that current workflow languages do not address the modularization of concerns which cut across process boundaries, thus the process code of those crosscutting concerns is spread across several workflow process specification and intertwined with the process code addressing other concerns. The workflow logic is encapsulated in a process module and non-functional concerns are encapsulated in aspects modules.

Park *et al.* (2007) affirm that business rules are an important crosscutting concern which should be distinguished from BP instances. They present a rule-based AOP framework, in which business rule aspects contained in BPs can be effectively separated and executed. They introduce a method, through which business rule aspects, separated through an external rule engine, can be represented and evaluated. Although concerned with the implementation of BPs, more specifically with composition, this proposal underlines the business rule perspective as an important modularity issue.

Wada *et al.* (2008) propose an AO language to separate functional and non-functional properties in BPs, in order to preserve the reusability of services. Each aspect specifies non-functional properties which crosscut among multiple services, and is woven to a BP and transformed to application code.

Both, Charfi (2007) and Wada *et al.* (2008) focus on an implementation level, one dealing with Business Process Execution Language (BPEL) and the other with services. Wada's approach is symmetric and uses an AO language, which allows for specifying BP non-functional properties as aspects (which are defined as features) and supports several join points. The solution is not graphic at BPMN level, but, instead, it transforms BPMN into Unified Modeling Language (a special Service-Oriented Architecture profile), and weave the non-functional properties into process models to generate code. In our case, we are proposing an AO strategy, which deals with the process abstraction level targeted by BPMN notations, so much closer to the business architect, and we propose a symmetric approach together with a domain-oriented pointcut language, implemented by an editor, which is an extension of an open-access BPMN editor.

Park *et al.* (2007) propose the use of AO by mixing two levels of abstraction, business rules (more akin to the requirements) and BPs. They present a method to implement an extension of BPEL execution engine to integrate a business rule engine in terms of dynamic weaving and aspect awareness so as to achieve a rule join point model. Our proposal deals with only the BPs level (therefore, BPMN), thus providing a way of dealing with the problems found by and Soffer and Wand (2005).

Correal and Casallas (2007) propose aspect viewpoint, a domain-specific language for software process modeling, based on the viewpoint join point model. It defines the set of valid join points, pointcuts, and advices in a process model. Similarly to our proposal, the join points and pointcuts are defined in terms of the elements of a BPMN process model. However, while our approach is generic, Correal and Casallas's work is particular for the software process domain.

3. The research process

According to Hevner *et al.* (2004), "the design-science paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts." Our process research is inserted in this context, due to the characteristic of

identifying a problem, addressing it through artifact building, and showing its feasibility by a concept proof or example.

To describe our research process, we use a Structured Analysis and Design Technique (SADT) model (Ross, 1977), where the following activities are mentioned:

- review;
- characterize;
- design;
- implement;
- exemplify; and
- verify.

SADT is a notation where boxes represent activities, left arrows represent input required by the activity, down arrows represent controls, up arrows represent mechanisms, and right arrows represents output from the activity. Figure 1 shows our research process. The activities, resources, and outcomes are described as follows, and the terms are used according to the ones of the model in Figure 1.

We started with a mix of Characterize and Review activities, since we have read the literature from the requirements engineering (RE), AO development (AOD), and BPs;

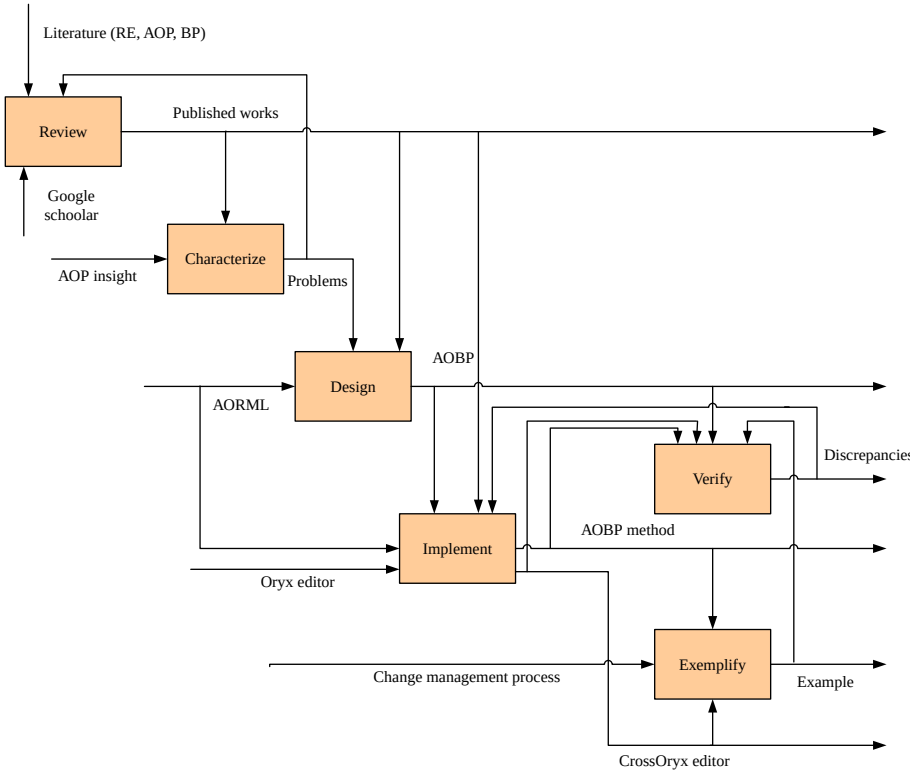


Figure 1.
The research process

and we started with an "AOP insight:" how crosscutting concerns would be related to BPs. As we characterized the "Problems," we provided feedback to the Review activity as more search and selection were conducted as to find the "Published works." Section 2 provides the overall literature review and reflections about the problems found.

To the Review activity, the research method was based on a systematic review (Biolchini *et al.*, 2005; Kitchenham, 2004; Travassos *et al.*, 2008). The goal of this study was finding the publications about representative methodologies, tools, and application examples related to crosscutting issues with respect to BPM and RE. This literature review was done using some keywords (BPM, AO paradigm, process modularity, and AO-BPM). The principal research base used was "Google scholar" which can identify the most relevant results in these research areas. Most of the works found were in software requirements or architecture. Besides keyword research and selection, we accessed proceedings from the International Conference on Advanced Information Systems Engineering and BPM conference. In BPM area, only four works attracted our attention. One of them was used as input in the Design activity.

The Design activity used as input the work of Silva (2006): AO requirements modeling language (AORML) to produce the concept of AOBP. Based on the concept of "AOBP" and on "Published works" and using as input the "AORML" and the "Oryx editor," we implemented the "AOBP method" and the "CrossOryx editor." A "change management process" description was one of the examples used, which was built using both the "AOBP method" as a guide and the "CrossOryx editor" as a support mechanism to produce the "Example." Section 4 presents the "AOBP method" proposed illustrated by the "Example," while Section 5 describes the "CrossOryx editor" implementation.

The Verify activity was used to analyze both the "AOBP method" and the "CrossOryx editor" with respect to the overall design "AOBP" as well as to analyze the resulting "Example" according to the "AOBP method." Eventual "Discrepancies" did provide feedback to the activities of Implement and Exemplify.

4. Analyzing the integration of AO with BPM

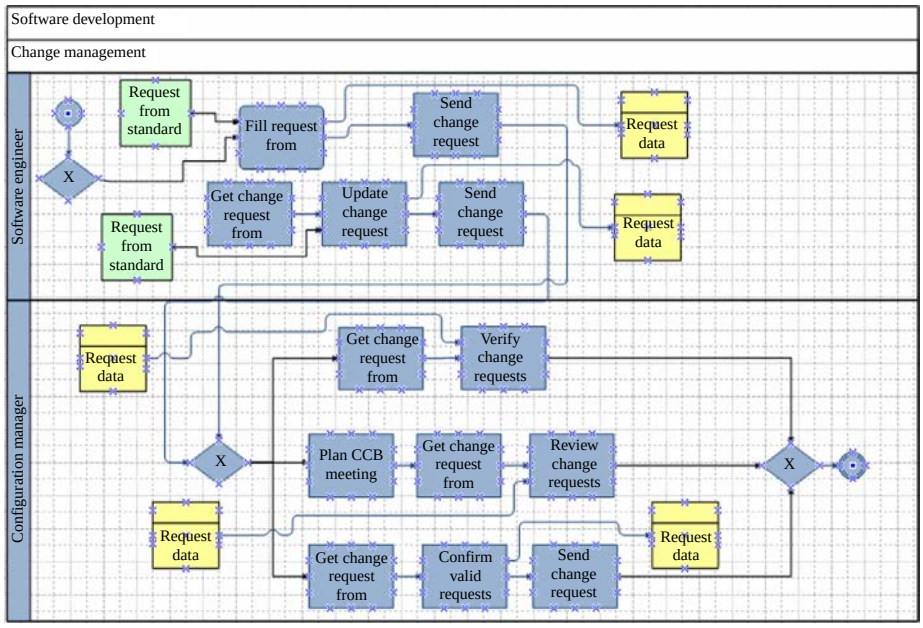
The work of [KKK] uses seven questions to address issues with respect to the applicability of AO concepts in software design; we have reused these questions so as to analyze how the AO concepts could be integrated with BPM. Our analysis is structured into three parts:

(1) with respect to the BP model entities definition which may be characterized by a crosscutting concern;

(2) with respect to the AO concepts application; and

(3) with respect to the representation language used to express AO for BPM.

The analysis is illustrated by a running example presented by [YYY]. Figure 2 introduces the example of a change management process, modeled in Stephen and Miers, 2008 notation. This process exists in a software development project context, being responsible for controlling change requests. It starts when a software engineer needs to perform a new change request or to update an existing one. The first action requires that the engineer sends the change request to the configuration manager. The second action calls for an update on the change request and for the engineer to send it to configuration manager. These forms are received by the configuration manager, who can follow three flows:



Source: Adapted from ([YYY])

Figure 2.
Change management process

(1) check the change request;

(2) plan a configuration control boarding meeting to review the change request; and

(3) confirm valid requests.

For the first one, he needs to get the change request form. For the second, before review change requests, he also needs to get the change request form. And, finally, for the third, in which he confirms valid requests, he needs to get the change request form and, after that, send the change request. As we can see from the description of the process represented in Figure 2, this process has repeated activities and uses the same artifacts at many points of the process.

4.1 Analyzing the BP model entities in the light of crosscutting concerns

The first two questions from [KKK] aim to describe which BP model entities can be modeled by a crosscutting concern, and how these entities are characterized as crosscutting concerns.

The elements of a process model are typically: processes, sub-processes, activities, events, data, actors, and connectors (sequence flow, message flow, and association). In theory, all of them are candidates to be a crosscutting concern. Thus, the crosscutting problem can occur in process, sub-process and activity models, as well as in goal models, organizational models, and groups of roles models.

Some situations suggest that different elements of a process may be modeled as a crosscutting concern:

- An existing or new activity or an event can affect several parts of the process, including connectors.
- Activities appearing in several processes related to the same macro-process.
- Data replicated in the same process or in different processes related to the same macro-process.
- Events replicated in the same process or different processes related to the same macro-process.
- Common goals among different processes related to the same macro-process.
- A role which performs activities in different processes related to the same macro-process.

In general, the impact of crosscutting concerns is observed only in the central elements of processes (activities). However, a general strategy for representing crosscutting concerns in BPM must be wide-ranging, and consider other possibilities. The example depicted in Figure 2 shows four elements which could be identified as crosscutting concerns: activity ("Get change request form" and "Send change request"), information ("Request data"), and document ("Request form standard"). Figure 3 highlights those elements.

Crosscutting concerns can be identified in the context of the same process (intra-process) or among different processes (inter-process). The following concepts characterize crosscutting concerns in a BP model:

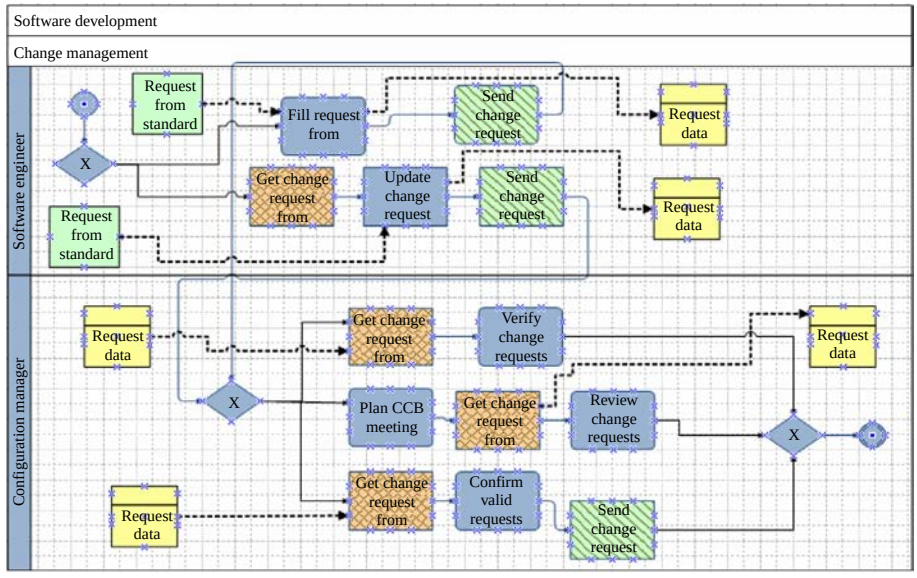


Figure 3.
Change management process crosscutting concerns candidates

- The repetition of activities/events/data in the same process or among different processes. The repetition characterizes that activities/events/data are scattered and/or tangled intra- or inter-process procedures. For example, in Figure 3, "Send change request" and "Get change request form" are repeated activities scattered over the process; the same happens with "Request form standard" and "Request data" which are resources in the process.
- Elements which are not repeated, but are not part of the basic functionality of a process, and may as well occur in another context. For example, elements representing an authentication mechanism acting only at the beginning of a banking application are not repeated, but they are not part of the basic functionality of the process, and may occur in other processes.
- Elements which have the reuse potential, and which are not part of the basic functionality of a process, despite repetition in a specific process model. The authentication mechanism is an example of this situation too.

4.2 The application of AO concepts

Using questions three, four, five and six from [KKK], we analyze the application of AO concepts with respect to four issues: crosscutting representation, crosscutting composition, quantification, and joint points exposure.

4.2.1 Crosscutting representation. Crosscutting concerns scattered and tangled in a BP complex model will hamper process understandability and evolution. To avoid these problems, crosscutting concerns must be represented in a modular form, in order to facilitate process modeling, the interpretation of the business model and the reuse of concerns within and across processes. Several solutions ([KKK]; Sullivan *et al.*, 2005; Krechetov *et al.*, 2006; Baniassad *et al.*, 2006) have been presented in the software engineering literature, and can be adapted to process models. These solutions include the modularization of crosscutting concerns by:

- Representing these concerns through a specific abstraction, the aspect, defined by a specific language.
- Representing these concerns in the same way as done in the multi-dimensional separation of concerns approaches.

These two ways to represent crosscutting concerns are, respectively, classified as asymmetric and symmetric approaches.

The asymmetric strategy provides different abstractions for modeling basic elements and crosscutting concerns. Despite this differentiation seeming to be intuitive, it does not benefit the reuse of concerns. In contrast, the symmetric strategy does not define a specific abstraction to distinguish crosscutting concerns and basic elements. Both are represented using the same abstraction. The difference lies in the way that a crosscutting concern is comprised with the basic elements. Crosscutting elements are composed with basic elements through a specific composition mechanism. For example, the authors ([XXX]) define a symmetric strategy with a crosscutting relationship by defining the composition of a crosscutting concern with a basic element.

We agree that a symmetric strategy is an interesting approach, for it avoids burdening the modeler with a new abstraction and, in fact, the crosscutting relationship may represent the transversal nature of a given concern. In terms of representation,

we propose to represent the crosscutting concern in a specific swimlane. For example, in Figure 3, "Send change request" and "Get change request form" are repeated activities. In Figure 4, they are separated in other swimlanes, orthogonal to the core process swimlane, but represented using the same abstraction as the other activities. The same happens to "Request form standard" and "Request data" elements. The benefits of separating the concerns (core and crosscutting) in orthogonal swimlanes are twofold:

(1) It is a clear representation to illustrate that a crosscutting concern is orthogonal to the core concepts.

(2) It makes the representation of the crosscutting relationships easier and more understandable.

Note that each swinlane represents a crosscutting concern (which could be described by a set of BP model elements). In this example, we have simple crosscutting concerns, each one represented by just one BP model element. Crosscutting concerns are identified at the bottom of the correspondent swinlane. At the top, we have the actor responsible by the aspect execution (similar to the horizontal swimlanes). In this example, it was necessary to create a group called "Software configuration group" to group the software engineer and the configuration manager, to demonstrate that the crosscutting concern can be executed by both.

4.2.2 Crosscutting composition. In BPM, typical connectors are:

- The sequence flow which shows the order of activities in the process.
- The flow of messages showing the messages exchanged between two participants in the process.
- The association, which shows the combination of data, text and artifacts with flow objects.

None of these connectors can be used to represent the crosscutting concern and basic element composition, since each one has a well-defined connection responsibility.

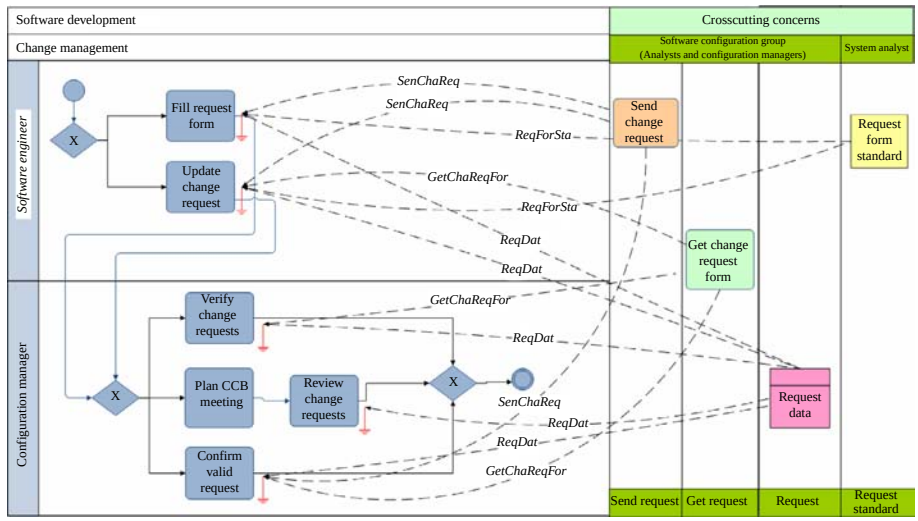


Figure 4.
AO change management process

Therefore, to represent the composition between crosscutting concerns and the elements on the modeling process, we propose a new connector object, the crosscutting relationship, which defines that a crosscutting element (the source) affects another element (the target).

In [XXX], the crosscutting relationship also represents the interaction among a crosscutting concern and the basic process activities. In Figure 4, "Send change request," "Get change request form," "Request form standard" and "Request data" are crosscutting concerns composed with basic process activities through crosscutting relationships named "SendChaReq," "GetChaReqFor," "ReqForSta," and "ReqDat."

4.2.3 Quantification. A quantification mechanism is necessary to avoid several references to each join point explicitly. Usually, a pattern matching expression is used to broaden the coverage of a given pattern. The mechanism allows for expressing quantification in order to reach various join points in a single statement, and, therefore, may include wildcards and logical expressions, such as: if we wish to consider all the expressions in which the suffix is "office," we will quantify it as "*office," and, as such, all symbols ending with "office" will return a positive match. This mechanism should be used in detailing the crosscutting relationship. Since process diagrams (like in BPMN) consist of activities and events, the quantifiers AllActivities and AllEvents are needed to select all set of activities and events. For instance, for detailing the Figure 4 crosscutting relationship, it is necessary to express that "Send change request" acts after AllActivities "Fill request form," "Update change request," and "Confirm valid request."

4.2.4 Join point exposure. As previously mentioned, the process model is composed of several elements, which may be the target of crosscutting concerns. A crosscutting element affects other process model elements through the crosscutting relationship. The question related to this issue is whether the elements affected by crosscutting concern need extra information to allow the connection with the crosscutting concerns. Some related works in the context of AOSD (Aldrich, 2005; Sullivan *et al.*, 2005) about join points exposition criticize the obliviousness property (Kiczales *et al.*, 1997), which suggests the affected elements (core) have no knowledge about the crosscutting elements and, therefore, need not be prepared to receive such information. Sullivan *et al.* (2005) proposes to define interfaces between them. However, BPM does not have the interface concept. Thus, the current elements of the process model are the potential join points, with no need to include additional information. Graphically, the join points can be represented by a ground element, which locates near it, allowing that the source of the crosscutting relationships be the crosscutting element, and the target be the ground element representing the join point. Figure 4 shows the join points with a red ground element. "Send change request" is a crosscutting concern which affects the "Update change request" activity. "SenChaReq" is the label pertaining to the crosscutting relationship between them.

4.3 Representation language

As pointed out in Section 4.2.1, we chose a symmetric strategy, thus we represent aspects using the same concepts as the base description language. However, we must define how the crosscutting concerns will be represented, and how the aspect behave when applied, that is, how it will weave into the core description. So, we first deal with

crosscutting representations, and later, we detail the pointcut language (a pointcut language defines patterns to write pointcut expressions) we have devised for BPM.

The best way of dealing with the crosscutting representation is by means of an example. In our example, "Send change request" and "Get change request form" are crosscutting concerns composed with basic process activities through crosscutting relationships named "SenChaReq" and "GetChaReqFor." Therefore, if a concern is a crosscutting sub-process, it will be still represented as sub-process, and there will be a crosscutting relationship between the crosscutting sub-process and the element affected by it. Similarly, if a data element is repeated in several parts of the model, there should be a crosscutting relationship between the data and the affected element. "Request data" and "Request form standard" are also crosscutting concerns and are composed with a basic process through a crosscutting relationship named "ReqDat" and "ReqForSta." Figure 4 shows the AO version of the process shown in Figure 2. In this figure, we segregate the crosscutting concerns into new orthogonal swimlanes. These concerns are related with basic process activities through a crosscutting relationship. They are inserted in a pointcut represented by a ground element, and the relationships being represented by dotted lines.

In Figure 5, we use the pointcut language to strip out the direct relationships of Figure 4. This is possible, since the pointcut language provides the information to rebuild Figure 4, but presenting the process with a cleaner and textual description. As can be seen, in the orthogonal swimlanes, we have the pointcut language in place of the aspect element, removing the ground element and the dotted lines. This renders the process model clearer and easier to understand, in case crosscutting modularization is well performed.

A pointcut language must be fully expressive to define the different types of join points which appear in a process model. This expressiveness has an impact on the modeling. It is necessary to define join points selectors with significant semantics to represent the different possibilities. The pointcut language represents, textually, the

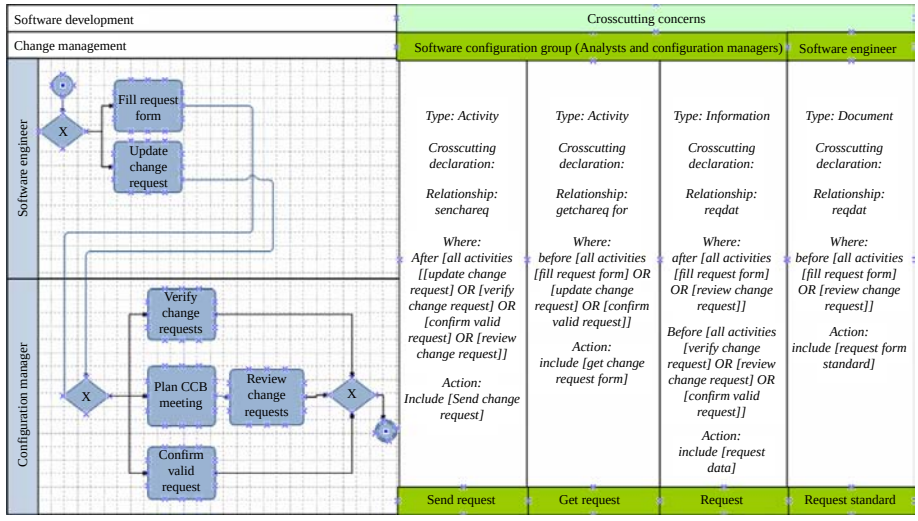


Figure 5.
Textual description of the crosscutting concerns in the change management process

points where the aspect acts, and the moment this is being applied (before, after, and during) at the core description. Therefore, the pointcut language should allow for the combination of join points which, in essence, express the inclusion of crosscutting concerns in a process. The include primitive is the main clause of the pointcut language, used in the advice part to specify the insertion of a crosscutting concern in a core process. In the case of Figures 4 and 5, in order to define the insertion of the "Send change request" crosscutting concern, we must describe:

- the crosscutting concern to be inserted;
- the joinpoints; and
- the crosscutting relationship composing the crosscutting concern with the basic process and the activities to be undertaken.

Two of these descriptions reproduced from Figure 5 as follows:

Aspect: Get Request

Type: Activity

Crosscutting Declaration:

Relationship: GetChaReqFor

Where: Before [all activities [fill change request] or [update change request] or [confirm valid request]]

Action: Include [get change request form]

Aspect: Request Standard

Type: Document

Crosscutting Declaration:

Relationship: ReqDat

Where: Before [all activities [fill request form] or [review change request]]

Action: Include [request document]

The crosscutting relationship called "GetChaReqFor" (Figure 4) defines all activities (AllActivities) "[fill change request] or [update change request] or [confirm valid request]" and provides an advice (Action) that includes the crosscutting concern "Get change request form" before the selected pointcut (Where), the crosscutting concern known as "Get change request form" should be included.

5. The CrossOryx editor

Decker *et al.* (2008) is an open-source web-based BPMN editor. It supports the following elements of an Extended Event-Driven Process Chain diagram: events, functions, connectors (and, or, xor), process interface, data, system, and flow control. It also exports the diagrams to the following formats: pdf, png, rdf, svg, and json.

CrossOryx editor extends Oryx by adding new elements to model the crosscutting concerns and to compose the crosscutting concerns with the core process. Figure 6 shows the initial window of the tool. The elements we have added are highlighted by a dotted red line (Figure 6). CrossOryx is available for download at [ZZZ].

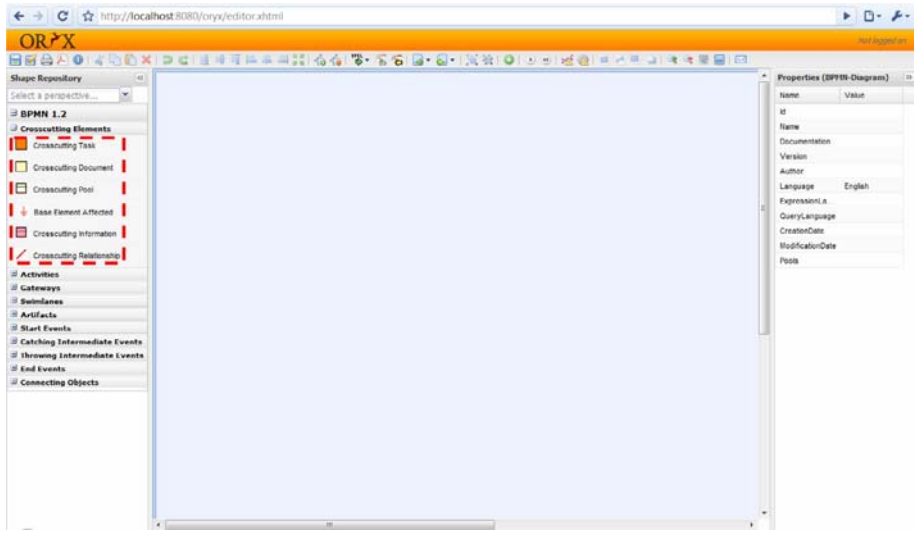


Figure 6.
CrossOryx editor

The new functionalities provided by the CrossOryx editor which supports aspects representation are available in the crosscutting elements menu (Figure 6) and summarized in Table II. We have defined a set of crosscutting elements representing the crosscutting type (task, document, information, etc.) and also crosscutting elements to represent the pool and the crosscutting relationship. The crosscutting type is also defined by the pointcut language.

Crosscutting element	Figure	Description
Crosscutting task		The crosscutting task (aspect) represents a task that cuts across elements of the core process
Crosscutting document		The crosscutting document (aspect) represents a document that cuts across the core process
Crosscutting pool		Crosscutting elements are modeled in a crosscutting pool, in the context of the swimlanes. The top compartment contains the actor responsible for the execution of the crosscutting elements represented in the horizontal swinlane
Base element affected		Base element affected represents in the activities of the core process, the element affected (target) in a crosscutting relationship (crosscutting connection)
Crosscutting information		Crosscutting information (aspect) represents information that cuts across elements of the core process
Crosscutting relationship		The crosscutting relationship represents a crosscutting relationship between elements of the aspectual process and the core process. It links a crosscutting element (source) with the base element (target). It represents an interaction between a crosscutting concern and elements of the core process

Table II.
CrossOryx crosscutting elements

Figure 7 shows an example with a core process and the indication (blue arrow) of the elements affected by the crosscutting concerns. Such elements are represented by the "Base element affected."

After the creation of the core process, a crosscutting pool is created and coupled to the core process pool. Figure 8 shows the pool and the actor or group of actors

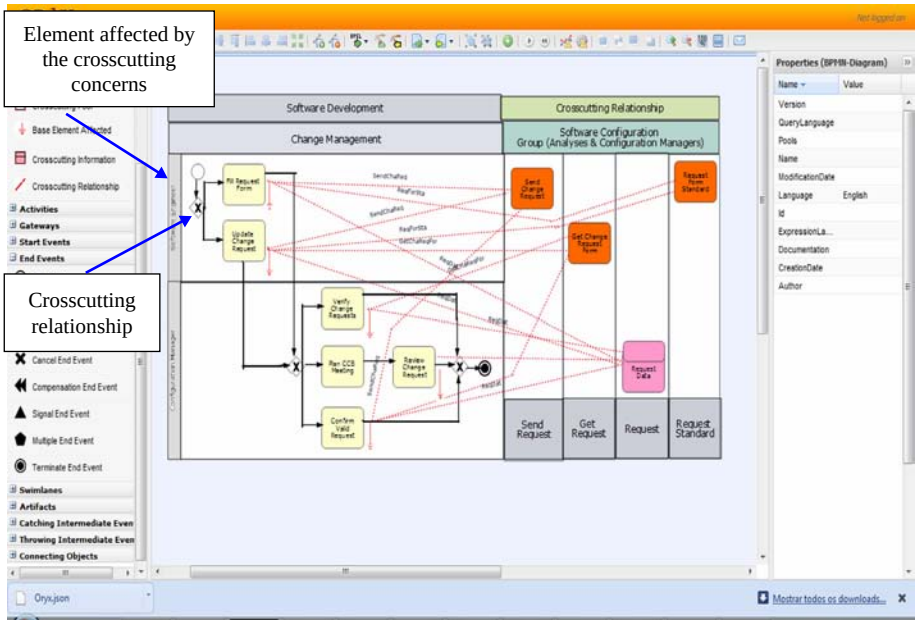


Figure 7.
Representation of the core process and the element affected by the crosscutting concerns

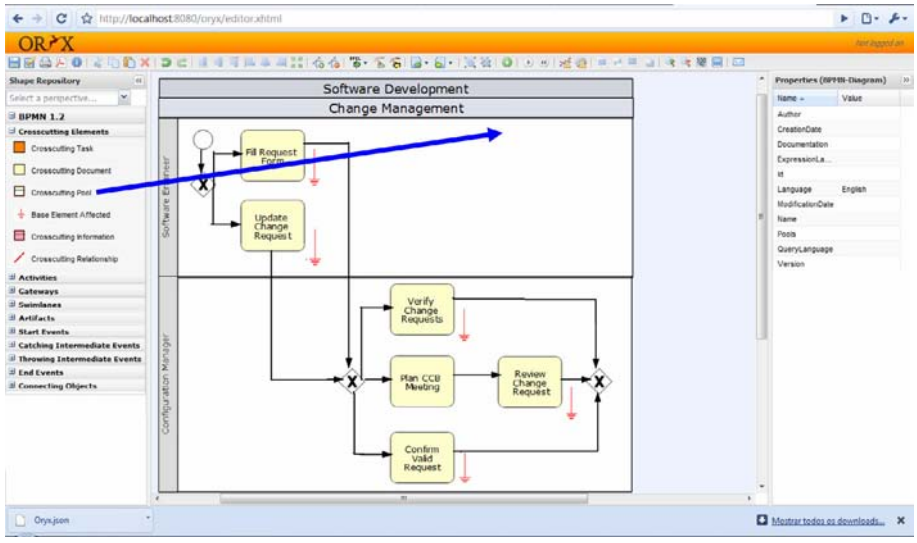


Figure 8.
Crosscutting pool

responsible for the crosscutting concern execution. Software configuration group and software engineer are actors.

The crosscutting elements are created following creation of a crosscutting pool. For instance, Figure 9 contains the following crosscutting tasks: "Send change request," "Get change request form," and "Request form standard." It also contains the "Request data" crosscutting information. These elements are source of crosscutting relationships.

Figure 7 shows some crosscutting relationships, represented by the lines linking the main pool elements and the crosscutting pool elements. "Send change request," "Get change request form," "Request form standard" and the "Request data,", respectively, are source crosscutting elements of the following crosscutting relationship: "SendChaReq," "GetChaReqFor," "ReqForSta," and "ReqDat" that are related with activities that compose the core process.

The implementation of the new modularity construct in Decker *et al.* (2008) was important from different perspectives. First, it has showed that the new construct is implementable in an editor. Second, it makes the dissemination of the new modularity possible. Third, it lessens the burden of using the new concept. Fourth, by automating the edition of models following the new construct, it makes it easier to conduct experiments with the new modularity.

Nonetheless, even with automated support, there is a learning curve on adopting the AO modularity. We are starting to conduct experiments with students and professionals on transforming processes with crosscutting problems into modularized processes. Initial feedback from the use of CrossOryx is positive.

Although the overall idea is not trivial, we believe that a staged approach is working: we start focusing on repeated items and show how process description could be made cleaner without repetition. That is, we are starting by providing processes in which the issue of generality and non-functionality is already modeled. However, we do understand that a major barrier is the education of modelers so as to increase the awareness with respect to non-functional issues (Kueng and Kawalek, 1997).

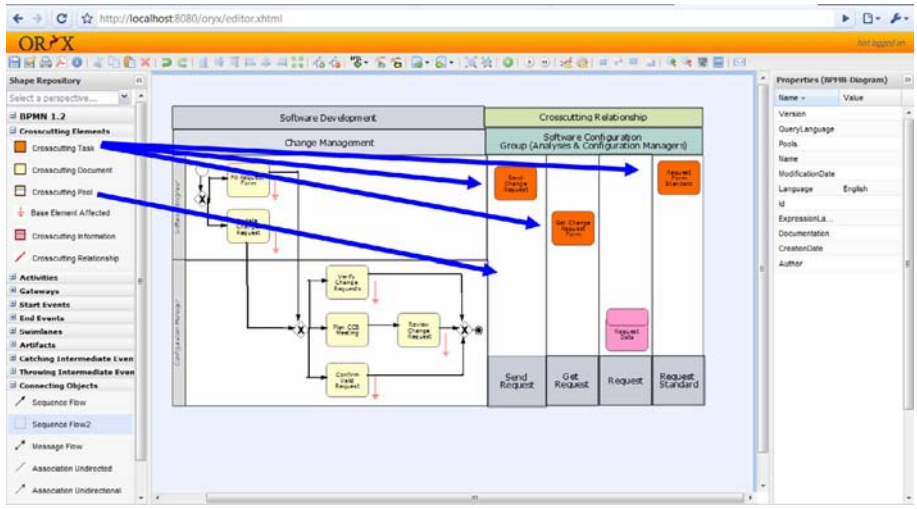


Figure 9.
Representation of the crosscutting element in the crosscutting pool

6. Conclusion

Research and practice point to modeling the different process abstraction levels, which addresses, to a certain degree, the issue of modularity, despite not being enough to directly deal with crosscutting concerns in these models. Moreover, although a few proposals aim to address the problem of crosscutting concerns and the relationships/interaction among the diverse process models and their views, none of them are broad enough to capture all the possibilities or diverse kinds of crosscutting elements. Besides, they are very much focused on process implementation (Wada *et al.*, 2008), or address specific issues, such as business rules (Park *et al.*, 2007), or are domain-oriented (Correal and Casallas, 2007), and do not tackle the representative issue at conceptual level.

6.1 Contributions

In this paper, we further advance our understanding on the role of AO concepts towards a different kind of modularization for BPs. Results so far point to the case of achieving better and more abstract process descriptions. More experimentation is needed to prune the overall strategy as well as to evaluate the gains of adopting AO concepts for BPM.

We have departed from the author's ideas ([XXX]), which was influenced by previous work on requirement engineering (Silva, 2006) and software design ([KKK]). The authors proposed a symmetric approach for dealing with aspects in the context of BPM by means of a graphic and textual language. The graphic language uses BPMN and extends it to allow for the representation of the introduction of the pointcut language.

We have refined the initial work of [XXX] by improving the pointcut language, and its interaction with BPMN, using important issues observed by [KKK] in the context of software ADL ([KKK]). Our contribution is novel to the extent that it is the first to propose and analyze the interaction of AO concepts with BPMN and deal with these issues at a high-level of abstraction. However, as described in Section 2, other researchers have identified the advantages of using AO in the BPM context.

In summary, this paper proposes a set of contributions:

- A proposal on the interplay of AO concepts and BPM in order to enhance modularity on BP models.
- The definition of a graphic and textual representation of the crosscutting concerns at the level.
- The definition of a symmetric and domain-specific pointcut language for defining the pointcuts and advices.
- The provision of an open-source web-based BPMN editor, CrossOryx, which supports the core process modeling, crosscutting elements, and the composition between them.

6.2 Limitations of the proposal

Our results are bound by what we have said in the Introduction: "A crosscutting concern at the BP model level could be any concern which cannot be effectively modularized using the given BPM abstractions," that is, we assume that AO concepts are applied together with other modularization strategies, since just relying on the base heuristic of replication may strip out important process information for its

understanding. Proper care with the intertwining characteristic of abstraction and modularization is not replaced by the application of AO concerns.

Our contribution, as of now, is sustained by the representation scheme we propose and the base components which may be target of crosscutting. In terms of AOSD, we have departed from a factoring-oriented only to activities and tasks, to a factoring considering other description entities besides activities. However, our results, although promising, demand that more work be performed on the nature of the crosscutting as well as on the heuristics of how to combine this modularization strategy with the usual ones proposed by the BPM literature.

6.3 Implications for future research

In terms of implications for future research and practice, based on our experience, AOSD techniques can certainly help organizations to improve their state of practice of BPM. They support modelers with enhanced modular and compositional reasoning, which are imperative throughout all the subsequent software development phases.

Another line of investigation is evaluating the impact of our approach as a basis for deriving AO requirements. We aim to associate our approach with an AO requirements model in order to derive the requirements from an AO business model. The explicit representation of aspects at the BP model must reduce the effort of concern identification and composition at the requirements level. As a consequence, it will benefit all the process of AOSD.

As mentioned in Section 4, we are starting to conduct evaluation experiments. We are using a staged approach together with automated support. CrossOryx is of great help, since it disseminates ideas, for being an open source project, but it also makes it easier to implement experiments. As we gather feedback, we will be improving the editor as well as the supporting documentation for the approach. These steps will be necessary to conduct a number of empirical studies to assess how our AO business approach improves the understandability and reusability of the BP model.

References

Abelson, H., Sussman, G.J. and Sussman, J. (1985), *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, MA.

Abran, P.A., Bourque, P., Dupuis, R. and Moore, J.W. (Eds) (2001), *Guide to the Software Engineering Body of Knowledge – SWEBOK*, IEEE Press, Piscataway, NJ.

Aguilar-Savén, R.S. (2004), "Business process modelling: review and framework", *International Journal of Production Economics*, Vol. 90, pp. 129-49.

Aldrich, J. (2005), "Open modules: modular reasoning about advice", *Proceedings of the European Conference on Object-oriented Programming* (*ECOOP'05*), *Utrecht, July*.

Baniassad, E., Clements, P.C., Araujo, J., Moreira, A., Rashid, A. and Tekinerdogan, B. (2006), "Discovering early aspects", *IEEE Software*, Vol. 23 No. 1, pp. 61-70.

Bertalanffy, L.V. (1969), *General System Theory Foundations, Development, Applications*, George Braziller, New York, NY.

Bhat, J.M. and Deshmukh, N. (2005), "Methods for modeling flexibility in business processes", paper presented at BPMDS Workshop in Conjunction with CAISE, Montpellier.

Biolchini, J., Mian, P.G., Natali, A.C. and Travassos, G.H. (2005), "Systematic review in software engineering", Relatório Técnico ES-679, PESC-UFRJ, Rio de Janeiro.

Bobrik, R., Reichert, M. and Bauer, T. (2007), "View-based process visualization", *5th International Conference BPM, Brisbane*, pp. 88-95.

Charfi, A. (2007), "Aspect-oriented workow languages: AO4BPEL and applications", Dr.-Ing. thesis, der Technischen Universitat Darmstadt, Darmstadt.

Chung, L. and Leite, J.C.S.P. (2009), "On non-functional requirements in software engineering", in Borgida, A., Chaudhri, V., Giorgini, P. and Yu, E. (Eds), *Conceptual Modeling: Foundations and Applications*, 1st ed., Vol. 5600, Springer, Berlin, pp. 363-79.

Chung, L., Nixon, B.A., Yu, E. and Mylopoulos, J. (2000), *Non-functional Requirements in Software Engineering*, Kluwer, Boston, MA.

Correal, D. and Casallas, R. (2007), "Using domain specific languages for software process modeling", paper presented at ACM OOPSLA, Workshop on Domain-Specific Modeling, Portland, OR.

Decker, G., Overdick, H. and Weske, M. (2008), "Oryx – sharing conceptual models on the web", *Proceedings of the 27th International Conference on Conceptual Modeling Lecture Notes In Computer Science*, Vol. 5231.

Filman, R., Elrad, T., Clarke, S. and Aksit, M. (2005), *Aspect-oriented Software Development*, Addison-Wesley, San Francisco, CA.

Hevner, A., March, S., Park, J. and Ram, S. (2004), "Design science in information systems research", *MIS Quarterly*, Vol. 28 No. 1, pp. 75-105.

Jablonski, S. and Goetz, M. (2008), "Perspective oriented business process visualization", *6th International Conference BPM, Milan*, pp. 144-55.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Videira Lopes, C., Loingtier, J.-M. and Irwin, J. (1997), "Aspect-oriented programming", paper presented at European Conference on Object-oriented Programming (ECOOP), LNCS 1241, Springer, Berlin, June.

[KKK], omitted for the blind review.

Kitchenham, B. (2004), "Procedures for performing systematic reviews", Technical Report TR/SE-0401, Keele University, available at: www.idi.ntnu.no/emner/empse/papers/kitchenham_2004.pdf

Kramer, J. (2007), "Is abstraction the key to computing?", *Communication of the ACM*, Vol. 50 No. 4, pp. 36-42.

Krechetov, I., Tekinerdogan, B., Garcia, A., Chavez, C. and Kulesza, U. (2006), "Towards an integrated aspect-oriented modeling approach for software architecture design", paper presented at 8th Workshop on Aspect-oriented Modeling (AOM'06), AOSD'06, Bonn, March.

Kueng, P. and Kawalek, P. (1997), "Goal-based business process models: creation and evaluation", *Business Process Management Journal*, Vol. 3 No. 1, pp. 17-38.

Kueng, P., Kawalek, P. and Bilchler, P. (1996), "How to compose an object-oriented business process model?", in Brinkkemper, S., Lyytinen, K. and Welke, R.J. (Eds), *Method Engineering proceedings of the IFIP WG8.1/WG8.2 Working Conference, Atlanta, GA*.

Lin, F.R., Yang, M.C. and Pai, Y.H. (2002), "A generic structure for business process modeling", *Business Process Management Journal*, Vol. 8 No. 1, pp. 19-41.

Melão, N. and Pidd, M. (2000), "TI: a conceptual framework for understanding business processes and business process modeling", *Information Systems Journal*, Vol. 10 No. 2, pp. 105-29.

Mendling, J. and Strembeck, M. (2008), "Influence factors of understanding business process models", *Proceedings of 11th International Conference on Business Information Systems, BIS 2008*, Lecture Notes in Business Information Processing, Vol. 7, Springer Verlag, Berlin, pp. 142-53.

Ould, M. (2005), *Business Process Management – A Rigorous Approach*, Meghan-Kiffer, Tampa, FL.

Park, C., Choi, H.-J., Lee, D., Kang, S., Cho, H.-K. and Sohn, J.-C. (2007), "Knowledge-based AOP framework for business rule aspects in business process", *ETRI Journal*, Vol. 29 No. 4, pp. 477-88.

Parnas, D.L. (1972), "On the criteria to be used in decomposing systems into modules", *Communication of the ACM*, Vol. 15 No. 12, pp. 1053-8.

Parnas, D.L. (1976), "On the design and development of program families", *IEEE Transactions on Software Engineering*, Vol. 2 No. 1, pp. 1-9.

Prieto-Diaz, R. and Neighbors, J.M. (1986), "Module interconnection languages", *Journal of Systems and Software*, Vol. 6 No. 4, pp. 307-34.

Recker, J., Rosemann, M., Indulska, M. and Green, P. (2009), "Business process modeling: a comparative analysis", *Journal of the Association for Information Systems*, Vol. 10 No. 4, pp. 333-63.

Rosemann, M., Recker, J., Indulska, M.K. and Green, P.F. (2006), "A study of the evolution of the representational capabilities of process modeling grammars", in Dubois, E. and Pohl, K. (Eds), *Advanced Information Systems Engineering, CAiSE – 18th International Conference on Advanced Information Systems Engineering, Luxembourg, June 5-9*, Springer, Berlin, pp. 447-61.

Ross, D.T. (1977), "Structured analysis (SA): a language for communicating ideas", *IEEE Transactions on Software Engineering*, Vol. 3 No. 1, pp. 16-34.

Scott, M.L. (2009), *Programming Language Pragmatics*, 3rd ed., Morgan Kaufmann, San Fransisco, CA.

Silva, L.F. (2006), "An aspect-oriented strategy for requirements modeling", PhD thesis, Computer Science Department, PUC-Rio, Rio de Janeiro, March (in Portuguese).

Soffer, P. and Wand, Y. (2005), "On the notion of soft goals in business process modeling", *Journal of Business Process Management*, Vol. 11 No. 6, pp. 663-79.

Stevens, W.P. (1991), *Software Design: Concepts and Methods*, Prentice-Hall, Hemel Hempstead.

Stephen, A.W. and Miers, D. (2008), *BPMN Modeling and Reference Guide: Understanding and Using BPMN*, Future Strategies, Lighthouse Point.

Sullivan, K., Griswold, W., Song, Y., Cai, Y., Shonle, M., Tewari, N. and Rajan, H. (2005), "Information hiding interfaces for aspect-oriented design", *Proceedings of ESEC/FSE 2005, Lisbon*.

Travassos, G.H., dos Santos, P.S.M., Mian, P.G., Neto, A.C.D. and Biolchini, J. (2008), "An environment to support large scale experimentation in software engineering", *Proceedings of 13th IEEE International Conference on the Engineering of Complex Computer Systems* (*ICECCS*), *Belfast, UK*, pp. 193-202.

van der Aalst, W.M.P., ter Hofstede, A.H.M. and Weske, M. (2003), "Business process management: a survey", *Proceedings of International Conference on Business Process Management*, Lecture Notes in Computer Sciences, Vol. 2678, Springer, Berlin, pp. 1-12.

Vanderfeesten, I., Reijers, H.A., Mendling, J., van der Aalst, W.M.P. and Cardoso, J. (2006), "On a quest for good process models: the cross-connectivity metric", *Proceedings of International Conference on Cooperative Information Systems* (*CoopIS*), LNCS, Vol. 4275, Springer, Berlin, pp. 183-200.

Wada, H., Suzuki, J. and Oba, K. (2008), "Early aspects for non-functional properties in service oriented business processes", *Proceedings of the 2008 IEEE Congress on Services – Part I – Volume 00, July 6-11, 2008*, SERVICES, IEEE Computer Society, Washington, DC, pp. 231-8, available at: http://dx.doi.org/10.1109/SERVICES-1.2008.76

[XXX], omitted for the blind review.
[YYY], omitted for the blind review.
[ZZZ], omitted for the blind review.

About the authors

Claudia Cappelli is an Researcher at the NP2Tec in Federal University of the State Rio de Janeiro (UNIRIO), Brazil. She received her PhD degree in computer science from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil and MSc in Federal University of Rio de Janeiro (UFRJ), Brazil. Her current research interests include process transparency, AO-BPM, BP management, IT architecture, and electronic government.

Flávia Maria Santoro is an Associate Professor at Applied Informatics Department of the Federal University of the State of Rio de Janeiro, Brazil. She received her PhD and MSc degrees in computer science from Federal University of Rio de Janeiro (COPPE-UFRJ). She has experience in computer science, focusing on information systems, acting on the following subjects: BP management, knowledge management, computer-supported cooperative work, and computer-supported collaborative learning. Flávia Maria Santoro is the corresponding author and can be contacted at: flavia.santoro@uniriotec.br

Julio Cesar Sampaio do Prado Leite received his PhD in computer science from University of California, Irvine in 1988. Member of the IFIP 2.9 Working Group. Founding member of the Brazilian Computer Society. Member of the Editorial Board of the *Requirements Engineering Journal*. Keynote speaker (2006) for the Brazilian software engineering symposium. Co-founder of the workshop on RE series. Advisor to 12 PhD dissertations. Author or co-author of 26 journal papers and 127 full conference papers. Member of the IEEE Computer Society and Member of the ACM. Julio Cesar Sampaio do Prado Leite has served as a program committee member for more than a hundred conferences and workshops.

Thais Batista is an Associate Professor at the Computer Science Department of the Federal University of Rio Grande do Norte (UFRN), Brazil. She received her PhD and MSc degrees in computer science from the PUC-Rio, Brazil. Her current research interests include software architecture, AOD, BPM, and distributed systems.

Ana Luisa Medeiros is a PhD student in the Computing Department at Federal University of Rio Grande do Norte since 2008. She completed her undergraduate studies in graduation in technology in Software Development at Federal Institute of Education, Science and Technology of Rio Grande do Norte (IFRN) in 2006 and obtained her master's degree in computer science at Federal University of Rio Grande do Norte (UFRN) in 2008. She has worked for more than two years at NP2TEC (Research and Practice Group in Information technology), participating in BPM projects. Her research interests include AOP, BPM, software product line, metrics, software architecture, RE, and model-driven development. In addition, she is a collaborator of the OPUS Research Group at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) oriented by Alessandro Garcia.

Clarissa S.C. Romeiro is currently a Bachelor's degree student in Information Systems at Federal University of State of Rio de Janeiro. She has worked for more than two years at NP2TEC (Research and Practice Group in Information Technology), participating in BPM projects for Petrobras, the biggest oil company of Brazil. Her areas of interest are BPM and software engineering.