TOWARDS AUTOMATION OF SOA-BASED BUSINESS PROCESSES

Fernando Lins[1,2], Julio Damasceno[1,2], Andre Souza1, Bruno Silva[1], David Aragão[1], Robson Medeiros[1,2], Erica Sousa[1] and Nelson Rosa[1]

[1]Center of Informatics, Federal University of Pernambuco, Pernambuco, Brazil

{faal2,jcd,arss,blbs,dllaa,rwam,etgs,nsr}@cin.ufpe.br

[2]Department of Statistics and Informatics, Federal Rural University of Pernambuco, Pernambuco, Brazil

{fernando.aires,julio,robson}@deinfo.ufrpe.br

ABSTRACT

The adoption of business processes to design business activities is becoming a reality to a significant number of companies. In addition, the Service-Oriented Architecture (SOA) is being used in diverse situations for the execution of business processes using computational resources. In this context, the need of business process automation appears with high relevance. In this work, we present a solution named BPA-SOA, which aims to automate SOA-based business processes, specified in BPMN, into WS-BPEL executable processes. In BPA-SOA, business and service information can be specified at the business level, translated into executable artifacts, deployed in execution-level resources and enforced at runtime. An illustrative scenario is presented to better illustrate and showcase the proposed solution.

KEYWORDS

Business Processes, Automation, Web Services, BPMN, WS-BPEL

1. INTRODUCTION

Actually, companies are widely adopting business process modeling standards to express and design the functional requirements of their businesses. In fact, Business Process Management (BPM) and Workflow Management (WfM) concepts have been used in the last three decades [12]. These techniques, in the past, have only been partially successful, or failed outright. One important reason for that is based on the lack of standards to guide the modeling, design and, mainly, execution of the business processes at that time. However, in the last years, considering the dissemination of standards that can be used to enforce business process (e.g. SOA/web services [18]), these initiatives have been revived with considerable interest.

One point that is being considered essential to the success of the BPM is the automation of the business process. Business users may want to automate the modeled business process for diverse purposes; for example, the business process can be automated for testing purposes or even to be directly used in the company. By the analysis of this context, the need of automation becomes clear; the utilization of business process should not be restricted to a design level phase, but to other phases (such as process enactment [26]).

To describe (and execute) executable business processes, a modeling standard is being widely used: the Web Services Business Process Execution Language (WS-BPEL) [13]. This modeling standard is able to specify an executable business process; a considerable number of WS-BPEL orchestration engines (computation resource to execute the business process, e.g., ActiveVos [1], Apache ODE [2] and JBOSS jBPM [11]) were developed to support the execution of WS-BPEL business processes. Adding web services capabilities to the BPEL (Business Process Execution Language) standard, WS-BPEL is actually the "the facto" standard to construct executable business processes. Typically, a business user defines a business process using a high-level language (which can be standards such as BPMN [16] or even in natural languages, such as English) and passes it to a WS-BPEL developer that takes responsibility of defining an executable business process. Despite the fact that this is a possible approach, the company may not have a WS-BPEL developer or this professional may require too much time (or significant financial costs) to perform this translation (depending on the complexity of the business process specification, the mapping of the high-level specifications into WS-BPEL executable code can be very complex). An automatic translation tool is recommended in this case. There are currently some efforts to automate the translation of the modeled business process to executable artifacts [5][8][17]. However, some of these solutions do not generate ready to deploy/execute business process artifacts. The main reason for that is because the associated approach/tool generally does not provide support for specification of all needed information (e.g., service information) at high-level. In fact, in general, the user must edit the produced artifacts to allow its execution (e.g., information related to the execution of the activities).

In this paper, we present a solution named BPA-SOA (stands for Business Process Automation using SOA) to enable the automation of SOA-based business processes (i.e., processes that can be executed using web services). This solution uses MDA (Model Driven Architecture) [14] concepts to conduct the transformations of the high-level business process to executable artifacts; it addresses from the business process modeling to its translation and deployment at execution level.

This document is structured as follows: Section 2 describes basic concepts that are necessary to better understand the proposed solution. Section 3 presents and details a methodology to automate SOA-based business processes. Section 4 introduces the BPA-SOA tooling, which provides support for the methodology described on Section 3. An illustrative scenario is introduced to showcase the BPA-SOA solution on Section 5. Section 6 presents related works. Finally, Section 7 presents the conclusions of this paper and describes some future work.

2. BASIC CONCEPTS

Prior to present the main contributions of this paper, it is important to introduce basic concepts that help to understand what is being proposed. Initially, this chapter introduces the Service Oriented Architecture (SOA), which includes key concepts to this work. Secondly, relevant concepts behind the Business Process Management (BPM), the Business Process Model and Notation (BPMN) and the Web Services Business Process Execution Language (WS-BPEL) are introduced. Finally, the Model Driven Architecture (MDA) is focused and some essential concepts presented.

2.1. Service Oriented Architecture

Service Oriented Architecture (SOA) [18] is a widely used approach based on the notion of service orientation. A service is a first-class element in SOA and systems operate through a collection of independent services, in which they can interact with several other services to perform a task. Relevant points of this technology includes the reuse of services previously implemented and tested, the possibility of shorten project time, costs and risks and the facilitation of system development and maintenance activities. SOA main objectives include: technological independence (in particular, interoperability between client/server); loose coupling (specification/ implementation of the client should not depend on the server); self-descriptive interfaces; independence of the communication protocol (the protocol choice should not influence the service functionality) and transparency in the services invocation (services should be invoked in a similar way both remotely and locally).

A key concept in the SOA context is web service. According to W3C [22], a web service can be defined as a "software system designed to support interoperable machine-to-machine interaction over a network". Considering this definition, it can be stated that this technology allows interoperable communication between different machine/users in a network (which may be the Internet). The main point is how to support this communication. Two main standards are being widely used in this aspect: SOAP and WSDL.

The Simple Object Access Protocol (SOAP) standard [23] is a XML-based protocol to enable message exchanging in a Web-based environment. This standard provides a standardize message format to be used in the communication process. The basic SOAP transmission unit is constituted by the SOAP envelope, used to describe the content of the message and to provide information on how to process that content. The envelope can be divided into two parts, namely body and header. Inside of the header, it can be found control information. In the body, it is described the content of the message. After solving the problem of how to communicate data using a standard protocol (SOAP), an important question appears: how the client can know which parameters have to be used to invoke a service? If the service interface is not described using a well known format, the client is not able to invoke the desired operation. Based on this fact, it was proposed the Web Services Description Language (WSDL) [20], which provides a formal standard to describe web service interfaces. This standard, based on XML, is used to describe what the service can do and how to invoke it. WSDL enable service developers to provide crucial information about their services, allowing customers to use it properly.

2.2. Business Process Management

In a relevant work, Weske [24] defines BPM as an approach to “support business process using methods, techniques and software to design, enact, control and analyze operational processes involving humans, organizations, applications, documents and other sources of information”. It is possible to assume, by this definition, that the design of a business process involves the adoption of a notation that should be used to model the business process. In addition, this definition also

refers to the need of the business process enactment. The business process may be executed without the utilization of computational resources (e.g., activities may be performed manually); however, one reason for the improvement of the utilization of BPM in the last years can be associated to the enforcement of business process using IT resources. In terms of business process automation, it is essential to consider processes that can be executed using such IT resources.
Figure 1, from [26], shows the BPM life-cycle, which includes four phases: process design, system configuration, process enactment and diagnosis. In the Process Design phase, the business process is modeled, which may involve the use of a high-level business process editor. The System Configuration phase is responsible to generate a set of configurations and resources needed to execute the business process. In turn, the Process Enactment phase is in charge of executing the business process using the resources generated in the previous phase. Finally, in the Diagnosis phase, some procedures (e.g., execution log analysis) can be executed to reason about the execution performance and some actions may be taken to improve the quality of the system. These actions may include, for example, a redesign of the business process in the next iteration of the business process life-cycle.

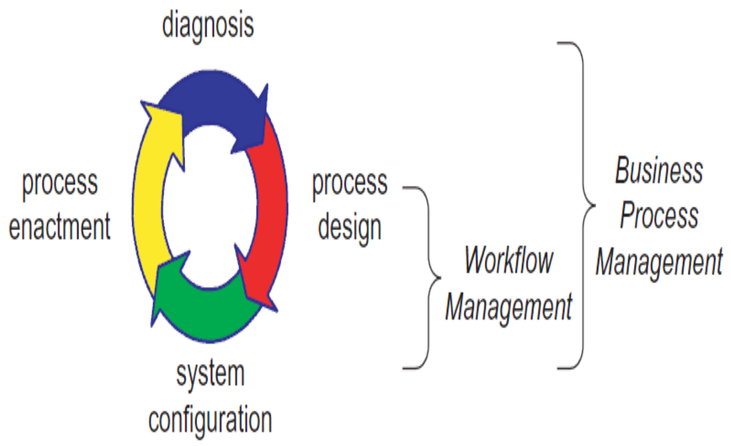


Figure 1 - Business process management life-cycle [26]

To describe executable business processes (i.e., processes that can be executed in orchestration engines [1][2]), a standard is being widely used: WS-BPEL [13]. However, business users are not usually familiarized with the technical concepts (e.g., WS-BPEL primitives) of this standard, and this fact does not help its utilization by high-level users. In this context, the Business Process Model and Notation (BPMN) [16] is being used, by high-level business users, for describing business processes based on flow-charts elements (which facilitates the modeling task, especially for non-familiarized high-level users). Both standards have played a relevant role in the Business Process Management area; however, BPMN is being adopted for specifying the business process at a high level of abstraction, whilst WS-BPEL serves as its executable version. To distinguish these two levels of business process specification in this work, the term executable business process refers to a business process that can be executed using IT resources (especially using service-oriented architectures/orchestration engines).

2.3. Model Driven Architecture

The Model Driven Architecture (MDA) [14] is an architectural solution to support software development. More specifically, MDA provides an approach to specify systems independently from the platform used to support it, to choose a particular platform to the system and to transform the system specification into specific configurations for a particular platform [14].
Three types of model are used on a MDA solution: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM). The Computation Independent Model considers the system from a computational independent viewpoint. The Platform Independent Model describes the system without referencing any specific platform. Finally, the Platform Specific Model is generated based on the previous platform independent model; when the target platform is chosen, it is inserted specific configurations directly related to the system execution considering the chosen platform.

Some important advantages can be obtained from using the aforementioned model types. Firstly, they can be used to better structure the transformations of the high-level requirements into executable configurations. Secondly, there are several orchestration engines that can be used to execute business processes. Hence, by using the platform independent and platform specific concepts, the task of generating configurations for these resources becomes more viable, because platform independent resources can be used as a basis for the development of specific business process configurations.

3. AUTOMATION OF SOA-BASED BUSINESS PROCESSES WITH BPA-SOA

This paper presents the BPA-SOA solution, which aims to automate SOA-based business processes. In this section, the main focus is the BPA-SOA Methodology, which presents a strategy to transform the high-level business process into an executable business process. In the next section, it will be introduced the tooling that was developed to support this methodology.
Figure 2 presents an overview of the proposed methodology. This methodology was described by the utilization of three basic elements: activity, artifact and connection. An activity is defined as a comportment/procedure to be realized (manually, by humans, or even automatically, by specific computational algorithms). The artifact element represents products generated/consumed by activities. Finally, a connection models temporal/logical relationships between activities and artifacts.

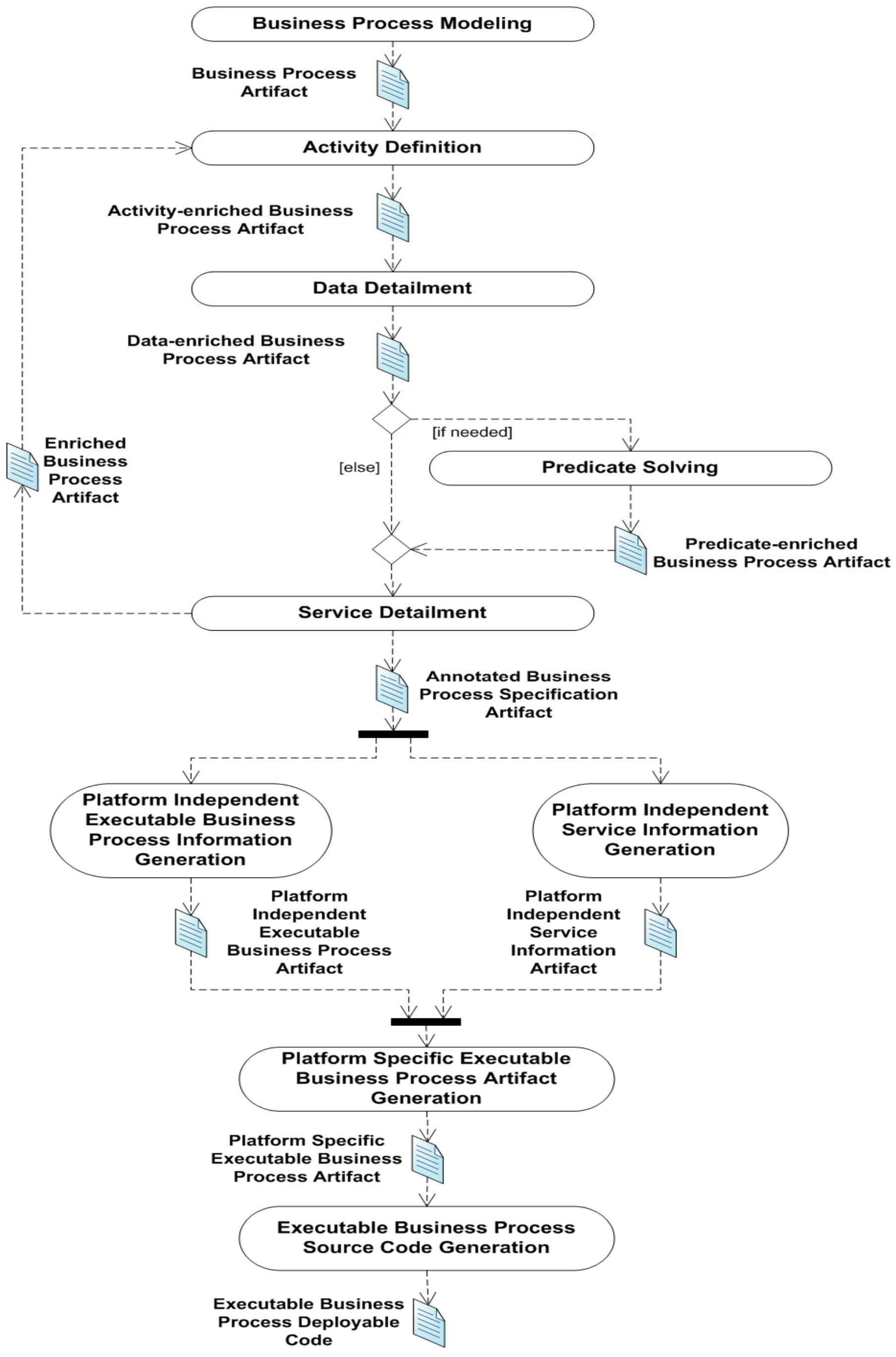

Business Process Modeling
Business Process Artifact
Activity Definition
Activity-enriched Business Process Artifact
Data Detailment
Data-enriched Business Process Artifact
[if needed]
[else]
Predicate Solving
Predicate-enriched Business Process Artifact
Enriched Business Process Artifact
Service Detailment
Annotated Business Process Specification Artifact
Platform Independent Executable Business Process Information Generation
Platform Independent Service Information Generation
Platform Independent Executable Business Process Artifact
Platform Independent Service Information Artifact
Platform Specific Executable Business Process Artifact Generation
Platform Specific Executable Business Process Artifact
Executable Business Process Source Code Generation
Executable Business Process Deployable Code

Figure 2. BPA-SOA methodology overview

The presented methodology was designed to be independent from a specific technological solution; for example, instead of using directly the WS-BPEL standard, that may not be desired in specific cases (e.g., the user may prefer to use, for example, WS-Choreography [21]), it is used the "executable business process" notion. The same idea is applied to BPMN; this solution is not directly cited in the methodology, because other business process modeling solutions may be used.

3.1. Activities detailment

The first activity of the BPA-SOA methodology is the Business Process Modeling, which consists in the functional modeling of the business process using core elements such as activity. This modeling should be based on available standards to model business process (e.g. BPMN and URN [10]). The core business process functional elements (e.g. activities and gateways) are inserted in this step. The most relevant point in this activity is related to the fact that it must be generated a valid business process considering automation purposes. For example, the business user cannot specify that a specific activity should be performed manually. Additional restrictions may also appear like the business process must not contain loops; these restrictions are necessary because, actually, only a subset of the possible BPMN processes can be automatically translated to an executable business process [17]. Next, the following activity is the Activity Definition. At this point, the user must inform how the business process activities (previously defined) are executed. An activity may be executed by diverse ways, e.g., using internal scripts. Considering a SOA-Based business process, the first option is preferred (i.e., business activities should be executed using services).

After the definition of the business process activities, it is essential to refine the business process data by associating data types and assigning variables to them; this procedure is carried out in the Data Detailment activity. For example, a specific data named "credit card" may be inserted in the business process; in this step, it must be described which data in practice represents this information (e.g., credit card number) and the data type (e.g., string). Business process data represents relevant information (e.g. credit card number) that appears in the business process; this type of information is crucial to the business process automation because they are used at execution level to perform the need functional actions. After the data detailment, it is also relevant to solve predicates used in business process (if exist). Business process modeling tools usually allow the utilization of conditional statements, but frequently do not provide resources to specify its resolution (because the focus is in the modeling task, and not in execution, which requires the specification of this resolution). Based on that, the Predicate Solving activity consists of explicitly refining predicates of the business process. In order to realize the business process is essential to refine the predicates because they must be precisely described to allow the correct translation of decision commands and conditions into executable code.

Another essential activity, especially considering SOA-based business processes, is Service Detailment. In this activity, occurs the service information enrichment of the business process by including additional information (e.g. service URI) related to the actual Web services used to realize functionalities. Service information may be selected from a local service repository to facilitate the modeling activity. A possible situation must be observed regarding to this activity: the inexistence of services to perform the functionality needed by a business process activity. In this case, the enriched business process artifact must return to the Activity Definition step. The business user needs to change the type/specification of the activity. After this definition, the artifact may continue in the methodology in order to update necessary information.

After the enrichment of the business process, two activities are executed to generate the corresponding platform independent artifacts. The first of these activities, Platform Independent

Executable Business Process Information Generation performs a mapping between the annotated business process specification into a platform-independent executable business process. In practice, “platform-independent” means that the generated business process must be independent from any particular orchestration engine. The second activity is the Platform Independent Service Information Generation. In this activity, a configuration file that contains information about services to be used in the executable business process is generated. After generating the platform independent artifacts, the goal is to generate the platform specific artifact. This generation is carried out by the Platform Specific Executable Business Process Artifact Generation activity, in which the platform independent executable process and service artifacts are transformed into a platform-specific executable business process artifact. This artifact is generated based on the target orchestration engine (e.g., Apache ODE and JBOSS jBPM). It is important to note that the aforementioned platform independent artifacts are merged to generate the platform-specific executable business process artifact. The translation rules of this phase are directly related to the target orchestration engine. If a business user/company wants to execute the business process in a specific orchestration engine, he must check if this resource is supported. If the chosen orchestration engine is not supported, he/she can act to implement the necessary translation rules or just use the generated platform-independent artifacts to guide the implementation of the business process.

Finally, in the Executable Business Process Source Code Generation activity, a ready to deploy executable business process source code is produced from the generated platform-specific executable business process artifact. Considering the target orchestration engines, it may be needed to generate other specific resources, such as additional configuration files (which may include configurations of the own orchestration engine). In addition, the generated business process code may have to be disposed in a specific format (in a compressed file, for example). The output of this activity can be directly used in the target engine.

4. AUTOMATING SOA-BASED BUSINESS PROCESSES USING THE BPA-SEC TOOLING

In the previous section, it was described the BPA-SOA methodology, proposed to automate SOA-based business processes. At this point, it will be presented the BPA-SOA tooling, a set of tools that was developed to support this methodology. Figure 3 presents the architecture of the BPA-SOA tooling.

The main components of the presented architecture are the BPA-SOA Editor, the BPA-SOA Translator, the BPA-SOA Deployer and the BPA-SOA Repository. Next subsections present details of these components. To implement this architecture, it was necessary to choose the standard used to describe the business process and its executable artifact. For the business process modeling, the Business Process Model and Notation (BPMN) [15] is being the most used standard to model business process. In terms of the description of executable business processes, the "de facto" standard in this context is the Web Services Business Process Execution Language (WS-BPEL) [13], which has been widely used by a considerable part of the SOA community. These two standards are used as a basis for the implementation of the BPA-SOA tooling.

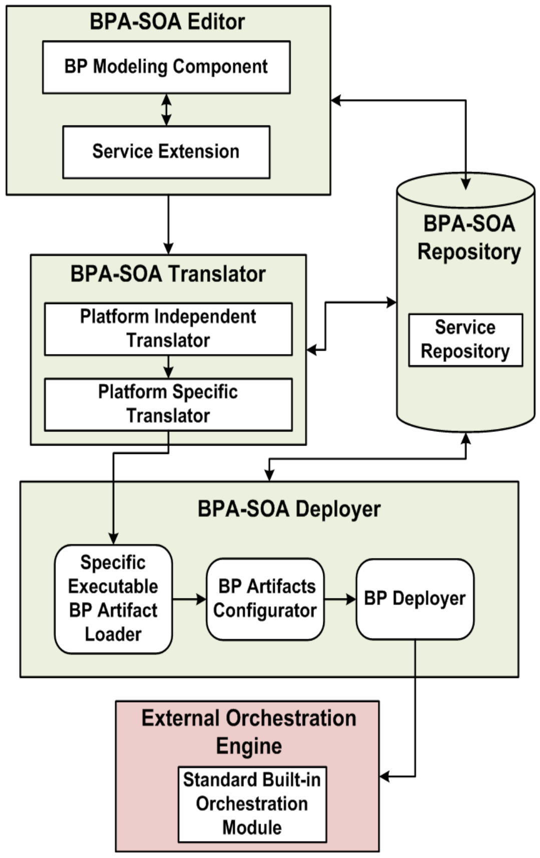

BPA-SOA Editor
BP Modeling Component
Service Extension
BPA-SOA Repository
Service Repository
BPA-SOA Translator
Platform Independent Translator
Platform Specific Translator
BPA-SOA Deployer
Specific Executable BP Artifact Loader
BP Artifacts Configurator
BP Deployer
External Orchestration Engine
Standard Built-in Orchestration Module

Figure 3. BPA-SOA tooling architecture

4.1. BPA-SOA Editor

The BPA-SOA Editor is a BPMN-based tool, which also provides support for the service modeling task. It was implemented as an extension of the Eclipse BPMN Modeler [7] and has two internal views, namely business view and service view, to support the modeling of the SOA-based business process.

It can be observed in Figure 3 that the BPA-SOA Editor is composed internally by two modules: BP Modeling Component and Service Extension. The first module, BP Modeling Component, implements the business view and provides support to the specification of the functional business process (by the utilization of BPMN standard elements). This module also provides support to the Activity Definition, Data Detailment and Predicate Solving activities. The Service Extension module implements the service view; this module provides support for the binding between external Web services to BPMN activities. Service information about supported services can be retrieved from the local service repository; based on this fact, the user must only bind the service to the BPMN activity, without inserting manually additional information such as service URI. A screenshot of the service view is available on Figure 5.

4.2. BPA-SOA Translator

Another relevant element presented in the BPA-SOA architecture (see Figure 3) is the BPA-SOA Translator. This element is responsible for performing the translation of the high-level SOA-based BPMN business process into platform independent and platform specific artifacts. The BPA-SOA Translator is composed internally by two internal translators: the Platform Independent Translator and the Platform Specific Translator. These internal translators are focused on this section.

According to the BPA-SOA methodology (see Figure 2), after generating the Annotated Business Process Specification Artifact, the next activities are related to its translation into platform independent artifacts. The first component of the BPA-SOA Translator, Platform Independent Translator, is responsible for generating the necessary platform-independent artifacts (related to the executable business process and service information). The goal of this internal translator is the generation of the Platform Independent Executable Business Process Artifact and Platform Independent Service Information Artifact (see Figure 2), which are used as a basis for the generation of platform specific artifact/deployable code. Considering the chosen standards, BPMN and WS-BPEL, the Platform Independent Translator maps annotated BPMN into a platform independent WS-BPEL along with service descriptions.

The generated platform independent artifacts are used directly as input in the Platform Specific Translator in order to generate the required specific artifact. If the chosen WS-BPEL orchestration engine is supported, the execution of platform specific translations will enable the automation process and the execution of the business process at low-level. If the chosen orchestration engine is not supported, two situations may occur: the platform specific translation rules must be implemented by the involved users or the platform independent artifacts can be used as a basis for the specification of the specific WS-BPEL process based on the required engine. In the second case, instead of direct generating executable WS-BPEL code, the involved users work on the manual specification of the WS-BPEL business process.

In a second moment, the platform independent artifacts are used to generated the Platform Specific Executable Business Process Artifact (see Figure 2), i.e., WS-BPEL code for a specific orchestration engine (e.g., Apache ODE and ActiveVos). The Platform Specific Translator is responsible for this task. The generated specific artifact is obtained from the merge between the Platform Independent Platform Independent Executable Business Process Artifact and Platform Independent Service Information Artifact.

It is worth observing that only a subset of BPMN elements may be translated into WS-BPEL and some constraints must be followed in the actual version of the BPA-SOA tooling to allow the automation process: (i) business processes cannot contain elements supporting parallelism (split/merge), (ii) only one empty start element (and one empty end element) in the BPMN business process is allowed, and (iii) business processes cannot contain loops. The first and second constraint do not have considerable impact in most BPMN business diagrams; parallelism is generally used only in very specific cases and most BPMN diagrams has one empty start and one empty end element (however, other start/end events are used with some frequency). On the other hand, constraint (iii) is more relevant, because the utilization of loops in BPMN business diagrams is more common. Based on this fact, the implementation/support for this restriction is considered a future work in the context of this paper.

4.3. BPA-SOA Deployer

The BPA-SOA Deployer is responsible for generating ready to deploy WS-BPEL code for the chosen orchestration engine. This functionality is implemented by three internal components (see Figure 3): Specific Executable BP Artifact Loader, BP Artifacts Configurator and BP Deployer. The Specific Executable BP Artifact Loader receives the deployment request and the Platform Specific Executable Business Process Artifact; after that, it performs an initial validation to check if required elements (e.g., needed service information) are available. By its turn, the BP Artifacts Configurator is able to perform required actions (based on the previous validation) on the received artifact (e.g., it can generate additional configuration files, depending of the chosen orchestration engine) and also organizes it in a format that is ready to deploy in the chosen WS-BPEL orchestration engine (in a considerable number of engines, the generated files must be disposed in a specific format, e.g., in a .zip file). Finally, the BP Deployer is invoked to deploy the generated files in the chosen engine. The BP Deployer can use a management interface or may just create/copy necessary files to the target engine. All presented elements are directly related to specific execution-level resources; based on this fact, if a chosen resource is not supported by the BPA-SOA tooling, the involved users may have to work in order to generate the required artifacts in the proper format required by the orchestration engine and to deploy it.

4.4. BPA-SOA Repository

The BPA-SOA Repository store information for the business process automation and it is used in different ways: for finding new candidate services for the SOA-based business process, as a log repository for registering important events in the process execution and so on. The BPA-SOA Repository is directly connected to the BPA-SOA Editor and to the BPA-SOA Deployer. The editor accesses the Repository to retrieve information required at modeling stage, e.g. the set of available services that can used in the business process. By its turn, the BPA-SOA Deployer retrieves additional parameters/resources that must be used to generate the required artifacts (e.g. the WSDL interface of the automated business process).

5. ILLUSTRATIVE SCENARIO: VIRTUAL TRAVEL AGENCY

In order to showcase/evaluate the BPA-SOA solution, it is introduced an illustrative scenario that will be introduced/implemented in this section. This illustrative scenario is the Virtual Travel Agency (VTA). Initially introduced in [19], VTA consists of a travel agency that acts as an interface between many different travel business companies and end-users interested in travel. The Virtual Travel Agency is responsible for all relevant aspects of customer´s travel in such way that they usually only defines the travel requirements (e.g., arrival/departure time) and the VTA, using computational resources (e.g., Internet and Web Services), works to achieve them.

A considerable number of use cases appear in the context of the Virtual Travel Agency, such as “buy a national airline ticket” and “rent a car”. For the purpose of this paper, the “buy a national airline ticket” is considered. This use case, modeled with the BPA-SOA solution, is presented on Figure 4. Initially, the business user defines the business process using standard BPMN elements (see activity Business Process Modeling in Figure 2). Next, the business process is enriched through the executing of the Activity Definition (e.g., defining the type of the tasks) and Data Detailment activities (e.g., defining the business process data). As the business process also includes predicates, it is necessary to perform the activity Predicate Solving (e.g., to check if the chosen flight is available).

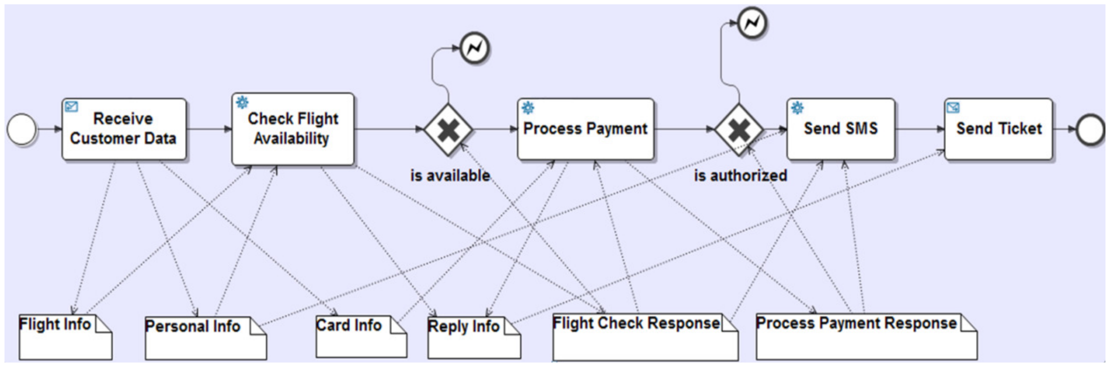
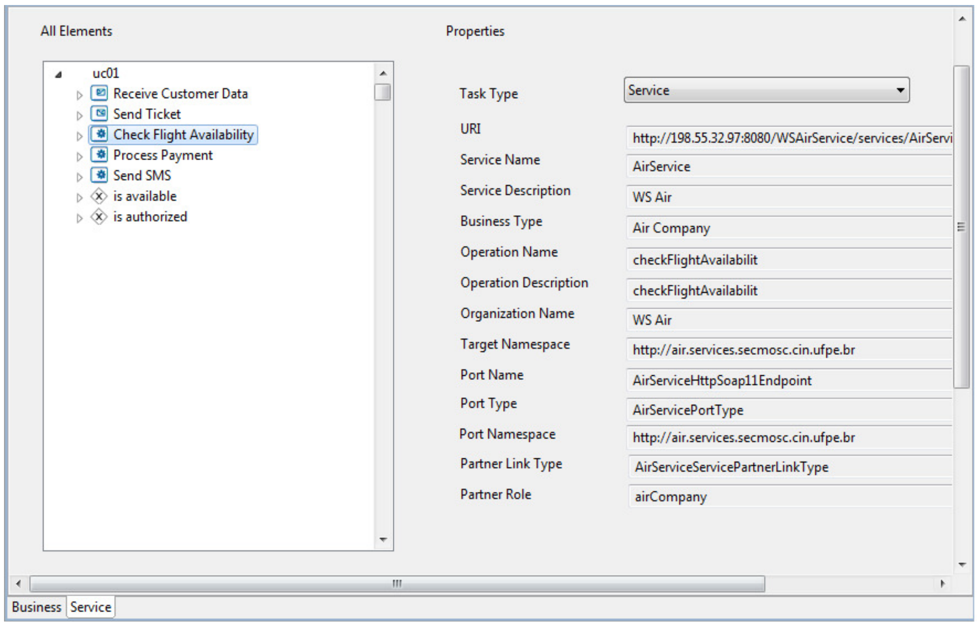


Figure 4. VTA use case "buy a national airline ticket"

The presented business process has five tasks, in which three of them (Check Flight Availability, Process Payment and Send SMS) have task type “service”. The Receive Customer Data task has task type "receive" and the Send Ticket task has task type "send". As can also be observed in Figure 13, data objects have also been modeled (e.g., Personal Info and Process Payment Response). Services were selected using the service view, which is presented (using a screenshot) on Figure 5. In the left part of the figure, the BPMN tasks and gateways are disposed. In the other side of the figure, the service information of the task "Check Flight Availability" is presented.


All Elements

- uc01
  - Receive Customer Data
  - Send Ticket
  - Check Flight Availability
  - Process Payment
  - Send SMS
  - is available
  - is authorized

Properties

Task Type	Service
URI	http://198.55.32.97:8080/WSAirService/services/AirServi
Service Name	AirService
Service Description	WS Air
Business Type	Air Company
Operation Name	checkFlightAvailabilit
Operation Description	checkFlightAvailabilit
Organization Name	WS Air
Target Namespace	http://air.services.secmosc.cin.ufpe.br
Port Name	AirServiceHttpSoap11Endpoint
Port Type	AirServicePortType
Port Namespace	http://air.services.secmosc.cin.ufpe.br
Partner Link Type	AirServiceServicePartnerLinkType
Partner Role	airCompany

Figure 5. Service view

At this point, the Annotated Business Process Specification is now ready to be translated into platform independent and platform specific artifacts. The BPA-SOA Translator is invoked and the artifacts are produced. Fragments of the generated Platform Independent Executable Business Process Artifact are presented as follows:

```
(1)     <process xmlns="http://www.cin.ufpe.br/BPA-SOA/bpel/process" name="vta">
(2)     ...
(3)     <sequence name="main">
(4)             <receive name="_nH2uE6ZYEeC2me24iA5k2g"/>
(5)             <invoke name="_nH3VIaZYEeC2me24iA5k2g"/>
(6)             <if name="_nH4jQaZYEeC2me24iA5k2g">
(7)                     <sequence name="if-sequence">
(8)                             <invoke name="_nH38MaZYEeC2me24iA5k2g"/>
(9)                             ...
(10)                    </sequence>
(11)                    ...
(12)            </if>
(13)            ...
(14)    </sequence>
(15)    </process>
```

It is relevant to observe, initially, that this platform independent file describes the "skeleton" of the WS-BPEL process; to become an executable process, additional information must be provided. For example, lines 4-6 and 8 include WS-BPEL primitives that must be enriched to be executed in an orchestration engine. Next, it is presented the generated Platform Independent Service Information Artifact:

```
(1)     <service-information xmlns="http://www.cin.ufpe.br/BPA-SOA/service-information">
(2)     <composition-interface operationName="buyNationalTicket" processName="vta">
(3)     <parameter name="person_name" type="string" paramType="input"/>
(4)     ...
(5)     </composition-interface>
(6)     ...
(7)     <services-desc>
(8)     <service-desc        name="AirService"        portName="AirServiceHttpSoap11Endpoint"
portNamespace="http://air.services.secmosc.cin.ufpe.br"
taskID="_nH3VIaZYEeC2me24iA5k2g">
(9)             <businessType>Air Company</businessType>
(10)            <operation portType="AirServicePortType" name="checkFlightAvailabilit"/>
(11)            <partnerLink type="AirServiceServicePartnerLinkType" role="airCompany"/>
(12)            ...
(13)    </service-desc>
(14)    ...
(15)    </services-desc>
(16)    </service-information>
```

By the analysis of the presented code, it can be noted that lines 2-5 contain the description of the interface of the SOA-based business process (which will be used later to generate the WSDL interface of the automated business process). In addition, lines 8-13 present information related to a participant Web service (e.g., service name).

After generating the platform independent artifacts, the Platform Independent Translator (described on Section 4.2) is called to generate the Platform Specific Executable Business Process

Artifact. The following code presents fragments of the generated WS-BPEL process for the Apache ODE orchestration engine [2].

```
(1)     ...
(2)     <wsbpel:partnerLinks>
(3)     <wsbpel:partnerLink name="airserviceInvoke" partnerLinkType=
"sss0:AirServiceServicePartnerLinkType" partnerRole="airCompany"/>

(4)     ...

(5)     </wsbpel:partnerLinks>

(6)     <wsbpel:variables>

(7)     <wsbpel:variable name="checkFlightAvailabilitRequest" messageType=
"sss0:checkFlightAvailabilitRequest"/>

(8)     ...

(9)     </wsbpel:variables>

(10)    <wsbpel:sequence name="main">

(11)    <wsbpel:receive name="_nH2uE6ZYEeC2me24iA5k2g" operation=
"buyNationalTicket" partnerLink="initializer" .../>

(12)    <wsbpel:assign>

(13)            <wsbpel:copy>

(14)            ...

(15)            </wsbpel:copy>

(16)    ...

(17)    </wsbpel:assign>

(18)    ...

(19)    <wsbpel:invoke name="_nH3VIaZYEeC2me24iA5k2g" operation=
"checkFlightAvailabilit" partnerLink="airserviceInvoke" inputVariable=
"checkFlightAvailabilitRequest" outputVariable="checkFlightAvailabilitResponse"
portType="sss0:AirServicePortType"/>
(20)    ...
```

It can be noted that that two relevant pieces of information were generated: partnerLinks (line 2 to line 5) and variables (line 6 to line 9). These variables are used in the variable association between the BPMN business process and web service composition described in WS-BPEL. This association (which consists of using the WS-BPEL primitive <copy>) is also shown in Lines 12-17. Relevant WS-BPEL activities, such as <receive> (line 11) and <invoke> (line 19) have been also included in this artifact with needed information (such as name and operation).

Finally, the BPA-SOA Deployer is called to receive the Platform Specific Executable Business Process Artifact and performs its deployment in the Apache ODE engine (one of the supported orchestration engines in the BPA-SOA solution).

6. RELATED WORK

Some research papers already focused on the business process automation problem. Ouyang [17] has a seminal paper about BPMN to WS-BPEL translation. Despite of advancing on this subject, the proposed translation generates artifacts that needs manual intervention to insert relevant information (e.g., all needed partnerLinks) and relevant parameters of the service invocation primitive <invoke> (e.g., portType and operation). Doux [5] proposes ATL transformations to guide the translation process; the implemented translation rules is based on the BPMN2BPEL tool, which is supported by Ouyang work [17]; based on this fact, the same considerations (e.g., relevant information is not generated by the tool) should be observed. White [25] also presents a BPMN to WS-BPEL translation, but it has not been implemented by a tool. In addition to that, only a reduced number of translation rules were presented (the main objective was to illustrate how the mapping could takes place, and not to provide a entire set of rules to be implemented in a translation tool). Giner [8] presents an MDA-based approach that uses EMF and ATL. This approach seems promising because MDA concepts are widely used to illustrate and guide the approach; however, relevant information (e.g., BPMN task type) is not considered in the translation process, generating a less rich WS-BPEL specification.

In addition to academic papers, some commercial tools that work with the BPMN to BPEL translation are also available. In this context, it can be cited the eClarus Business Process Modeler [6], the Intalio BPMS Designer [9] and the Cloud Apps [4], developed by the Business Process Incubator [3]. All of these tools were tested and generated WS-BPEL code for the Virtual Travel Agency illustrative scenario presented on the previous section; some of these tools present similar concepts of the proposed BPA-SOA Methodology (e.g., related to the Predicate Solving and Service Detailment activities) and are able to generate an executable WS-BPEL code. However, they do not generate platform independent WS-BPEL, which can be considered a disadvantage when the chosen orchestration engine is not supported; the involved users will not have an intermediate-level file to use as a basis for the development of the specific WS-BPEL process.

7. CONCLUSIONS AND FUTURE WORK

The dissemination of business process management standards and service-oriented architectures, actually, is a fact. The Business Process Model and Notation (BPMN), for example, is being widely adopted by companies to model their businesses. The need of automating business processes, in this context, is also relevant; business users may want to execute the modeled business process due to diverse reasons (e.g., to provide a system that fulfill the business needs or for testing purposes). SOA-based systems are being used to support this task due to diverse reasons: reuse of pre-existing and already tested services, pay per usage business model and so on.

To support the aforementioned needs, the present work proposed a solution named BPA-SOA, which automates SOA-based business processes. To accomplish this task, it was presented a methodology that comprises all necessary steps from the business process modeling to its execution. A tooling was also presented to support the aforementioned methodology. In addition, BPA-SOA was showcased in a real-world scenario (VTA) to better illustrate how it can be used in practice.

Some future works are being considered in the context of this work. Firstly, it is planned to minimize the restrictions of the business process modeling activity (i.e., to reduce/eliminate restrictions that are necessary to allow the automation process). Secondly, a web-based version of the BPA-SOA Editor is also planned. Finally, the adoption of non-functional requirements (such as performance, cost and security) in the business process modeling and automation is also being planned and executed.

REFERENCES

[1] Active Endpoints (2009) ActiveVos Business Process Management Suite. Available at http://www.activevos.com (last visit at 3rd May 2010).

[2] Apache Software Foundation (2008) Apache Orchestration Director Engine (ODE). Available at http://ode.apache.org/ (last visit at 15rd May 2010).

[3] Business Process Incubator (2011) Business Process Incubator Initiative. Available at http://www.businessprocessincubator.com (last visit: 30rd March 2012).

[4] Business Process Incubator (2011) Cloud Apps for Windows. Available at http:// www.businessprocessincubator.com/cloud-apps-fatware.html (last visit: 30rd January 2012).

[5] Doux, G., Jouault, F., And Bezivin, J. (2009) Transforming BPMN process models to BPEL process definitions with ATL. In 5th International Workshop on Graph-Based Tools, 2009.

[6] Eclarus Software (2009) eClarus Business Process Modeler v. 2.1. Available at http:// www.eclarus.com/ (last visit: 18rd November 2011).

[7] Eclipse Foundation (2008) The BPMN Modeler. Available at www.eclipse.org/bpmn (last visit: 15rd February 2012).

[8] Giner, P., Torres, V. And Pelechano, V. (2007) Bridging the Gap between BPMN and WS-BPEL: M2M Transformations in Practice. Available at http://citeseerx.ist.psu.edu/ viewdoc/summary?doi=10.1.1.83.6295 (last visit: 20rd December 2011).

[9] Intalio (2011) Intalio BPMS Designer. Available at http://www.intalio.com/bpms (last visit: 30rd March 2012).

[10] ITU-T 2008 (2008) Recommendation Z.150 (02/03): User Requirements Notation (URN) – Language definition. Geneva, Switzerland; 2008. 206 pages.

[11] JBOSS Comunitty (2009) JBoss jBPM WS-BPEL Runtime User Guide. Available at http://www.jboss.org/jbossjbpm/bpel/ (last visit at 20rd December 2011).

[12] Ko, R.K.L. (2009) A Computer Scientist's Introductory Guide to Business Process Management. Crossroads, vol. 15, n. 4.

[13] OASIS (2007) Web Services Business Process Execution Language. Available at http:// www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel (last visit: 12th February 2012).

[14] OMG (2003) MDA Guide Version 1.0.1. OMG, Technical report.

[15] OMG (2011) Business Process Management Initiative. Available at http://www.bpmn.org/ (last visit: 25th Dec 2011).

[16] OMG (2011) Business Process Model and Notation v 2.0. Available at http://www.omg.org/spec/ BPMN/2.0/PDF/ (last visit: 24th March 2012).

[17] Ouyang, C. et al. (2009) From business process models to process-oriented software systems. ACM Transactions on Software Engineering and Methodology, vol. 19, no. 1, pp 1--37.

[18] Papazoglou, M. and Heuvel, W. (2007) Service oriented architectures: approaches, technologies and research issues. VLDB Journal, v. 16, 389-415.

[19] Stollberg, M. et al. (2004) WSMO Use Case Modeling and Testing. Available at http://www.wsmo.org/2004/d3/d3.2/20041004 (last visit: 29th November 2011).
[20] W3C (2001) Web Services Description Language v. 1.1. Available at http:// www.w3.org/TR/wsdl (last visit: 28th February 2012).
[21] W3C (2004) Web Services Choreography Description Language Version 1.0. Available at http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/ (last visit 30th January 2011).
[22] W3C (2004). Web Services Glossary. Available at http://www.w3.org/TR/ws-gloss/ (last visit: 30th January 2012).
[23] W3C (2007) SOAP v. 1.2. Available at http://www.w3.org/TR/soap/ (last visit: 30th January 2012).
[24] Weske, M., Van Der Aalst, W. M. P. And Verbeek, H.M.W. (2004) Advances in Business Process Management. In Data & Knowledge Engineering, v. 50, pp. 1-8.
[25] White, S. A. (2005) Mapping BPMN to BPEL Example. Technical report, IBM Corporation. Available at www.bpmn.org (last visit: 04th March 2012).
[26] van der Aalst, W.M.P. et al. (2003) Business Process Management: A Survey. In Proceedings of the Business Process Management International Conference (BPM'03), LNCS 2678, pp. 1-12.