Bus Inf Syst Eng 62(6):561–584 (2020)
https://doi.org/10.1007/s12599-020-00643-2

RESEARCH PAPER

Implementation of Aspect-oriented Business Process Models with Web Services

Hercules Sant Ana da Silva Jose · Claudia Cappelli · Flavia Maria Santoro · Leonardo Guerreiro Azevedo

Received: 29 September 2018 / Accepted: 24 January 2020 / Published online: 17 March 2020

Abstract In software development, crosscutting concerns, such as security, audit, access control, authentication, logging, persistence, transaction, error handling etc. can be modularized using the aspect-oriented paradigm. In business process modeling, aspects have been used to reduce visualization complexity, increase reuse and improve model maintainability. There are techniques which address aspects in modeling and implementation phases of business process; however, these techniques adopt different semantic representations, hindering the integration of these phases into the BPM lifecycle. This work proposes an architecture for service discovery capable of selecting web services that implement crosscutting concerns and meet the goals established in the aspect modeling phase, executing them accordingly with a prioritization. A proof of concept to analyze the proposed architecture and generated artifacts was performed. Afterwards, the proposal was evaluated by means of an experiment. The results suggest that the definition of an operational goal enables the business specialists to concentrate on the modeling of the aspect without necessarily concerning its implementation, since a proper option for implementation is discovered during the execution of the process.

Keywords Business process management · Aspect orientation · Semantics · Automatic discovery of services · Aspect-oriented modeling in business process · WSMO

1 Introduction

The aspect-oriented paradigm (Kiczales et al. 1997) brings several advantages for software development, such as the reduction of scattered codes, transparency of the responsibilities of each module, and independence of modules, as well as facilitating the application evolution with low coupling between the modules, and the code reusability. The business process management (BPM) community has been actively studying the application of this paradigm in all phases of the BPM life cycle in order to modularize crosscutting concerns that are scattered and tangled within process models (Cappelli et al. 2010; Charfi et al. 2010; Jalali 2011; Jalali et al. 2015).

In the process modeling, aspects and their components are represented by visual elements that facilitate the understanding of the model by business experts and stakeholders. Examples are the works of Cappelli et al. (2010) and Charfi et al. (2010) that use the concepts of aspects in the modeling phase, aiming to facilitate the understanding of process models. Yet, in the process implementation phase, the aspects are related to technical

Accepted after two revisions by Jörg Becker.

H. S. Ana da S. Jose (✉)
Federal University of Rio de Janeiro State (UNIRIO), Avenida Pasteur, 458 – Urca, Rio de Janeiro, RJ, Brazil
e-mail: hercules.jose@uniriotec.br

C. Cappelli
Federal University of Rio de Janeiro (UFRJ), Avenida Athos da Silveira Ramos, 274 – Cidade Universitária, Rio de Janeiro, RJ, Brazil
e-mail: claudia.cappelli@gmail.com

F. M. Santoro
University of the State of Rio de Janeiro (UERJ), Rua São Francisco Xavier, 524 – Maracanã, Rio de Janeiro, RJ, Brazil
e-mail: flaviamariasantoro@gmail.com

L. G. Azevedo
IBM Research, Avenida Pasteur 138/146 – Botafogo, Rio de Janeiro, RJ, Brazil
e-mail: lga@br.ibm.com

solutions described in a programming language and software components, which can often be offered as components or services. Charfi and Mezini (2007) address aspects in the implementation phase by specifying new BPEL language elements to incorporate crosscutting concerns in the main process.

The process "*Send articles to reviewers*" (Cappelli et al. 2009) (Fig. 1) exemplifies the ideas of aspects representation in the modeling phase. In this process, the activities related to the logging concept were modularized in the "*Log Information*" aspect. Arrows indicate in which activities the aspect will be inserted during the execution of the process. Figure 2 shows a representation of the "*Log Information*" aspect using the AO4BPEL (Charfi and Mezini 2007), an aspect-oriented extension to BPEL, illustrating the ideas of aspects in the implementation phase. Specific tags are used to represent each element of the AOP paradigm in the BPEL language: tag "*<aspect>*" names the aspect; tag "*<pointcut>*" indicates which step of the process will be invoked (in this case, the "*Send Invitation*" activity); tag "*<advice>*" indicates the event related to the pointcut and where it will be inserted (i.e., *before or after*), the variable declaration that will receive the content resulting from the execution of the activity and which web service will be invoked to record the message log.

Analyzing the two approaches illustrated in Figs. 1 and 2, we notice semantic differences in the representation of aspects and their related elements, making it difficult to provide an integration of the BPM life cycle phases. While the notation used to represent the "*Log Information*" aspect in Fig. 1 aims to facilitate understanding of the model by business experts and stakeholders, the notation used in Fig. 2 is linked to technical solutions described in a programming language and by software components. Elements such as the service name, ports, variables and operations to be called, present in Fig. 2, are not indicated in the notation used in Fig. 1, making it difficult to correlate the elements of each representation.

Bastos et al. (2014) highlighted the problem of representation of aspects in business processes. In order to reduce the gap between the representations, they extended an ontology proposed by van den Berg et al. (2005) to address aspects in the business process domain, in order to allow the documentation and development of aspects in the modeling, configuration and implementation phases. The extended ontology (Fig. 3) defined a representation of the aspect behavior: the process (Process), the code base (BaseCode) and objective (Goal). The authors emphasize the Goal class of the ontology, since the aspect must have an operational objective that expresses the desired behavior of a crosscutting concern within the business process.

The concept of Goal is the same as used by Santos et al. (2011). The use of operational goals as part of the aspect's description during process modeling allows the modeling and code base to be interconnected during the execution of the process. In addition, when setting a goal to be achieved, we have the possibility to choose an implementation for the aspect, without already knowing the details of its implementation. Thus, our research is based on the following hypothesis: *if an operational goal written in a formal language is defined as an aspect in the modeling phase of the process life cycle, then it can be linked to an implementation also written in formal language in order to comply with the intentions of the business experts.*

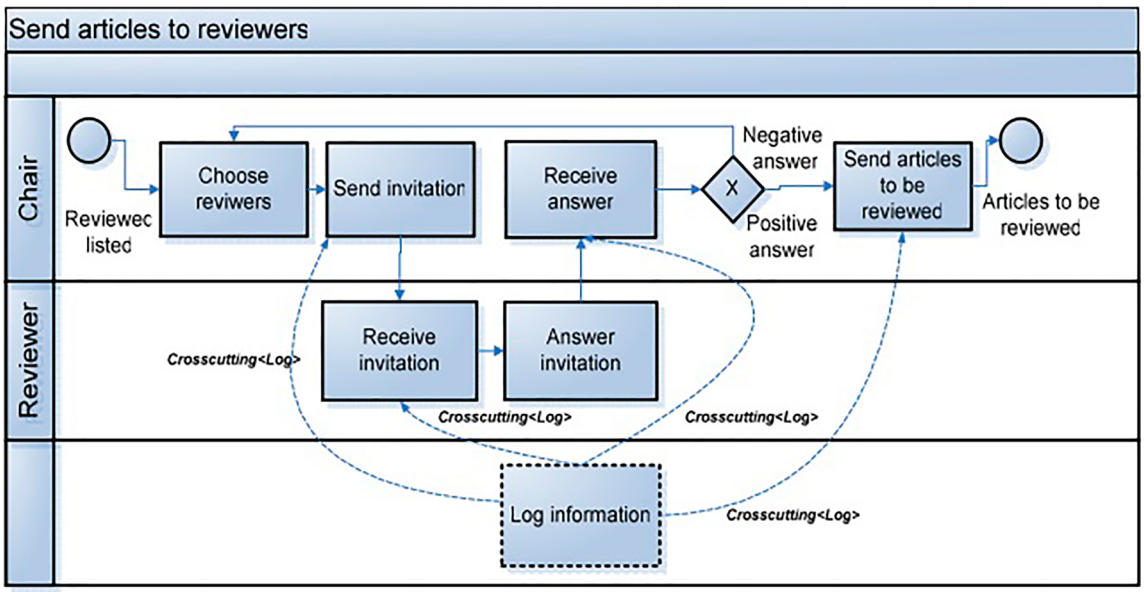


Fig. 1 "*Send articles to reviewers*" process (Cappelli et al. 2009)

Fig. 2 Aspect "*Log information*" adapted to the AO4BPEL language (Charfi and Mezini 2007)

```
<aspect name="Log_Information">
    <partnerLinks>
        <partnerLink name="LogInformationWS" partnerLinkType="LogInformationPLT"
            myRole="caller" partnerRole="log" />
    </partnerLinks>
    <variables>
        <variable name="logRequest" messageType="logInformationInput" />
    </variables>
    <pointcutandadvice>
        <pointcut name="Send Invitation" contextCollection="true">
            //process//invoke[@portType="SendArticleToReviewPT" and
            @operation="sendInvitation"]
        </pointcut>
        <advice type="after">
            <sequence>
                <assign>
                    <copy>
                        <from variable="ThisJPActivity" part="message" />
                        <to variable="logRequest" part="message" />
                    </copy>
                </assign>
                <invoke partnerLink="LogInformationWS"
                    portType="LogInformationPT" operation="registerLog"
                    inputVariable="logRequest" outputVariable="logResponse" />
            </sequence>
        </advice>
    </pointcutandadvice>
</aspect>
```

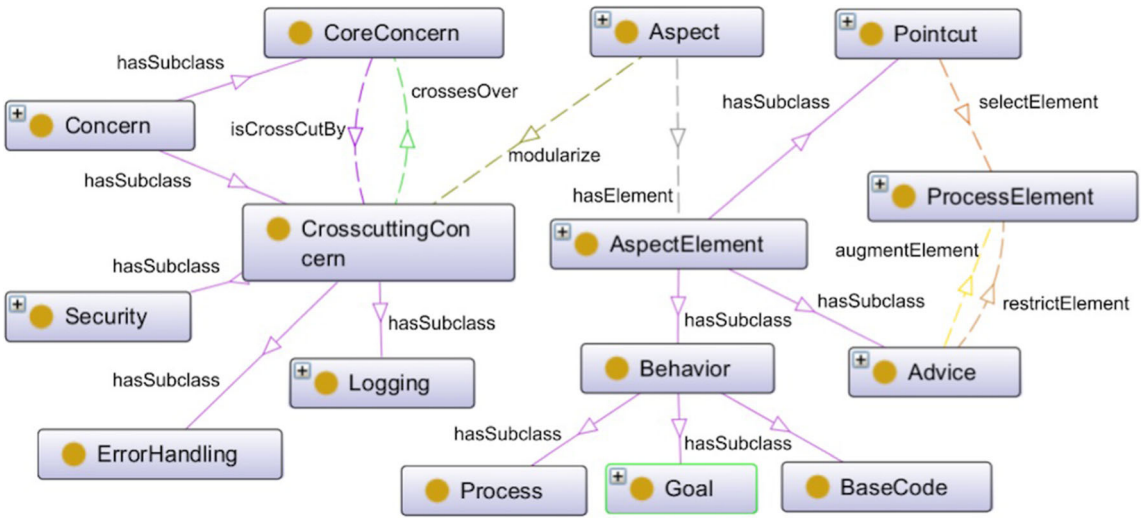


Fig. 3 Ontology base for the aspect-oriented domain in business processes (Bastos et al. 2014)

We presented a preliminary idea of this proposal (José et al. 2016) which was extended by a complete implementation as well as the results from an evaluation. Thus, this paper presents the current implementation of an architecture for service discovery capable of selecting web services that implement crosscutting concerns that support the objectives established in the modeling phase and execute them according to a prioritization. The Goals are described using the formal language Web Service Modeling Language (WSML) containing the desired characteristics of the service to be discovered.

This research was conducted following the Design Science Research approach (Hevner et al. 2004) as detailed in Sect. 2. First, the research problem and hypothesis were defined, extending the work of Bastos et al. (2014) and formalizing the goal concept. Second, a literature review was carried out to obtain the theoretical basis, searching for existing techniques to be used in the solution developed (Sect. 3). The artifact proposed was designed and implemented. The details are presented in Sect. 4. Finally, a proof of concept was performed for the initial evaluation of the solution, followed by an experiment with the objective of evaluating the proposed architecture and artifacts

(Sect. 5). The discussion of results, conclusions and proposals for future work are presented in Sect. 6.

2 Methodological Approach

This research has followed the Design Science Research (DSR) approach (Hevner et al. 2004), which has a dual purposes: (1) generate an artifact to address the problem in a real context; and, (2) conduct empirical research with the application of the artifact in order to generate new knowledge about the phenomenon investigated and answer knowledge questions. Besides, we adopted the method proposed by Peffers et al. (2008), who suggest the following steps to organize the research: (1) Problem Identification; (2) Definition of goals for a Solution; (3) Design and Development; (4) Demonstration; (5) Evaluation; and, (6) Communication (Fig. 4).

Phase 1 is concerned with identifying the problem and providing motivation for the research. As briefly presented in the Introduction and furthered discussed in Sect. 3, literature shows that although aspect-oriented BPM has proved its benefits, it still lacks a practical approach due to difficulties in conciliate modeling and implementation. The research question investigated in this paper is formulated as: "How to integrate the modeling and implementation phases of aspect-oriented business processes management?". We assume (as a hypothesis) that: if an operational goal written in a formal language is defined in an aspect in modeling phase, then it can be linked to an implementation also written in the formal language in order to comply with the intentions of the business experts. In other words, we state that the formal definition of a Goal will guide and connect both the modeling and implementation of aspects in processes. This assumption guided the definition of objectives for the proposed solution (Phase 2).

Accordingly, in Phase 3, in order to solve the research problem stated before, we developed an artifact (the solution) which stands for an architecture for service discovery capable of selecting web services that implement cross-cutting concerns to find the Goals established in the modeling of aspects and execute them according to a prioritization.

Moreover, artifact evaluation is critical for DSR in order to rigorously prove its relevance for practice (Sonnenberg and vom Brocke 2012) as well as to support answering the research question and delivering contributions to the knowledge body about the phenomenon under consideration. Therefore, we first made a Proof of Concept (Phase 4 – Demonstration) which aims at providing evidence that the artifact works according to the requirements defined. Then, we evaluated the artifact in a simulated environment (Phase 5 – Evaluation) through a controlled experiment to answer the research question and verify the viability of its application in a real setting. The details about each phase are described in the next sections.

3 Background Knowledge

To support this work, we have carried out a literature review to identify the most relevant research papers in the BPM area where the aspect orientation concepts were applied. We focused on the approaches applied in the modeling and implementation phases of the BPM life cycle. Some papers such as Cappelli et al. (2010), Charfi et al. (2010), Bastos et al. (2014) and Jalali et al. (2015) motivated us to revisit the previous works and to review the BPM concepts, the AO concepts applied in BPM area and WSMO concepts and technologies, in order to provide a knowledge base for conducting this research. In the subsequent sections, we recapitulate these concepts and relate to the most relevant work in the area.

3.1 Business Process Management (BPM)

Business process is defined as "a set of activities that are carried out in a coordinated manner in a technical and

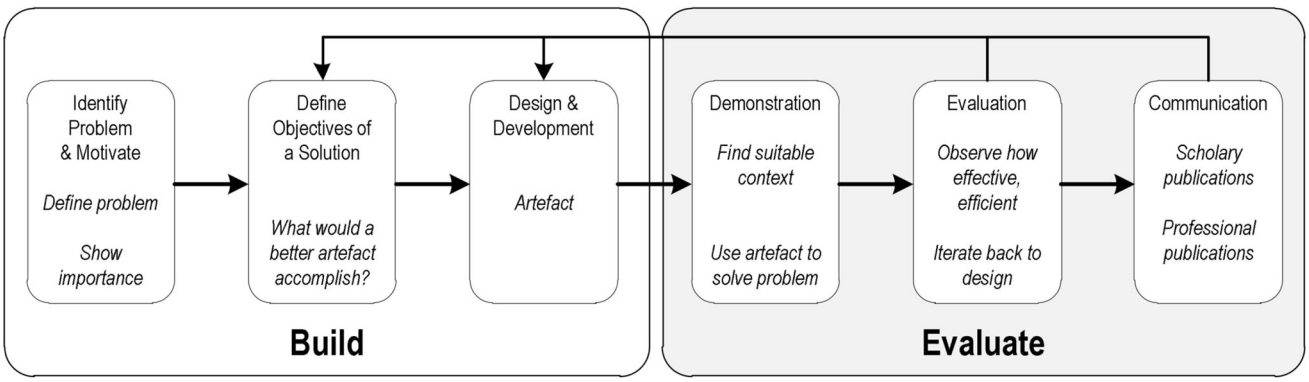


Fig. 4 Steps for a DSR approach (cf. Peffers et al. 2008)

organizational environment. These activities together accomplish a business purpose. Each business process is promoted by a single organization" (Weske 2012). A business process is described by one or more procedures that together achieve a business goal. The execution of a process has well defined start and end conditions, and can combine manual and automated procedures (Hollingsworth and Hampshire 1995).

Business Process Management includes concepts, methods and techniques that support the design, administration, configuration, implementation and analysis of business processes. The basis of BPM is the explicit representation of processes with their activities and constraints. BPM supports analysis, improvement and implementation of processes (Weske 2012).

Research in this area has resulted in various methods, techniques and tools to support the BPM life cycle and its phases, such as design, implementation, management and analysis of operational processes. Figure 5 presents the cycle proposed by Dumas et al. (2013), which is widely adopted in the current BPM literature.

3.2 The Aspect Orientation Paradigm

Separation of concerns, also known as the "divide and conquer" principle, has been a strategy for dealing with software complexity. Two application examples of this principle in the information systems area are: to separate management of data from an application, performed by a DBMS; to separate the main business logic from accessory interests such as security and privacy. The idea of separating concerns and presenting them as individual aspects related to the main concern was applied for the first time in programming and was called aspect-oriented programming (Kiczales et al. 1997).

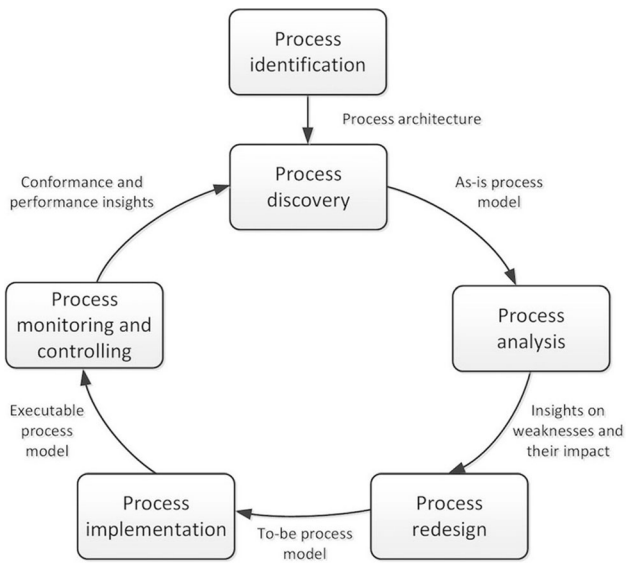


Fig. 5 BPM life cycle (Dumas et al. 2013)

Aspect-Oriented Programming (AOP) is a paradigm that addresses the complexity problem in programming. This problem is dealt with by separating the main concerns of an application from the crosscutting concerns that are usually scattered throughout the application code (Laddad 2003).

The paradigm also solves two common problems of traditional programming approaches: scattering and tangling of software codes (Fig. 6). The first problem refers to implementing the same crosscutting concern within different application functionalities or in other applications. In Fig. 6b we present an example of the dispersed security concern in various software modules. The second problem refers to the application modules containing the code of different concerns. In Fig. 6a a software module implements security interests and transaction control intertwined with business logic.

AOP proposes to implement crosscutting concerns as software modules and join them to the main concern to solve the problems of scattering and tangling. To allow the separation of crosscutting concerns, AOP implements a set of constructs (Charfi and Mezini 2007): *Join points* – are points that enable the addition of one or several crosscutting concerns to the main functionality; *Pointcut* – a language that enables the definition and selection of relevant join points in the running application; *Advice* – a construct that makes it possible to add or change the behavior of the main functionality, which can be performed before, during or after the monitored join points. This construct has the tasks that represent the behavior of crosscutting concerns; *Weaving* – a mechanism to change the static structure of the main functionality by running the advice on each monitored join point and which have been defined by the pointcut language; and *Aspect* – a module containing the advice and pointcuts that will be called before, during or after the join point.

In addition, there are two types of weaving:

- *Static Weaving*: compiles the aspect code along with the code of the main class and produces the application code. The advantage of static weaving is its speed and facility.
- *Dynamic Weaving*: the weaving process is done at runtime. The engine is responsible for orchestrating execution of the main class and aspect at runtime, redirecting the execution flow from the main class to the aspect. The advantage of dynamic weaving is that it provides a dynamic integration of the aspect with the application. Aspects can be added and removed from the application without interrupting program execution.

The AOP acts during the implementation of software systems when looking for code parts where separation of

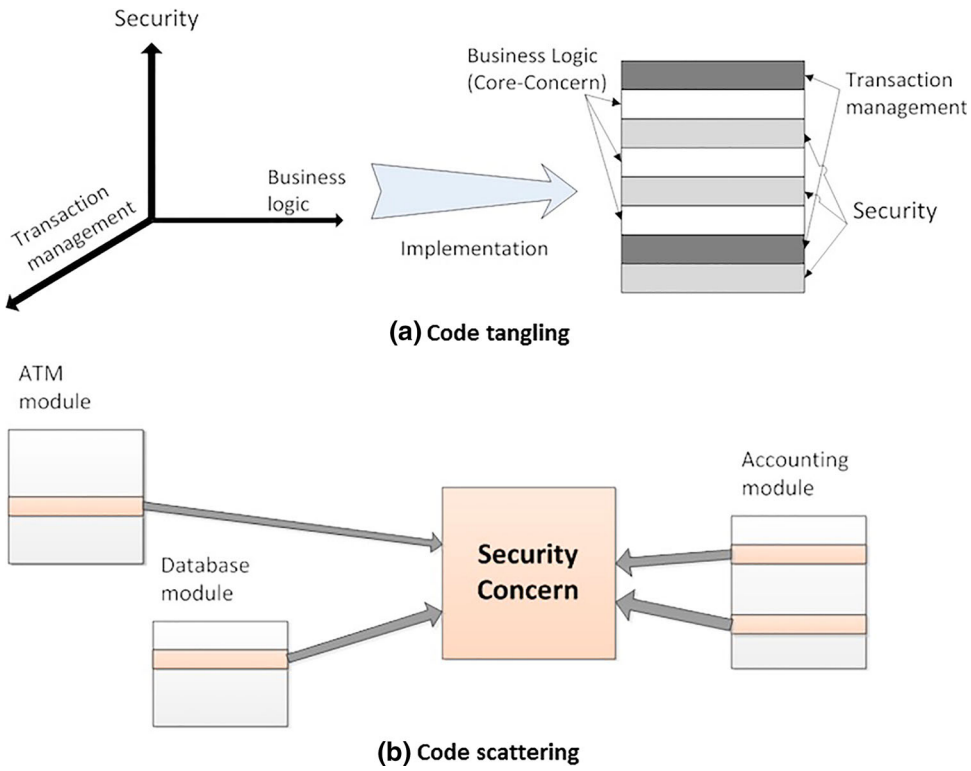


Fig. 6 **a** Tangling code problem; **b** Scattering code problem (Jalali 2011)

concerns could solve the problems of scattering and tangling of software code. Through those set of constructs, the AOP allows for the integration of concerns in various software programs, which facilitates the maintenance and reuse of the functionalities.

3.3 Aspect Orientation Paradigm in BPM

Inspired by AOP ideas, crosscutting concerns have gained visibility in the BPM community. Several studies have been carried out to implement the AOP ideas in the field of business processes in order to deal with the problems of scattering and tangling in the process models (Cappelli et al. 2010; Charfi et al. 2010; Jalali et al. 2015). Scattered and tangled elements can be part of the process (e.g., business rules) or secondary features such as logging and security.

Modularization of a business process by means of sub-processes (i.e., hierarchical structuring or vertical modularization) brings several benefits, such as the possibility to reuse of business process models, to increase maintainability, to allow concurrent development, to enable scalability (e.g., sub-processes running on different engines), and to hide less relevant information (Turetken et al. 2019). On the other hand, modularization increases cognitive effort as readers have to divide their attention between different fragments (Zugal et al. 2015). The use of sub-processes in BPMN may negatively influence the understanding of process models when compared to fully-flattened or vertically modularized using BPMN groups (Turetken et al. 2019). The former benefits are achieved when modularizing process using aspects. Although the latter disadvantage may arise, aspect modularization of business process deals with scattering and tangling elements in the process models, which may be easier to understand when separated from the core of the business process as is the case in aspect-oriented programing (Cappelli et al. 2010).

3.4 Web Service Modeling Ontology (WSMO)

The growing number of Web Services available within an organization presents the challenge to locate appropriate Web Services that meet user expectations (el Bouhissi et al. 2014). Thus, there is a need to use an approach to find the Web Services desired by the user, and also to explore the semantic markup in order to automate discovery, composition, and invocation tasks. Such an approach allows transparent interoperability between tasks and ensures minimal user intervention (Fensel et al. 2011). An adequate approach to perform a Web Service discovery, which has the goal concept presented by Bastos et al. (2014), is the Web Service Modeling Ontology (WSMO) proposed by Fensel et al. (2011).

WSMO describes the relevant aspects related to services that are accessible through a Web Service interface,

allowing the total or partial automation of the tasks (e.g., discovery, composition, mediation, execution, monitoring) involved in intra and inter-company Web Services integration. WSMO's conceptual basis is the Web Service Modeling Framework (WSMF) (Fensel and Bussler 2002) that defines a conceptual model for the development and description of technology-decoupled Web Services with support for ontology, mediation and goal definition by the client. WSMO refines and extends this framework by developing a formal ontology and a formal first-order description language called Web Service Modeling Language (WSML) for the description of WSMO elements.

Following the main features identified in the WSMF, the WSMO identifies four basic elements (Fensel et al. 2011) as main concepts:

- Ontology: defines a common terminology providing concepts and relationships between concepts, describing relevant aspects of the domain, providing formal definitions that can be processed, and thus allowing other components and applications to take into account the meaning of terms in the real world, not just its syntactic meaning.
- Web Service: provides a conceptual model for explicit and unified descriptions of all aspects of a concrete Web Service, including its non-functional properties, its functionality (capability) and interfaces to invoke it, as well as constituting the component capable of achieving a user objective.
- Goal: describes users' interests related to the requested functionality. Using ontologies to define the terminology of the domain, the goals provide the means to specify the objectives of the requesting side when invoking a Web Service, and providing a high-level description of a concrete task to be achieved. Goals model the features and functionality of a concrete Web Service that the user would like to invoke.
- Mediator: describes elements that address interoperability issues between different WSMO elements. A mediator connects WSMO elements in a decoupled form and provides mediation facilities to address issues related to the connection of different elements. It resolves incompatibilities at data, process and protocol level to resolve differences between the terminologies used (data level), the methods of communication between Web Services (protocol level) and the level of combination of Web Services (process level).

WSMO decouples the request for user goals from the services that can fulfill them. The goals should be met by selecting from the available Web Services (described using WSML) the one that best meets the interests of the user (Keller et al. 2004). Goals are symmetrical to semantic Web Services in the sense that the goals describe the desired functionality and the Web Service describes the functionality offered. Therefore, a goal description consists of the same modeling elements as a Web Service description (non-functional properties, capability, and interface).

In addition, an extra non-functional property called the "Type of Match" can be attached to a Goal to represent the desired match type to be achieved between the Goal descriptions and the Web Service descriptions during a discovery process. Keller et al. (2004) identified five types of correspondence to compare the semantic descriptions of capacity between Goals and Web Services. The important feature of these notions is that each one denotes a different logical relationship that has to be maintained in order to consider an adequate service to achieve a particular goal.

Figure 7 shows the five match types: Exact Match – semantic descriptions between Goal and Web Service are the same; PlugIn Match – Goal has all or more semantic descriptions that the Web Service; Subsumption Match – Web Service has all or more semantic descriptions that the Goal; Intersection Match – there are semantic descriptions in common between Goal and Web Service; and Non Match – there is no match between Goal and Web Service.

3.5 Web Service Execution Environment

Semantic Web Services require special execution environments, which control and monitor the exchange of information with their clients while, internally, the concrete Web Services run in their physical location. The process of invoking semantic Web Services needs to be managed and controlled by a Semantic Execution Environment (SEE). The Web Service Execution Environment (WSMX) (Haller et al. 2005) implements this environment from the conceptual model provided by the WSMO, making it a reference implementation. Given a WSMO Goal from a requestor, WSMX invokes functional components for

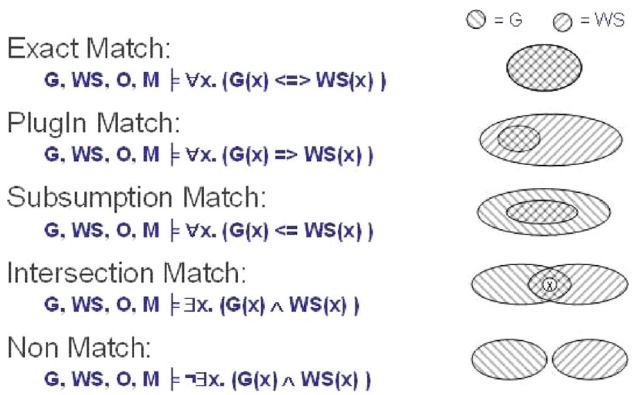


Fig. 7 "Types of Match" attached to a goal

discovery, selection, mediation, composition, and invocation of services in order to resolve the Goal automatically.

The main advantage of the WSMX environment is its event-based component architecture, with formal execution semantics, that enables dynamic invocation of system components as needed to process a specific client goal.

All incoming and outgoing messages are represented in WSML, and these messages are fragments of WSMO ontologies or WSMO entities (Web Service, Goals, Mediators, or Ontologies). WSML is used as an internal data representation of WSMX, and all necessary adaptation operations to and from other representation formats are handled by adapters.

The WSMX architecture follows the fundamental principles of the Service-Oriented Architecture (SOA) (Josuttis 2007), which consist of the integration of heterogeneous components by decoupled functionalities provided by distributed software components, which together can execute a task. Even though WSMX offers standardized implementations for all components of the architecture, standalone components with well-defined functionalities can be connected and disconnected at any time.

3.6 Related Work

Some work has been conducted in recent years to investigate the gap in the representation of aspects between the modeling and implementation phases in the BPM life cycle. However, these papers were directed at identifying new techniques and solutions to apply the concepts of AOP in the modeling of business processes as well as in the service composition.

In Pourshahid et al. (2009), the authors proposed an aspect-oriented framework based on the User Requirement Notation (URN) standard (UTI-T 2008) to apply process improvement patterns. From the information obtained in the process monitoring phase, the proposed framework is able to select the most appropriate redesign patterns among several candidate patterns, considering the impact on process performance and business goals. In order to demonstrate the application of the proposed framework, the authors conducted a case study where process redesign patterns were applied in a healthcare process. The work presented by the authors does not give more details about the implementation of the improved process and the impact of applying redesign patterns on the implementation phase.

Klusch et al. (2008) propose the Model-Driven Service Matchmaker (MDSM) to support human domain specialists and service orchestrators in finding appropriate services in design-time. A service request is modeled in a metamodel called PIM4SWS, where abstract specifications are mapped to specific definitions of sematic web languages OWL-S, WSML and SAWSDL. MDSM automatically transforms this request into semantically equivalent service requests using platform-specific matchmakers, and an orchestrator aggregates and classifies the set of semantic services returned. The returned semantic services contain the necessary information for the orchestrator to be able to invoke the web services. The work does not provide evidence for how the approach can be properly applied in an aspect-oriented context, and the business expert needs to know the details of metalanguage to define which service he desires to invoke.

In Charfi and Zhi (2015), the authors proposed a generic approach to realize crosscutting concerns, mapping BPMN elements to non-functional profiles, and then converting these profiles to AO4BPEL. The authors developed an Eclipse-based tool to support the presented approach and also an execution engine for WS-BPEL. The proposed approach does not provide more details about the definition of aspect in the modeling phase. It partially solves the representation problems between modeling and implementation phase.

Shankardass (2009) proposed an extension called "Aspect wrapper", to the BPMN, in order to encapsulate crosscutting concerns in modules and proposed the "Aspect dot" element to relate these modules to the main process model. According to the position in which "Aspect dot" appear in the activity of the main process, a "Aspect wrapper" can be invoked before, during or after the execution of the activity. However, this approach does not point out any graphical tools to support the definition of aspect-oriented process models. The author proposes the AMAP tool at the conceptual level, and there is no indication in the work that the tool was effectively implemented. Shankardass's work also seeks to combine modeling and implementation of aspects with a tool to map the new elements to BPEL language, besides he suggests an engine where the main process and its aspects would be executed.

Jalali et al. (2015) proposed a systematic approach to support orientation to aspects ranging from business process modeling to process implementation. The authors defined a formal syntax for business process modeling and formally specified operational semantics using Colored Petri Nets for the execution of these processes based on the principle of dynamic weaving of AOP (Jalali et al. 2018). Furthermore, Jalali (2018) describes a formal AO-BPM language including the semantics to support the enactment of models. A hybrid weaving use case drives the proposal in which the process designer should define a configurable aspect-oriented business process model that distinguishes between retroactive and non-retroactive cross-cutting concerns. The proposed solution was implemented in the YAWL environment, and the approach was demonstrated using a case study of bank processes. In contrast to the

proposal presented by the authors, our work focuses on the definition of a semantically described objective that assists to form a union of modeling and implementation, in order to meet the objectives of the business specialists, and also to discover an implementation for the aspects.

Bastos et al. (2014) propose an ontology for the domain of aspect-oriented business processes. The ontology supports the operational goal setting to enable documentation and development of aspects across all phases of the BPM life cycle. The authors performed a proof of concept where they used the WSML language to specify the logging ontology, and provided evidence that it is possible to correlate the objectives defined in the modeling phase with the implementation of services described using WSML. The authors restrict themselves to a proof of concept, and there is no evidence of an implementation of the services and the use of WSMX.

We analyzed each related work to identify which BPM lifecycle it addresses. The results are presented in Table 1.

4 Semantics in the Aspect Implementation in BPM

4.1 Aspect Behavior

According to the ontology proposed by Bastos et al. (2014), the behavior of an aspectualized crosscutting concern can be represented by a "process", "source code" and "goal". The ontology proposed by Bastos et al. (2014) provides a template to document the characteristics of the aspectualized crosscutting concern, thus defining a process model, a Web Service or an abstract objective that will represent the desired behavior for the concern, depending on the phase in the BPM life cycle the concern is working on.

For example, in the modeling phase, we define which process model or abstract goal represents the behavior of the aspect, and in the implementation phase we have the definition of the routine, component or Web Services that represents the behavior of the aspect in that phase (Fig. 8).

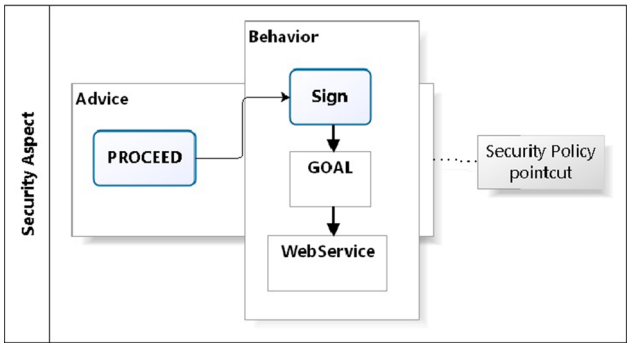


Fig. 8 Representation of the aspect behavior according to Bastos et al. (2014) in the modeling phase

To obtain a better alignment between the modeling and implementation phases, the ways of representing the aspect behavior defined by Bastos et al. (2014) should be used to define the behavior of the aspect, in order to allow: (1) to define the graphical representation of the aspect in the modeling phase; (2) to establish an operational goal to be achieved; and, (3) from the semantic description of the operational goal, to execute the crosscutting concern to discover an implementation that meets the defined goal.

This insight is necessary because of the characteristics of crosscutting concerns that represent non-functional requirements. In the process model, these concerns are usually represented by few activities describing their tasks, because the details of these steps are encapsulated in routines, modules or remote Web Services, whether they are provided by a BPMS or made available via remote services or components. In this way, the aspect behavior is linked with its respective implementation in the implementation phase.

Thus, the *Aspect* has a *Behavior*, and this behavior is composed of *Process*, *Base Code* and *Goal*, as shown in Fig. 9a. We define the elements of the aspect behavior as follows:

- Process: a graphical notation of process modeling (e.g., BPMN, Petri-Net);

Table 1 Comparison of related work

Approaches	BPM phase where approach was applied	Approach based on
Pourshahid et al. (2009)	Redesign	URN standard
Klusch et al. (2008)	Redesign and Implementation	Model-driven service matchmaker
Charfi and Zhi (2015)	Redesign and Implementation	BPMN and AO4BPEL
Shankardass (2009)	Redesign and Implementation	BPMN and BPEL
Jalali et al. (2015, 2018), Jalali (2018)	Redesign and Implementation	BPM and AOP concepts
Bastos et al. (2014)	Redesign	AO-BPM ontology (van den Berg et al. 2005)
Our approach	Redesign and Implementation	AO-BPM (Jalali et al. 2015)

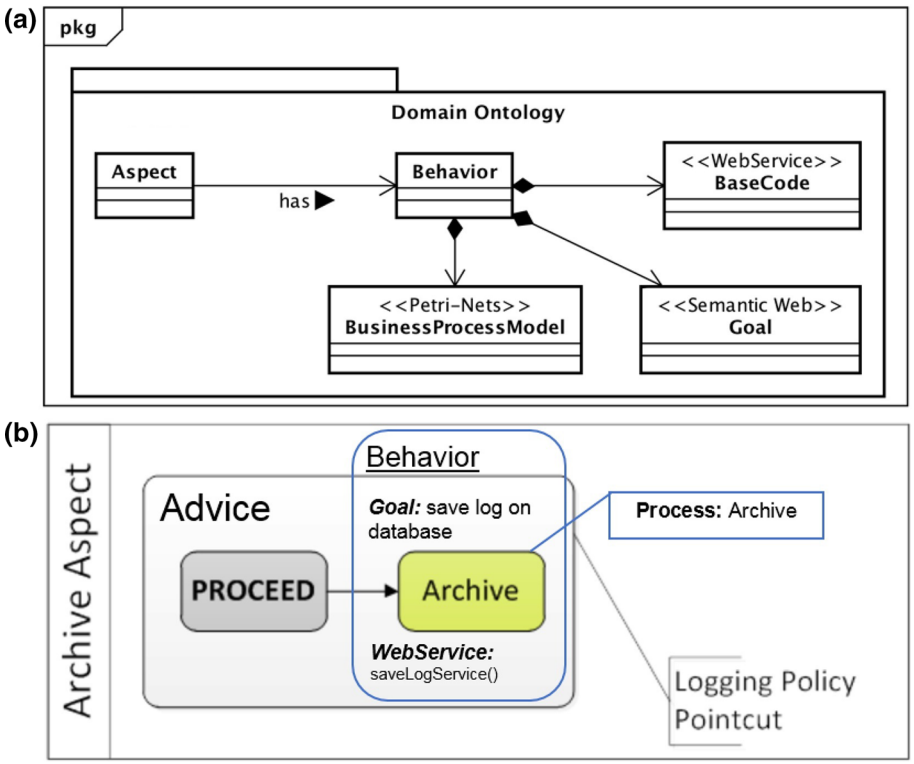


Fig. 9 Representation of aspect behavior

- Base code: functionality (component or Web Service) to be executed in the execution phase of the process, written in a programming language (e.g., Java, C#);
- Goal: abstract definition of an objective expressing the desired behavior of the aspect within the business process, written using a semantic language.

Figure 9b illustrates these elements by means of the Archive Aspect. The "*Archive*" activity represents the "*Process*" element of the behavior. We also have the operational goal "*Save log on database*", indicating the objective to be achieved when the activity is performed, and finally the indication of which concrete Web Service method will be invoked, in this case it will be the "*saveLogService()*" method. The "*PROCEED*" activity indicates that the aspect will be performed after the main process activity (Jalali et al. 2015), and "*Logging Policy Pointcut*" identifies the rule that will be applied to the joint point.

4.2 Operational Goal

Bastos et al. (2014) used the concept of Goal to represent an operational objective that expresses the desired behavior for the crosscutting concern within the business process. The adoption of this concept was inspired by the work of Santos et al. (2011) where operational objectives were used to identify aspects.

The idea to associate goals with business processes comes from the fact that processes exist to satisfy an organization's goal. Soffer and Wand (2004) present some definitions to relate the concept of operational objectives to process models. Santos et al. (2011) and Soffer and Wand (2004) emphasize that the idea of "goal" is related to an operational objective of the process only, as opposed to the organization's business objectives. In other words, a goal is to be achieved by the process.

To relate the concept of objectives to processes, Soffer and Wand (2004) present the following definitions: a goal is a set of stable states that one wishes to achieve. The objective of the process means each execution of the process allows the goal to be reached, in other words, to achieve one of these stable states. To relate the established objective with process design, the concept of goal is operationalized through the criterion function, which maps the values of state variables in the domain, allowing to conclude whether the process has reached its objective or not.

When we have crosscutting concerns modularized in aspects, we have an aspectualized process with an operational objective to be fulfilled. We can say that a process was successfully completed when it achieves the goal assigned to it. According to our proposal, achieving the goal means finding an implementation which performs the task that it has been assigned to by the business expert through its operational goal.

4.3 Operational Goal Formalization in AO-BPM

The purpose of defining a formal syntax for aspects in business processes has the following advantages. First, it provides a solid foundation for developing a concrete syntax for BPM aspect modeling, which may be an extension of a notation of existing processes such as BPMN (OMG 2011) or YAWL (van der Aalst and ter Hofstede 2005). Second, it serves as a standard reference for the development of operational semantics for aspect-oriented business processes. Finally, it is an essential step in defining a formal semantics to support design-time analysis in order to verify its integrity.

We based our formalization on the proposal of Jalali et al. (2015), extending some of definitions of abstract AO-BPM syntax as follows: (1) we defined the concept of operational goal and incorporated it into the advice's definition (Definition 3); (2) we included a set of operational goals assigned to an advice in the aspect definition (Definition 4). In this way, we allow the AO-BPM to be extended to other process modeling languages with the concept of operational goal incorporated in it.

Definition 1 (*Business process model*) A business process model P can be defined as a tuple (T, C, C_I, C_E, X, F, L, D, R, *Name*, *Dat*, *Act*) where:

- T is a set of tasks.
- C is a set of conditions.
- $C_I \subset C$ is a set of initial conditions.
- $C_E \subset C$ is a set of end conditions.
- X is a set of routing constructs.
- $F \subseteq (C \backslash C_E(T \cup X)) \cup ((T \cup X) \times C \backslash C_I) \cup ((T \cup X) \times (T \cup X))$ is the flow relation, such that every node in the graph $(T \cup C \cup X, F)$ is on a direct path from an initial condition $i \in \mathrm{C_I}$ to an end condition $i \in \mathrm{C_E}$.
- L is a set of task labels.
- D is a set of data objects.
- R is a set of roles (human resource).
- *Name*: $T \rightarrow L$ assigns a label to a task.
- *Dat*: $T \nrightarrow D$ associates a data object with a task.
- *Act*: $T \nrightarrow 2^R$ designates one or multiple roles to a task (that requires user interaction).

An *advice* is a special business process that contains at least one `PROCEED` task. There are three types of advice processes, which capture the "before", "after" and "around" advice.

Definition 2 (*Advice*) An advice P^a can be specified by a business process model (T, C, C_I, C_E, X, F, L, D, R, *Name*, *Dat*, *Act*) in which $\exists_{t \in T}$ *Name(t)* = 'PROCEED' (where 'PROCEED' $\in L$ is a reserved label for `PROCEED` tasks).

Let P^A be a set of advice processes and AN a set of advice names, *Advice*: $P^A \rightarrow AN$ assigns to each advice process an advice name, and *Type*: $P^A \rightarrow$ {'*before*', '*after*', '*around*'} specifies when an advice process can occur given the associated join point.

According to our proposal, operational goals are associated with advice, indicating the desired behavior for those advice at runtime.

Definition 3 (*Goal*) Let G be a set of operational goals that can be assigned to an advice P^a ($G \rightarrow P^a$) such that $\exists_{g \in G} \nrightarrow P^a$.

An aspect specification comprises a number of advice processes belonging to the same aspect and a set of rules (pointcut) specifying when these advice are expected to be used by the main business processes.

Definition 4 (*Aspect specification*) Let P^A be a set of advice process, Q a set of business process, and T_P the set of tasks in each P ∈ Q, an aspect specification S can be defined as a tuple (A, *Pointcut, G*) where:

- $A \in 2^G$ represents one or more operational goals.
- $A \in 2^{(P^A)}$ is a number of advice processes.
- Pointcut: $(\bigcup_{P \in Q} \mathrm{T_P}) \times \mathit{BoolExpr} \nrightarrow A$ defines a set of predicates that relate a specific task (which is called and advised join point), under a given condition (from the set of Boolean expressions *BoolExpr*), to the corresponding advice process in A.

Let $\mathbb{S}$ be a set of aspect specification and SN a set of aspect names, *Aspect*: $\mathbb{S} \rightarrow SN$ assigns to each aspect specification an aspect name.

An aspect-oriented business process model is composed of a main process and a number of relevant aspects. The main process contains advised join point tasks, which are associated with the relevant advice processes according to specific pointcut rules and has no `PROCEED` task.

Definition 5 (*Aspect-oriented business process model*) An aspect-oriented business process W is a tuple (P, T_A, $\mathbb{S}$) where:

- P is a business process model that captures the core concern, where given the set of tasks T of process P, $\neg\exists t \in T$ such that *Name(t)* = 'PROCEED'.
- $T_A \subseteq T$ is the set of advised join points.
- $\mathbb{S}$ is a set of aspect specifications.
- $\forall\ t \in T_A$, $\exists S \in \mathbb{S}$ e $\exists c \in \mathit{BoolExpr}$ such that $(t,c) \in \mathit{dom(Pointcut)}$.

4.4 Architecture and Artifacts

In the previous sections, we defined the concept of the aspect behavior in order to narrow the gap between modeling and implementation using an operational objective. We formally conceptualized the operational goal for the aspect using the abstract syntax of AO-BPM. Aiming to demonstrate the applicability of the operational goal in the aspects and how it narrows the gap between modeling and implementation, we propose an architecture involving the redesign and implementation phases in the BPM life cycle (Dumas et al. 2013).

The architecture makes use of two environments that act in each phase of the BPM life cycle: (1) the Business Process Management System (BPMS), which acts in the redesign phase, allowing: the modeling of the main process; the modeling of its aspects; and the definition of the operational goals; and, (2) the Semantic Execution Environment (SEE), which acts in the implementation phase and which is responsible for the discovery and invocation of semantic web services that can meet the goals defined for the aspects (Fig. 10).

We chose the YAWL system as BPMS because it is an extensible, open source environment and allows the inclusion of new components to interact with the engine. The aspect support in the YAWL system was made using the approach proposed by Jalali et al. (2015), enabling the support of dynamic weaving and thus giving the necessary flexibility to the discovery of the aspects' implementation.

The static weaving is not appropriate because the goals defined by the business experts may change, forcing each change in the goal to make a change in the main process and its aspects. The WSMX was the best choice as SEE environment because it implements the WSMO conceptual model and supports discovery and invocation of semantic web services written in the WSML language.

In addition to the two mentioned environments, the proposed architecture also includes two components to help establish the link between modeling and implementation of aspects. The first one is a Rule Editor that is responsible for linking operational goals to their respective advice. Each goal defined by the business expert has an identifier, which is assigned to each advice. The rules are stored in the rule repository, where they can be queried every time an aspect is executed. The Rule Editor is an auxiliary utility and it can be an internal component, a BPMS module, or an external component. For this work we decided to use the Pointcut Editor (Jalali et al. 2013) as the Rule Editor, since it already includes the definition of aspects, advice and pointcuts. We extended the functionality of the artifact to allow the definition of operational goals.

The second component is responsible for performing the process of discovery and invocation of the concrete Web Services that implements the desired crosscutting concern. This component acts during the execution of the aspect, and it is responsible for starting the set of activities that compose the aspect. This responsibility is delegated to the component by the BPMS engine. From the information obtained in the rule repository, the component uses the SEE discovery and invocation interfaces to find out which concrete Web Service to invoke. The component provides the identifier of the goal which is to be discovered by the SEE, and performs the discovery process from the information contained in the goal. Once a semantic web service is found that meets the stated goal, the component sends its identifier through the invocation interface. The component waits for the SEE response containing details about the invocation of the concrete Web Service. The custom component acts within the BPMS engine because it needs access to the data provided from the aspect in order to route them to the sematic web service found, and thus send them to the concrete Web Service that it will be invoked to answer the goal.

The goals created by business experts are written using WSML. These goals are stored in a specific repository that is read by the SEE during the discovery process. Concrete Web Services that implement crosscutting concerns also receive a semantic description in WSML to be able to match with one or more goals, and allow its invocation by the SEE. The developer describes the concrete Web Services semantically and stores the descriptions in a specific SEE repository.

In Fig. 11 we have an example of a semantic definition of a concrete Web Service, written in WSML, that sends log messages by email. A developer defines the following sections in the semantic web service: nonfunctional requirements (*nfp* and *endnfp* blocks); *importsOntology*, indicating which ontology will be used (in the example "*LogOntology*" is used); a capability section defining preferred matching strategies in the nonfunctional section of the capability; and pre and post conditions to invoke the concrete Web Service. In the preconditions' section, the developer defines variables and values that are required for the invocation of a concrete Web Service, and in the postconditions section, he defines the required information the response should have to indicate the invocation was valid. Each variable has its respective representation in the ontology used by the semantic Web Service.

The *interface* section contains the required details for the SEE to interact with the concrete Web Service. In order for this interaction to occur, the developer defines three sections: (1) the *choreography* section, which tells the SEE how to interact with a concrete Web Service; (2) the *stateSignature* section, which defines details about the

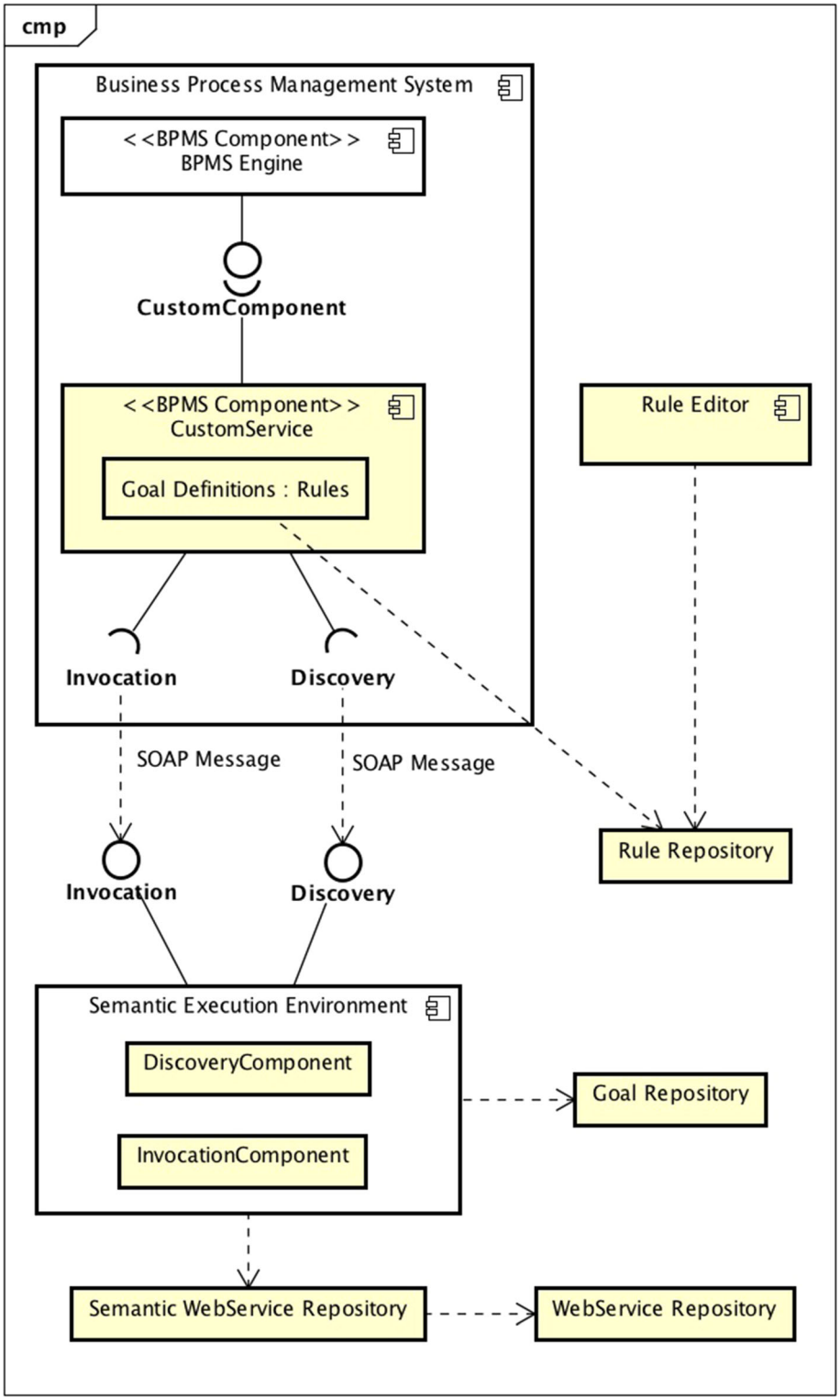


Fig. 10 Interaction schema between the BPMS and the SEE through the customized component

communication with the concrete Web Service, such as which ontology will be used (*importsOntology*), which concepts of the ontology represent the request object (*in*) and the response (*out*), and their respective methods; (3) and the *transitionRules* section that specifies how objects sent and received from the concrete Web Service will be handled. In the example in Fig. 11, every time a request object is a "*LogRequest*" type, the response object will be a "*LogEffect*" type.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace { _"http://www.uniriotec.br/aspect#",
    onto _"http://www.uniriotec.br/aspect#",
    wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
    dc _"http://purl.org/dc/elements/1.1#",
    discovery _"http://wiki.wsmx.org/index.php?title=DiscoveryOntology#"
 }

webService MailLog
    nfp
        dc#title hasValue "Mail Log service"
        dc#contributor hasValue "Hercules S. S. Jose"
        dc#description hasValue "Send the information logged by e-mail"
        _"http://owner" hasValue _"http://www.uniriotec.br/aspect/logging"
    endnfp
    importsOntology {
        onto#LogOntology
    }
    capability MailLogCapability
        nfp
            discovery#discoveryStrategy hasValue discovery#LightweightDiscovery
            discovery#discoveryStrategy hasValue discovery#NoPreFilter
        endnfp

        postcondition
            definedBy
                ?logEffect memberOf onto#LogEffect
                    and ?logEffect[messageLogged hasValue _boolean("true")]
                    and ?logEffect[mailSent hasValue _boolean("true")].

interface MailLogGoalInterface
    choreography MailLogGoalChoreography
    stateSignature MailLogGoalStateSignature
        importsOntology onto#LogOntology
        in
            onto#LogRequest withGrounding _"http://localhost:8080/LogApp/service
        out
            onto#LogEffect withGrounding _"http://localhost:8080/LogApp/services

    transitionRules MailLogGoalTransitionRules
        forall {?request} with (?request memberOf onto#LogRequest) do
            add (_# memberOf onto#LogEffect)
        endForall
```

Fig. 11 Example of semantic web service defined by a developer

Figure 12 shows an example of operation goal definition using WSML. The business specialist defines the nonfunctional requirements section in a similar way to how the developer proceeds with semantic Web Services. He imports the WSMO ontology and defines the section capability, stating which matching strategy will be effectively used (in the semantic Web Service, it is only a suggestion), and what pre and post conditions the desired semantic Web Service should have. The matching strategy chosen in the goal will compare the definitions in the capability section to identify the possible semantic Web services that meet the pre and post conditions defined in the goal.

Finally, Fig. 13 shows the execution process of the proposed architecture where the focus is to show the roles in the modeling and implementation phases of the main process and its aspects, and later the execution in a BPMS. The process shows the implementation of aspect features in a programming language and subsequent publication via web service. It is also demonstrated which the steps of the semantic descriptions of the published web services and goals are and the interaction of the aspect with the semantic execution environment (SEE), showing the verification of which web service meets the goal defined in the aspect and the invocation of the found web service.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace { _"http://www.uniriotec.br/aspect#",
    onto _"http://www.uniriotec.br/aspect#",
    wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
    dc _"http://purl.org/dc/elements/1.1#",
    discovery _"http://wiki.wsmx.org/index.php?title=DiscoveryOntology#"
 }

goal MailLogGoal
    nfp
        dc#title hasValue "Mail log goal"
        dc#contributor hasValue "Hercules S. S. Jose"
        dc#description hasValue "Achieve the goal of send the log by e-mail"
    endnfp
    importsOntology {
        onto#LogOntology
    }
    capability MailLogGoalCapability
        nfp
            discovery#discoveryStrategy hasValue discovery#LightweightDiscovery
            discovery#discoveryStrategy hasValue discovery#NoPreFilter
        endnfp

        postcondition
            definedBy
                ?logEffect memberOf onto#LogEffect
                    and ?logEffect[messageLogged hasValue _boolean("true")]
                    and ?logEffect[mailSent hasValue _boolean("true")].
```

Fig. 12 Example of operational goal defined by a business expert

5 Proposal Evaluation

Following the phases of the Design Science Research (DSR) approach (Hevner et al. 2004), in this section, we performed the Phase 4 (Demonstration) and Phase 5 (Evaluation). For the demonstration phase, we conducted a proof of concept where we applied the proposed architecture and artifacts produced to an example business process available in the literature, in order to select an implementation for the aspect from an operation goal. For the evaluation, we performed an experiment simulating the selection and discovery of semantic web services from operational goals defined on the aspect by business experts.

5.1 First Step: Proof of Concept

A proof of concept (Carsten 2006) was applied to our study in order to: (1) validate the technical feasibility of the proposed architecture and artifacts produced[1]; (2) identify potential technical barriers and opportunities when using the YAWL environment in conjunction with the semantic execution environment; (3) demonstrate in a practical way the issue of implementation flexibility for the aspect evidenced in the proof of concept elaborated by Bastos et al. (2014); (4) investigate if the proposed approach is able to select an appropriate service to implement the aspect according to the goal defined in the modeling phase.

The proposed architecture was applied in an example scenario. The scenario was an adaptation of the "*Send Articles to Review*" process described by Cappelli et al. (2009), comprising the "*Log Information*" aspect (Fig. 14). The chosen process deals with the selection and invitation of article reviewers by the conference chair. In the following paragraphs, the execution of the proof of concept is presented, comprising: process and aspect modeling using the YAWL editor; the implementation of the web services representing the "*Log Information*" aspect; the use of tools to create the ontology, semantic web services and goals; and, the use of the WSMX semantic execution environment tool.

The main process and aspect were modeled in the YAWL editor. To allow the aspect to be incorporated into the main process, we configured the *AspectService* artifact (Jalali et al. 2015) in the YAWL environment, thus enabling support for dynamic weaving. We used the concepts presented by Jalali et al. (2015), where a special task called PROCEED must be entered into the advice,

[1] All artifacts produced for the proof of concept are available at https://github.com/herculeshssj.

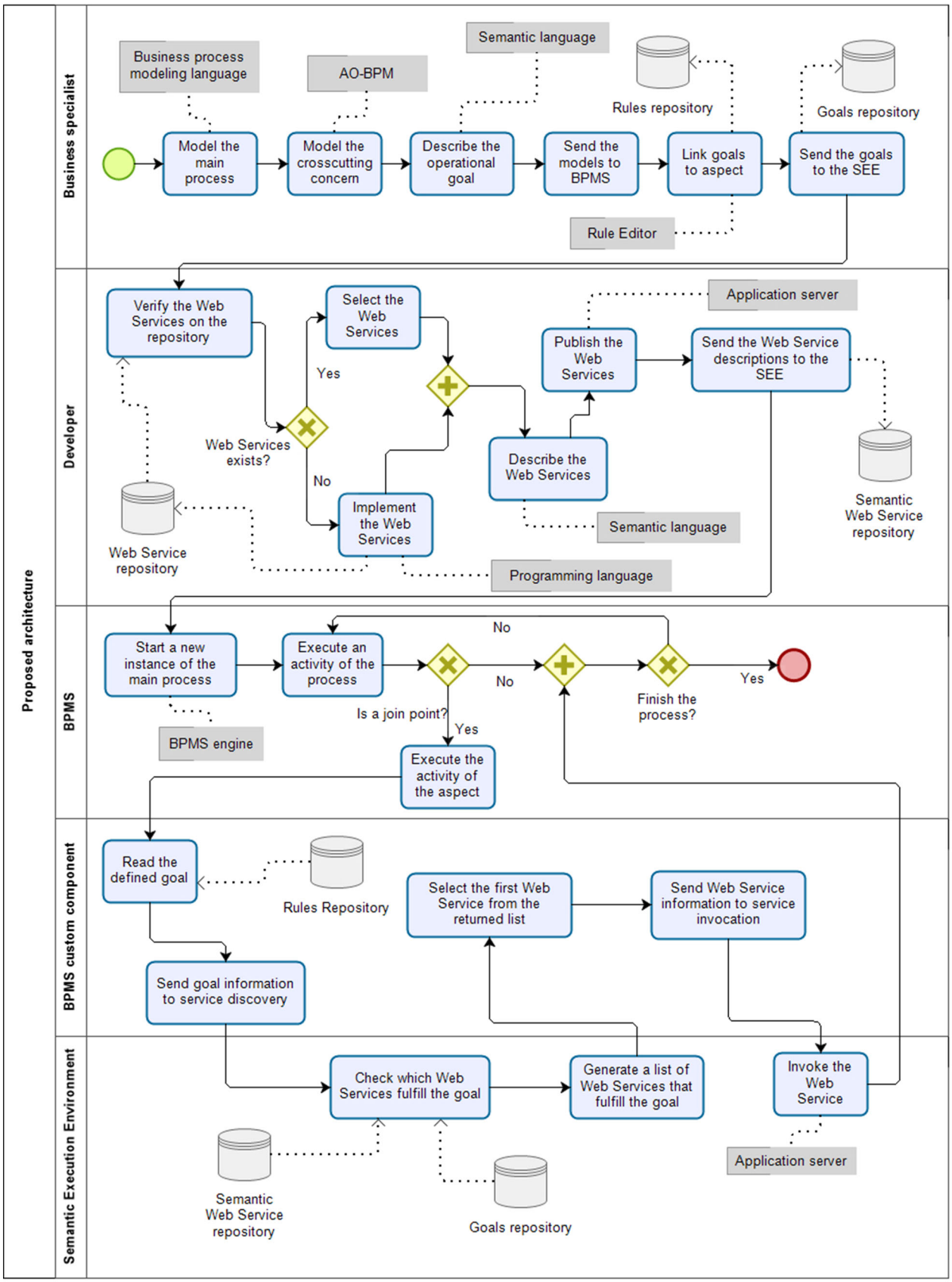


Fig. 13 Overview of proposed architecture execution

indicating the join point in the main process. We set up the "*Custom Service*" item in the YAWL Editor properties side window, thus indicating which activity must be performed by *AspectService*. In addition, we configured the *GoalService* component, previously configured in the YAWL environment, in the "*Log Information*" activity, thus indicating that the execution of the activity will be performed by the custom component developed (Fig. 15).

The definition of aspects, their advice and pointcuts for the *AspectService* was conducted using the extended version of Pointcut Editor. Along with the definition of advice, we inform the identifier of the operational objective that will be used in the discovery process. The identifier must be the same as the one given in WSMX (Fig. 16).

Ontology, semantic web services and goals were written and validated using the Web Service Modeling Toolkit (WSMT)[2] tool, which has native support for WSML syntax and integration with the WSMX environment. This tool allows to send the ontologies, semantic web services and goals created to WSMX. It also performs the process of discovering, selecting and invoking through the "*Achieve Goal*" option.

Three goals were defined, each one indicating a desirable logging strategy: log on console or terminal ("*SimpleLogGoal*"), log by email ("*MailLogGoal*") and log in database ("*DBLogGoal*"). The definitions of the three logging strategies were created based on the ontology proposed by Bastos et al. (2014), which was also written in WSML.

We created three web services descriptions in WSML ("*SimpleLog*", "*MailLog*" and "*DBLog*"), each one dealing with a logging strategy, where the details of the service's functionality (i.e., its capacity) meet one of the goals created. We also described the interface of the web service that will be invoked, with the respective rules for invoking it.

In order to verify the invocation features, we created a concrete Web Service ("*LogService*") containing three remote methods, each method indicating a logging strategy. The concrete Web Service was created using the Eclipse IDE with Apache Axis2.[3] In the invocation description of each semantic web service, we defined the remote methods, in order to allow WSMX to invoke the corresponding remote method at the end of the discovery and selection process.

At the beginning of the "*Send Articles to Review*" process, the "*MailLog*" goal was set for the "*Log Information*" activity using the Pointcut Editor. After the "*Send Invitation*" activity was executed, the *AspectService* was invoked, which resulted in the aspect being weaved into the main process. Once the "*Log Information*" activity had been enabled for execution, the *GoalService* started up, invoked the WSMX and provided the defined goal for the discovery interface.

The discovery process identifies the three semantic web services previously created and compares ontologies, functionalities and declared objects with what was described in the goal. Once the semantic web service is selected, WSMX produces a response containing the service identified, in this case the "*MailLog*". Thus, the *GoalService* proceeds with the invocation, passing the identifier found on to the WSMX invocation interface as a parameter. Once this process is completed, the process execution control returns to the YAWL engine to proceed with the main process execution.

In addition, during the execution of the process, we changed the goal "*MailLogGoal*" to "*DBLogGoal*" (Fig. 16 – Advice part). When the *GoalService* was invoked in the "*Receive Answer*" activity, a new goal was sent to WSMX, and, as result, the "*DBLog*" service was returned for later invocation. This procedure did not interrupt the execution of the main process, occurring when the "*Receive Answer*" activity waited for a user input.

WSMX provides a graphical interface, the WSMX Monitor, where helps following the process of discovering, selecting and invoking web services. Figure 17 shows the WSMX Monitor window displaying the discovery and selection process in textual form, while the left screen shows the selection of the "*MailLog*" service defined at the beginning of the process execution. The right part of the screen shows the selection of the service "*DBLog*" after changing the goal.

The results of the proof of concept showed that it is possible to successfully carry out the dynamic weaving process using the artifacts developed by Jalali et al. (2015), since the "*Log Information*" aspect was incorporated into the main process at run time. It was also possible to verify the link created between the YAWL environment and WSMX through the *GoalService* custom component. In addition, it was possible to observe the process of discovering, selecting and invoking a concrete Web Service through a semantic goal, written in WSML, that it was previous defined in the "*Log Information*" aspect using the extended version of Pointcut Editor. Finally, it was possible to observe the change in the aspect's behavior when another web service was selected and invoked to meet the new defined goal, confirming the issue of the implementation flexibility pointed out by Bastos et al. (2014).

[2] Web Service Modeling Toolkit, available at https://sourceforge.net/projects/wsmt/.

[3] Apache Axis2, available at http://axis.apache.org/axis2/java/core/.

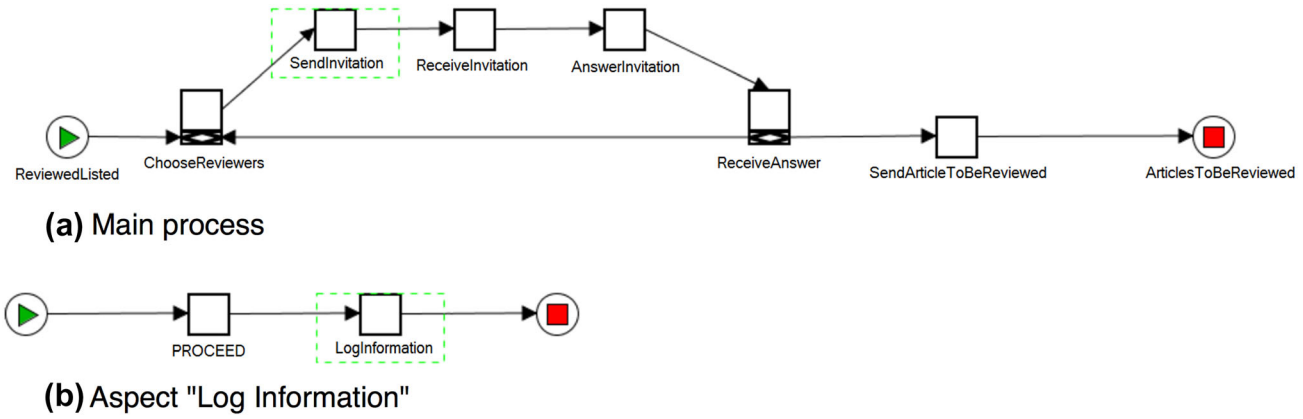


Fig. 14 "*Send articles to review*" process and the "*Log information*" aspect, adapted from Cappelli et al. (2009)

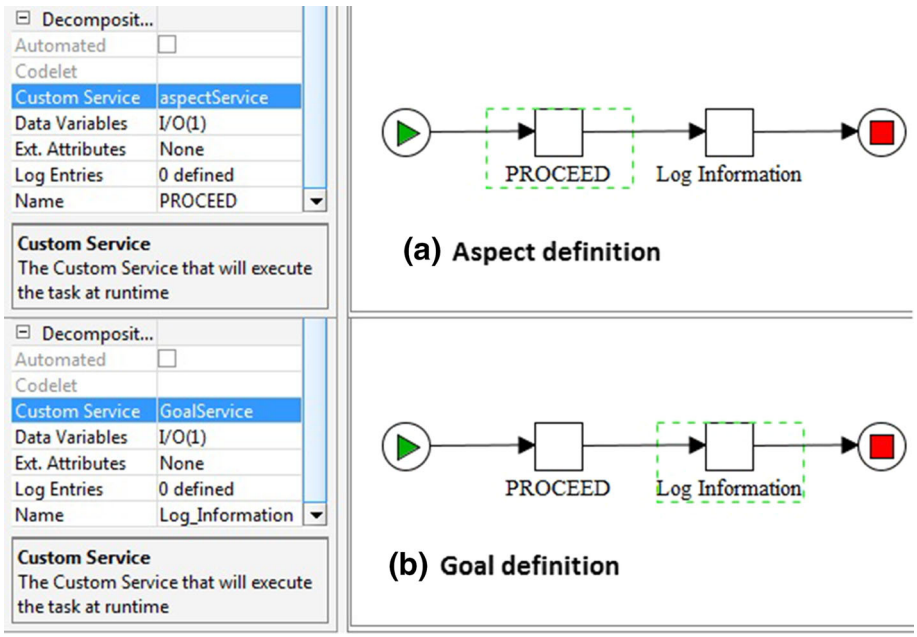


Fig. 15 Assigning the AspectService to the PROCEED task, and GoalService in the "Log Information" task, in the YAWL Editor

5.2 Second Step: Experiment

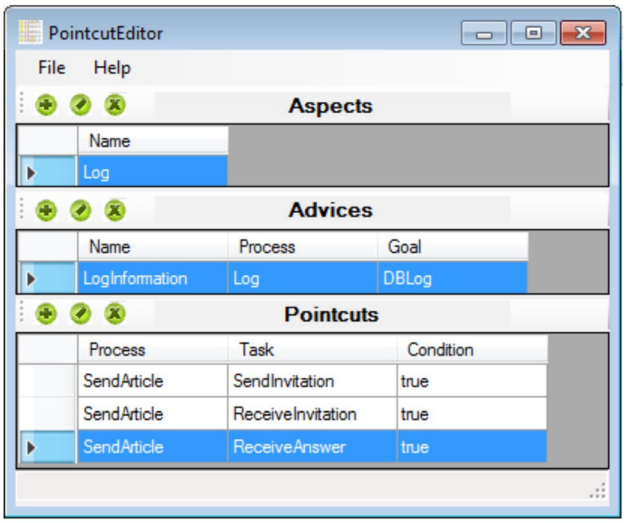
An experiment (Recker 2013) was carried out with the purpose of evaluating the interaction of the business expert and the developer with the proposed architecture, with the first acting in the modeling phase and the second acting in the implementation phase. As a result, evidence about the confirmation of the hypothesis was produced. The experiment aimed at simulating: (1) the definition of operational goals for aspects by the business experts in the modeling phase; and (2) the construction of service repositories by the developers, in the implementation phase, to realize the defined aspects.

Furthermore, the following scenarios most likely to occur in real situations were addressed in order to define the goals by the business experts:

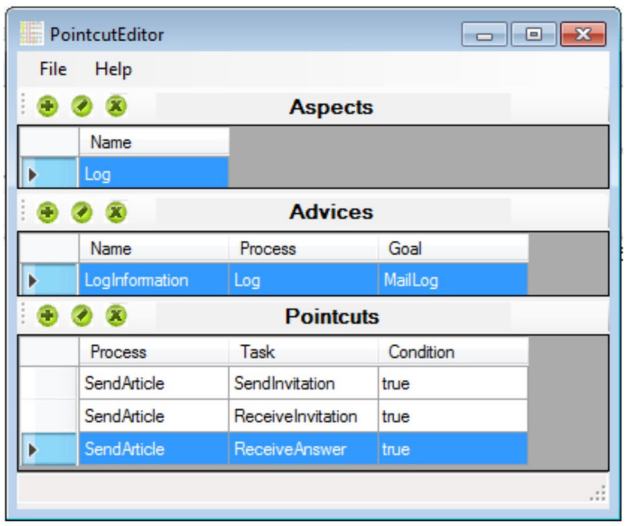
- Scenario 1: goals described by an existing web service, being present or not on the repository;
- Scenario 2: goals described without prior knowledge of the web services in the repository;
- Scenario 3: goals described by the concepts of the ontology used by the web services contained in the repository.

Scenario 1 aims to achieve the following objectives:

(a) identify the service used as the basis for defining the goal;
(b) select the service used as the basis for defining the goal;
(c) confirm that the service used as the basis for the goal is the first in the list of selected services in the discovery process.



(a) Goal "DBLog" set in the advice



(b) Goal "MailLog" set in the advice

Fig. 16 Pointcut editor illustrating the change of goal during the execution of the aspect

The 2nd and 3rd scenarios aim to achieve the following objectives:

(a) verify the quantity of services identified;
(b) verify the quantify of selected services;
(c) verify if each defined goal has at least one service selected to carry out the invocation.

The SWS-TC[4] repository was used for this experiment. This repository contains 241 services and a single ontology with all the concepts used by the services. The repository was suitable for the experiment because: (1) of the number of existing services; (2) all services are based on the same ontology definition; and, (3) there was no need to align different ontologies. Each service was converted from OWL-S to WSML for the simulation. In order to analyze the influence of the size of the service repository, we created 12 repositories numbered from 1 to 12, adding 20 more web services to each following repository (i.e., the 1st with 20, the 2nd with 40, the 3rd with 60, and so on). In the 12th repository, a web service was purposely left out to evaluate the impact of its absence. The selection of each service in the repositories was random. Each scenario was run on top of the 12 created repositories.

We defined 70 goals, numerically identified, distinct from each other by the definitions of the concepts in the postcondition section. The goals were defined for each scenario of the experiment, and therefore grouped as follows:

- 1st group (1–10): the definitions were based on existing web services;
- 2nd group (11–40): the definitions were randomly specified from the concepts presented in the ontology;
- 3rd group (41–70): the definitions were randomly specified from the concepts present in the repository containing 60 web services.

Groups 2 and 3 were further divided into three subgroups, each subgroup containing 10 goals with one variable definition, 10 containing the definition of 2 variables, and another 10 containing the definition of 3 variables.

Finally, to simulate the execution of the aspect in the YAWL environment, the *GoalService* custom component was used to perform the service discovery process starting from the defined goals. A command line utility called YSimulation, which is part of the YAWL source code, was adapted to allow sequential execution of "*Log Information*" instances. The list of identified and selected web services was recorded in a PostgreSQL database to make it possible to analyze the discovery process.

The experiment was performed on a virtual machine with 8 GB of RAM, running a 64-bit version of Windows 7 Home Premium. This VM was created using VirtualBox version 5.1.14 running on a MacBook Pro with Intel Core i7 2.3 GHz, 16 GB RAM and OS X 10.11.6 El Capitan. Java JDK 1.7_80 32 bits and PostgreSQL 9.4 was installed on this virtual machine.

All the 70 goals were executed in the 12 repositories – 840 executions of identification and selection in total. We recorded how many and which web services were identified, and how many and which ones were selected. We used the standard configuration of the WSMX discovery configuration, which consists in verifying the correspondence between goal and web service by a keyword ("Keyword Discovery") and then by similarity in the declarations included in the sections Capability and Interface ("Lightweight Discovery"). In keyword matching, the

[4] http://projects.semwebcentral.org/projects/sws-tc/.

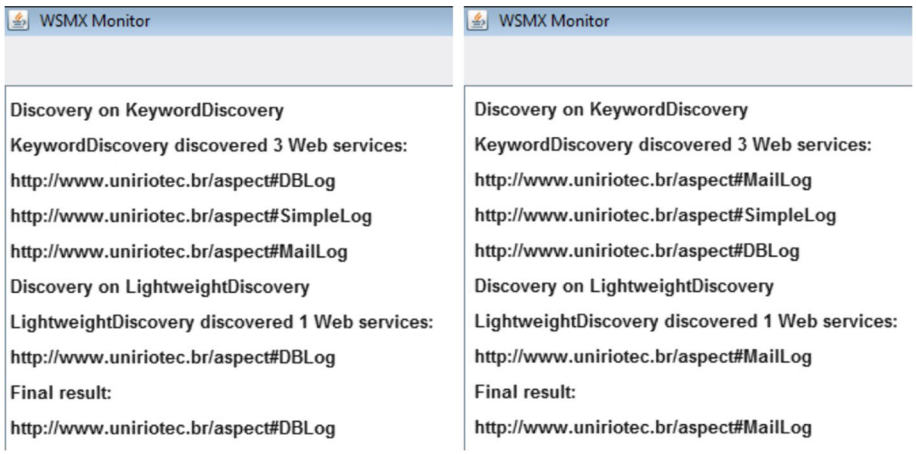


Fig. 17 WSMX monitor illustrating goal change during aspect execution

words in the non-functional property definitions are compared, and up to 6 candidate services are returned to the next step. Regarding the correspondence by declaration similarity, three types of correspondence (Exact, PlugIn and Subsumption) are applied in the declarations to return the web service that meets the sent goal. Table 2 summarizes the results.

5.3 Result Analysis and Discussion

Analyzing the data presented in the first scenario in Table 1, we observe an increase in the number of identified services and selected services. This increase occurs due to the number of services present in the repository, thus increasing the probability of the web service that was used to create the goal to be present in the repository. Likewise, we also raise the possibility of having more services that can meet that goal, but not necessarily meet the business expert's intent.

Figure 18 specifically shows the result of the web services' selection for each of the 10 goals of Group 1, where we indicate for the ones among the 10 goals, how many had web services selected (in blue) for invocation. Among the goals with web services selected, it was verified if the semantic web service used to define the goal was present in the selection (in red), and if it was the first one in the list (in yellow). From the graph we can see that the fact that a goal was created from a Web Service does not necessarily guarantee that it will be the first one in the selection, but it is very likely that it will be one of the selected services.

Another point that the graph shows is that, even with the entire collection of semantic web services, we still have a goal that obtained no selection. This may occur due to the absence of a Web Service in repository 12, although we confirmed after the simulation that all Web Services used to create the Group 1 goals were present, but it was not the first to be selected in the returned list, thus indicating that the presence of many services with similar definitions affects the discovery process.

Analyzing the second scenario, where the business experts did not know the concepts present in the repository, the arbitrary definition of the goals affected the discovery process, since it was not possible to make the definitions created randomly match with the definitions present in the collection of services. But when compared with scenario 3, in which the business expert knows the definitions of the repository, the chances are greater of obtaining a service that fulfills the requirements of the business expert.

The expectation for the 2nd and 3rd scenario was detecting the proportion of a service selected for each goal created. In total, 30 goals were created for each scenario, where each goal was submitted to the 12 repositories, 360 executions per scenario in total, where one-third (120 executions) were expected. Table 3 summarizes the execution data and shows a 25.25% difference between the expected result and the 2nd scenario, and 2.75% between the expected result and the 3rd scenario.

The execution data showed the impact caused by the business specialists' lack of knowledge of the characteristics of the service repository. The more aware the business expert was of the concepts present in the repository, the greater the chances were that the goal would be reached.

Comparing with the results presented in Table 2, the amount of services in the repository directly impacts the discovery process. When calculating the variance of the results of the 2nd and 3rd scenario (Fig. 19), we obtain a greater dispersion in the repositories containing almost all the services of the collection (repository of 10–12), and peaks in the dispersion due to the success in the discovery of services where a service had been selected for each goal.

It would be possible to obtain better results by means of the discovery process, mainly for the second scenario. However, the default configuration of the WSMX environment returns only 6 Web Services, and the version of the environment used does not have all match types (e.g.,

Table 2 Summary of simulation execution

Group	Scenario 1		Scenario 2						Scenario 3					
	1st		2nd		3rd		4th		5th		6th		7th	
Repository	I	S	I	S	I	S	I	S	I	S	I	S	I	S
1	44	1	60	1	60	–	60	–	55	5	56	–	57	–
2	56	3	58	6	59	–	58	–	60	6	60	–	57	2
3	59	2	55	3	54	–	56	–	59	10	56	1	55	2
4	59	6	60	6	53	–	60	–	56	1	55	1	55	3
5	59	7	57	7	56	1	60	–	59	6	59	3	55	1
6	55	12	60	4	60	–	58	–	54	8	58	1	56	2
7	59	13	60	8	56	1	58	–	56	10	55	2	56	2
8	57	13	60	5	58	1	55	–	52	8	54	–	60	4
9	60	13	60	9	60	1	52	–	50	7	53	2	60	4
10	59	16	60	10	60	–	53	–	57	9	58	3	60	3
11	58	15	60	9	60	–	55	–	53	9	59	2	60	3
12	59	21	60	10	60	–	58	–	54	9	59	2	60	3

I: Total identified services; S: Total of selected services

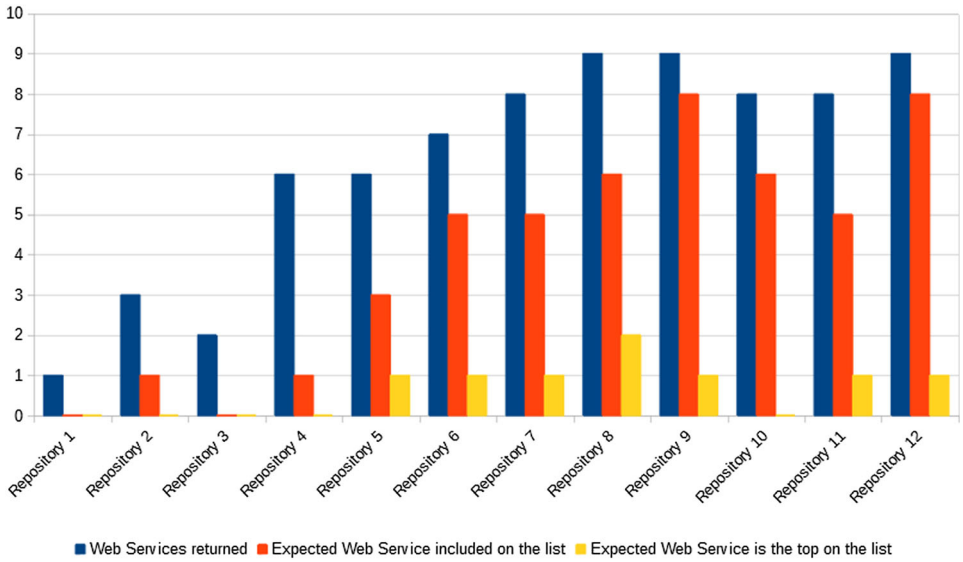


Fig. 18 Result of selecting web services from the 1st scenario

Intersection Match is not implemented[5]). Another limitation is the influence of Keyword Discovery, which considers the information contained in the non-functional properties of goal and Web Service in the evaluation. We tried to define the non-functional properties in order to impact the identification of the services as little as possible, and not direct the result to a particular group of services. Finally, another limitation are the characteristics of the service collection, with the ontology only providing the name of the concepts without their respective attributes, and the predominance of a set of concepts in specific parts of the collection. For example, the concept "Book" appears more frequently in services starting with the letters A, B and C, and does not appear much in services starting with letter G.

6 Conclusions

An architecture for service discovery that implements crosscutting concerns allows a link between the modeling and implementation phases as long as this link is defined through operational goals. These goals represent the business experts' desired services to realize the crosscutting concern in question. In the Charfi and Zhi (2015) and Shankardass (2009) approaches, the business expert needs to know previously which service will realize this crosscutting concern, and also to know if the service will comply with what he wants.

[5] Information obtained by analyzing WSMX source code.

Table 3 Summary of successful executions for the 2nd and 3rd scenario

	Executions	Percentage (%)
Total	360	100
Expected	120	33.3
2nd Scenario	29	8.05
3rd Scenario	110	30.55

The experimental results have demonstrated that this link makes it possible to find a suitable aspect implementation starting from an operational goal defined in the modeling phase. Bastos et al. (2014) conceptually demonstrated that this link was possible. The experiment has shown the viability of this link, and also revealed the strengths and limitations in defining operational goals using the WSML language.

The results obtained in the experiment suggest that the business expert can focus on the aspect modeling without necessarily worrying about its implementation since the implementation will be discovered and invoked during the execution of the process. The aspect goal definition based on the concepts present in the service repository allows that association.

To obtain better results with the proposed architecture, it would be necessary to use the concepts present in the ontology, both the business expert to describe the goals and the developer to semantically describe the services available in the repository. Table 4 shows the objectives of each scenario and which were fully or partially attended to.

A previous proof of concept showed the technical feasibility of the artifacts produced, since the "Log Information" aspect was incorporated into the main process at run time. It was also possible to verify the link created between the YAWL environment and WSMX through the *GoalService* custom component. In addition, it was possible to observe the process of discovering, selecting and invoking a Web Service through a semantic goal written in WSML. The operational goals were defined in the aspect using the extended version of Pointcut Editor. Finally, it was possible to verify the change in aspect behavior when another semantic Web Service was selected to meet the newly defined goal. The richer in detail the desired service was, the greater the chances were of it being selected in a service repository.

In addition, this work has brought the concept of operational goal nearer to the aspect, where different implementations well specified in the use of a semantic language can be identified and used to perform the activities of each aspect, and which allows different behaviors for the same advice, thus obtaining flexibility and adaptability. No related work has adopted the use of a semantic language for the aspect realization.

Our work differs from Jalali et al. (2015) by focusing on the realization of aspects, the use of a semantic language to discover the implementation for the aspects, the elaboration of an architecture that makes use of all the concepts and artifacts presented, and the use of simulation to evaluate the elaborated proposal.

A limitation of our proposal is the lack of WSMO service repositories. The WSMO concepts can be applied only to SOAP Web Services and we have not been able to t identify recent work that allows the WSMO to work with Representational State Transfer (REST) (Richards 2006). There is also the fact that the architecture is designed to work with BPMS with the ability to orchestrate and choreograph Web Services. We consider the lack of repositories to be a practical limitation of our proposal. However, service-based development is still a growing area and organizations adopt it even if they must construct

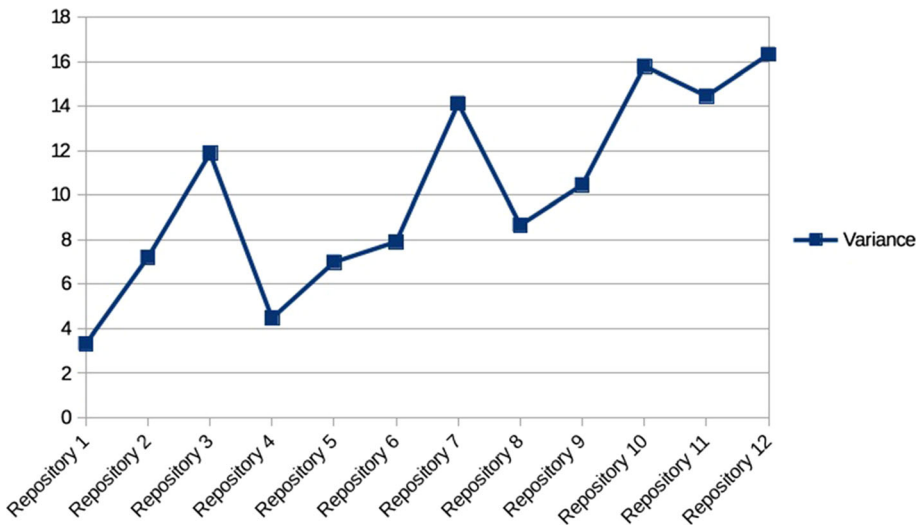


Fig. 19 Impact of the quantity of services for the discovery process

Table 4 Objectives reached by each scenario

1st scenario			2nd scenario			3rd scenario		
(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
✔	✔	!	✔	!	!	✔	✔	✔

(✔) Fully attended; (!) partially attended

services internally. Service registry/repository/directory is one of the critical success factors in implementation of SOA (Emadi and Hanza 2013; Moeini et al. 2011; Niknejad et al. 2018). The Service Broker is the stakeholder responsible to manage the registry and its main responsibility is to allow the Service Provider to register data about services and make the information available to any potential requester (i.e., Service Consumer) (Gu and Lago 2007). Service brokering has an increasingly prominent role in bridging the gap between business requirement and technology enablement (Duan et al. 2014). Vendors provide several tools for service registry, such as IBM WebSphere Service Registry and Repository,[6] Oracle Enterprise Repository,[7] Anypoint service registry[8] (MuleSoft), WSO2 Governance Registry.[9] Besides, open and shared software implementations seem to be a rising tendency. Therefore, we firmly argue that in future more and more available components can be discovered and reused that could benefit from the conceptual architecture proposed here.

As future work, we intend to further investigate the issue of discovering and selecting services returned by WSMX to the GoalService component. We also intend to improve the accuracy of the search and selection in cases where two or more Web Services are returned to the same goal, or no Web Service is returned. To do so, we may use simulation tools and metadata for the description of services. In addition, another goal is to verify the impact of the use of mediators and to realize the compatibility between different ontologies.

We will extend the solution to make it as abstract as possible, so as not to be restricted to a specific BPMS and technologies, as well as generalize the description of the goal to facilitate the work of the business experts in modeling it. Moreover, we will work with other types of services, such as REST, using wrappers for example, evaluating other semantic languages such as OWL and WSMO-Lite, and simplifying the goal modeling using tools that generate the WSML code and send it to WSMX. Finally, we will carry out case studies in real scenarios involving process modeling specialists focusing on the improvement of the architecture and produced artifacts.

[6] https://developer.ibm.com/integration/docs/wsrr/.

[7] https://www.oracle.com/middleware/technologies/enterprise-repository.html.

[8] https://www.mulesoft.com/resources/esb/service-registry-repository.

[9] https://wso2.com/products/governance-registry/.

References

Bastos A, Santoro FM, Siqueira SWM (2014) Bringing semantics to aspect-oriented business process management. In: Business process management workshops, Beijing. Springer, pp 291–302. https://doi.org/10.1007/978-3-319-06257-0_23

Cappelli C, Leite J, Batista T, Silva L (2009) An aspect-oriented approach to business process modeling. In: Proceedings of the 15th workshop on Early aspect. Charlottesville, p 7. https://doi.org/10.1145/1509825.1509828

Cappelli C, Santoro FM, do Leite JCSP et al (2010) Reflections on the modularity of business process models: the case for introducing the aspect-oriented paradigm. Bus Process Manag J 16:662–687. https://doi.org/10.1108/14637151011065955

Carsten B (2006) Bruce Carsten: lifetime achievement award winner. In: Power electron technol. http://powerelectronics.com/site-files/powerelectronics.com/files/archive/powerelectronics.com/power_systems/switch_mode_power_supplies/609PETOnlineLifetimeAchievement.pdf. Accessed 28 Feb 2017

Charfi A, Mezini M (2007) AO4BPEL: an aspect-oriented extension to BPEL. World Wide Web 10:309–344. https://doi.org/10.1007/s11280-006-0016-3

Charfi A, Zhi H (2015) Aspect-based realization of non-functional concerns in business processes. In: Bouajjani A, Fauconnier H (eds) Third international conference on networked systems, Agadir. Springer, Cham, pp 140–154. https://doi.org/10.1007/978-3-319-26850-7_10

Charfi A, Müller H, Mezini M (2010) Aspect-oriented business process modeling with AO4BPMN. In: ECFMA 2010. Paris, pp 48–61. https://doi.org/10.1007/978-3-642-13595-8_6

Duan Y, Narendra N, Du W, Wang Y, Zhou N (2014) Exploring cloud service brokering from an interface perspective. In: 2014 IEEE international conference on web services. pp 329–336. https://doi.org/10.1109/icws.2014.55

Dumas M, La Rosa M, Mendling J, Reijers HA (2013) Fundamentals of business process management. Springer, Heidelberg. https://doi.org/10.1007/978-3-642-33143-5

el Bouhissi H, Malki M, Cherif MASA (2014) From user's goal to semantic web services discovery: approach based on traceability. Int J Inf Technol Web Eng 9:15–39. https://doi.org/10.4018/ijitwe.2014070102

Emadi S, Hanza RH (2013) Critical factors in the effective of service-oriented architecture. Adv Comput Sci Int J 2(3):26–30

Fensel D, Bussler C (2002) The web service modeling framework WSMF. Electron Commer Res Appl 1:113–137. https://doi.org/10.1016/S1567-4223(02)00015-7

Fensel D, Facca FM, Simperl E, Toma I (2011) Web service modeling ontology. In: Fensel D, Simperl E (eds) Semantic web services. Springer, Heidelberg, pp 107–129. https://doi.org/10.1007/978-3-642-19193-0_7

Gu Q, Lago P (2007) A stakeholder-driven service life cycle model for SOA. In: 2nd International workshop on service-oriented software engineering. pp 1–7. https://doi.org/10.1145/1294928.1294930

Haller A, Cimpian E, Mocan A et al (2005) WSMX – a semantic service-oriented architecture. In: IEEE international conference

on web services. IEEE, pp 321–328 vol 1. https://doi.org/10.1109/icws.2005.139
Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems. MIS Q 28(1):75–105. https://doi.org/10.2307/25148625
Hollingsworth D, Hampshire UK (1995) Workflow management coalition: the workflow reference model. Document Number TC00-1003, 19, 16
ITU-T (2008). Recommendation Z. 151 (11/08): user requirements notation (URN) – language definition. Geneva, Switzerland
Jalali A (2011) Foundation of aspect oriented business process management. Master thesis. Stockholm University
Jalali A (2018) Weaving of aspects in business process management. Complex Syst Inform Model Q 2018(15):24–44. https://doi.org/10.7250/csimq.2018-15.02
Jalali A, Wohed P, Ouyang C, Johannesson P (2013) Dynamic weaving in aspect oriented business process management. In: OTM confederated international conferences "on the move to meaningful internet systems". Springer, Heidelberg, pp 2–20. https://doi.org/10.1007/978-3-642-41030-7_2
Jalali A, Ouyang C, Wohed P, Johannesson P (2015) Supporting aspect orientation in business process management. Softw Syst Model 16(3):903–925. https://doi.org/10.1007/s10270-015-0496-7
Jalali A, Maggi F, Reijers H (2018) A hybrid approach for aspect-oriented business process modeling. J Softw Evol Process 30(8):e1931. https://doi.org/10.1002/smr.1931
José HSS, Gonçalves FE, Cappelli C, Santoro FM (2016) Providing semantics to implement aspects in BPM. In: Business process management workshops, pp 264–276. https://doi.org/10.1007/978-3-319-58457-7_20
Josuttis NM (2007) SOA in practice: the art of distributed system design. O'Reilly, Sebastopol
Keller U, Lara R, Polleres A et al (2004) WSMO web service discovery. In: WSML work. Draft. http://wsmo.org/2004/d5/d5.1/v0.1/20041112/d5.1v0.1_20041112.pdf. Accessed 23 Feb 2020
Kiczales G, Lamping J, Mendhekar A et al (1997) Aspect-oriented programming. In: Akşit M, Matsuoka S (eds) ECOOP 1997 – Object-oriented programming. Springer, Heidelberg. https://doi.org/10.1007/bfb0053381
Klusch M, Nesbigall S, Zinnikus I (2008) Model-driven semantic service matchmaking for collaborative business processes. CEUR Workshop Proc 416:51–65
Laddad R (2003) AspectJ in action: practical aspect-oriented programming. Dreamtech Press, New Delhi
Moeini A, Modiri N, Azadi T (2011) Service oriented architecture adoption management roadmap. In: 7th IEEE international conference in digital content, multimedia technology and its applications. pp 119–124
Niknejad N, Prasetyo YA, Ghani I, Fajrillah AAN et al (2018) Service oriented architecture adoption: a systematic review. Int J Integr Eng 10 (6): 49–58. https://doi.org/10.30880/ijie.2018.10.06.007
OMG Business process model and notation specification (2011). https://www.omg.org/spec/BPMN/2.0/About-BPMN. Accessed 23 Feb 2020
Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2008) A design science research methodology for information systems research. J Manag Inf Syst 24(3):45–77. https://doi.org/10.2753/MIS0742-1222240302
Pourshahid A, Mussbacher G, Amyot D, Weiss M (2009) An aspect-oriented framework for business process improvement. In: Babin G, Kropf P, Weiss M (eds) E-technologies: innovation in an open world. MCETECH 2009. Lecture notes in business information processing, vol 26. Springer, Berlin, Heidelberg, pp 290–305. https://doi.org/10.1007/978-3-642-01187-0_25
Recker J (2013) Scientific research in information systems. Springer, Heidelberg. https://doi.org/10.1007/978-3-642-30048-6
Richards R (2006) Representational state transfer (REST). In: Richards R (ed) Pro PHP XML and web services. Apress, Berkeley, pp 633–672. https://doi.org/10.1007/978-1-4302-0139-7_17
Santos FJN, Cappelli C, Santoro FM et al (2011) Using goals to identify aspects in business process models. In: Proceedings of the 2011 international workshop on early aspects – EA'11. ACM Press, New York, p 19. https://doi.org/10.1145/1960502.1960507
Shankardass A (2009) The dynamic adaptation of an aspect oriented business process in a service oriented architecture platform. Cambridge University Press, Cambridge
Soffer P, Wand Y (2004) Goal-driven analysis of process model validity. In: International conference on advanced information systems engineering. Springer, Heidelberg, pp 521–535. https://doi.org/10.1007/978-3-540-25975-6_37
Sonnenberg C, vom Brocke J (2012) Evaluation patterns for design science research artefacts. In: Helfert M, Donnellan B (eds) Proceedings of the European design science symposium (EDSS) 2011, Dublin. Springer, Heidelberg, vol 286, pp 71–83. https://doi.org/10.1007/978-3-642-33681-2_7
Turetken O, Dikici A, Vanderfeesten I, Rompen T, Demirors O (2019) The influence of using collapsed sub-processes and groups on the understandability of business process models. Bus Inf Syst Eng. https://doi.org/10.1007/s12599-019-00577-4
van den Berg K, Conejero JM, Chitchyan R (2005) AOSD Ontology 1.0: public ontology of aspect-orientation. (AOSD-Europe-UT-01; No. AOSD-E). Enschede: AOSD Europe. https://research.utwente.nl/en/publications/aosd-ontology-10-public-ontology-of-aspect-orientation. Accessed 23 Feb 2020
van der Aalst WMP, ter Hofstede AHM (2005) YAWL: yet another workflow language. Inf Syst 30:245–275. https://doi.org/10.1016/j.is.2004.02.002
Weske M (2012) Business process management: concepts, languages, architectures. Springer, New York
Zugal S, Soffer P, Haisjackl C et al (2015) Investigating expressiveness and understandability of hierarchy in declarative business process models. Softw Syst Model 14(3):1081–1103. https://doi.org/10.1007/s10270-013-0356-2