AN ARCHITECTURAL PERSPECTIVE OF THE RELATIONSHIP BETWEEN BPMN 2.0 AND BPEL

Diego M. Campos, Jorge L. R. Becerra
Universidade de São Paulo - Escola Politécnica
Rua Luciano Gualberto, travessa 3, n. 380. CEP 05508-900 - São Paulo - Brazil
diegomcampos@usp.br, jorge.becerra@usp.br

ABSTRACT
The BPMN 2.0 brings a notation that is readily understandable by all business users, it is a bridge for the gap between business process design and process implementation and a standard mechanism to exchange formats between different vendor tools. And BPEL is meant to be used to execute the process implemented. Following the above objectives, there are several articles that demonstrate how to relate both languages in various forms inside the software development process. But this paper shows the relationship using the concepts of software architecture instead of only the concepts of software process.

This work aims to demonstrate the BPMN 2.0 and BPEL relationship in a software architecture perspective by defining a model driven architecture, using the concept of views, viewpoints and a software process. An example of a process is given in BPMN and the generated BPEL code is demonstrated.

KEY WORDS
BPEL, BPMN 2.0, business process, software architecture, architectural views, MDA.

1. Introduction

The goal of BPMN is to provide a notation that is understandable by all business users, from the business analysts to the technical developers responsible for implementing the technology that will perform those processes and to the business people who will manage and monitor those processes. Another goal is to ensure that XML languages designed for the execution of business processes, such as BPEL (Business Process Execution Language) can be visualized with a business-oriented notation [2].

The first article about relating BPMN and BPEL shows a direct relationship utilizing the basic elements of the languages. This technique relates only the basics elements and is not efficient for complex models [13]. From this research, others articles appeared, trying to relate the models utilizing patterns [14][15], utilizing automata theory, conversion to Petri nets, formal logic approach, functional logic programming languages and others [10]. Only in 2008 appeared some researches suggesting a relationship from architectural views [16][17]. At the present moment the problem is not completely solved because there is a fundamental discrepancy between BPMN, which is based on the graph theory and BPEL, which is essentially a programming language based on blocks [10].

The main objective of this work is to demonstrate a different approach by rising the level of abstraction to investigate the relationship between the BPMN 2.0 and BPEL, looking through a software architecture perspective. And an example process of a purchase order is given to clarify the process and the relationship. To do this, a MDA [1] approach is utilized to define a modelling environment in a business process oriented way [4,6]. The concept of architectural views is used to define the relationship.

This paper is structured as follows. Section 2 presents the notion of BPMN 2.0, BPEL and a brief explanation of architectural views and viewpoints. Section 3 shows the MDA modelling approaches and the relationship of BPMN and BPEL in the software architecture perspective. Section 4 presents the related work, Section 5 the results and discussions and Section 6 presents the conclusion.

2. Conceptual Knowledge

2.1 BPMN 2.0 Specification

BPMN eases business process modelling by providing a notation and an interchange format that can be used to exchange BPMN process definitions between different tools. The goal of the specification is to enable portability of process definitions, so that users can take process definitions created in one vendor’s environment and use them in another vendor’s environment [2,12].

The specification organizes its modelling elements into categories. The four basic categories are: flow objects, data, connecting objects, swim-lanes, and artifacts. Flow objects are the main elements to describe the behaviour of a business process. There are three kinds of flow objects: events, activities, and gateways [12].

Flow objects are connected to each other by connecting objects, which are four kinds: sequence flow, message flow, association and data association. Data objects are used to provide input and output information on activities and are represented with five elements: data objects, data inputs, data outputs, data stores and properties. Modelling elements are grouped through swim-lanes, which may be pools and lanes. Artifacts are used to provide additional information about the process, there are two kinds, group and text annotation [2,3,12].

The Figure 1 shows the core subset of BPMN elements of the flow and connecting objects categories.
Several elements in BPMN are subject to store or convey items during process execution. This is similar to the variable construct common to many languages. The primary construct for modeling data within the process flow is the DataObject element. A DataObject has a well-defined lifecycle, with resulting visibility constraints. A DataStore object provides a mechanism for activities to retrieve or update stored information that will persist beyond the scope of the process.

Activities and sub-processes may produce data during or as a result of execution. Data requirements are captured as DataInputs objects and data that is captured using DataOutputs objects [2]. The Figure 2 shows some examples of the data objects category.

While BPMN shows the flow of data and the association of data artifacts to activities, it is not a data flow language and does not provide a built in model for describing the structure of data. Instead it formalizes hooks which allows data structures to be externally defined [2].

### 2.2 BPEL

“A business process defines how to coordinate the interactions between a process instance and its partners. In this sense, a BPEL process definition provides and/or uses one or more WSDL services and provides the description of the behaviour and interactions of a process instance relative to its partners and resources through Web Service interfaces. That is, BPEL is used to describe the message exchanges followed by the business process of a specific role in the interaction” [5].

BPEL is a language for specifying business process behaviour based on Web Services. BPEL is meant to be used to model the behaviour of both executable and abstract processes. Executable business processes model the behaviour of a participant in a business interaction. Abstract business processes are partially specified processes that are not intend to be executed [5]. An abstract process may hide some of the required concrete operational details. Abstract Processes serves as a descriptive role, with more than one possible use case, including observable behaviour and process template [5].

![Figure 3. General structure of BPEL](image)

The Figure 3 contains the general structure of BPEL. The process can be explained as follows: “Inside the process tag, which is the outer element of such an XML document, there are basically three different sections. In the first section partners and partnerLinks are defined. The partnerLinks announce the external programs which are involved such as clients and Web services to the BPEL process. The second section contains all the variables of the process. Variables embody all messages and XML documents used in a BPEL process. These variables can be documented in the BPMN using the defined data objects elements. The third section is the orchestration logic. Invoke elements call external programs by using their SOAP interface. They correspond to the activities in the BPMN graph. The switch tag maps the Gateway Object of BPMN to BPEL. Sequences are sequential actions while flows correlate to the fork objects of BPMN. All tags can include many attributes, which have to be filtered out of the participating Web service description” [4].

The optimization of BPEL for software operations renders them less suited for direct use by humans to design, manage, and monitor business processes. There is a human level of inter-operability or portability that is not addressed by the language so this is where the BPMN comes to fulfill this gap.

### 2.3 Architectural Views and Viewpoints

In the conceptual framework of the recommended practice for architecture description, an architectural description is organized into one or more constituents views. Each view addresses one or more of the concerns of the system stakeholders. The term view is used to refer to the expression of a system's architecture with respect to a particular viewpoint [9,11].
“A viewpoint establishes the conventions by which a view is created, depicted and analyzed. In this way, a view conforms to a viewpoint. The viewpoint determines the languages (including notations, model, or product types) to be used to describe the view, and any associated modelling methods or analysis techniques to be applied to these representations of the view. These languages and techniques are used to yield results relevant to the concerns addressed by the viewpoint” [9,11].

This concepts are used in this work because the basics definition of the BPMN and BPEL indicates that, BPMN is used by the business stakeholders and BPEL by technicians stakeholders who have particular concerns. In the next section the concerns that involving the BPMN and BPEL stakeholders are separated, using the concepts of view and viewpoints of the software architecture and a software development process.

3. Modelling Process

According to [4], the author defines a modelling approach in the analysis and design phase of the software engineering development process, using a business process oriented way.

The figure 4 illustrates the concept of the modelling technique, where in the analysis phase the business process is defined utilizing the BPMN and the participants of the process are identified. In the design phase an automatic map from BPMN to BPEL is made and the generated XML code may be refined with the definition of the SOAP interfaces missing [4].

![Figure 4. Modelling approaches in the analysis and design phase](image)

According to [6], “two elements (or sets of elements) of different models may represent the same concept at different levels of abstraction or from different viewpoints” [6]. Such dependencies must also exist between a business process model and the workflow model that implements it. Moreover these mapping dependencies may stands between concepts from separate domains. When modelling an application on a business-oriented way, this situation must have to take into account. This could be achieved by defining the task and the participant correspondent to a process and the structural mapping to the service components that realized the task.

The figure 5 shows the structural mapping where there is a package called Process, representing the activity and the agent involved. And a package called Application that represents how the process package is implemented utilizing an use case model.

![Figure 5. Structural mapping from process to application](image)

The software architecture presented in Figure 6 defines the modelling environment. It establishes how a process may be described. It therefore drives the way the execution environment is implemented [6].

![Figure 6. “Relations between meta-models”](image)

In Figure 6 an activity may invoke a service using parameters. A role may be assigned to an organizational entity. This assignment may be constrained by a condition that holds on the Organizational Entity. For example, a role may be assign to the immediate superior of the person who performs the preceding activity [6].

Each package in figure 6 has specific concerns, the Organization concerns about how the workers are structured in the company and Process package concerns about the many process defined in the same company. In this sense, these packages can be defined as different views from the software architecture and the Service package is another view responsible for executes the process that workers in the organization are responsible
for. The Relation package explicits the relationship of two or more elements from different models that represent the same concept at different levels of abstraction.

This work utilizes this approach to specify the relationship between BPMN and BPEL in a software engineering process and software architecture.

### 3.1 From BPMN To BPEL

This section presents the relationship between BPMN and BPEL. It focuses on the control-flow perspective. First, an example is shown in BPMN language. Second, the process is modelled in the context of the software process and architecture defined earlier in this section.

Not all BPMN processes can be mapped to BPEL in a straight-forward way [2]. That is because BPMN allows the modeller to draw almost arbitrary graphs to model control flow, whereas in BPEL, there are certain restrictions such as control-flow being either block-structured or not containing cycles. For example, an unstructured loop cannot directly be represented in BPEL.

#### 3.1.1 Process for handling a purchase order

The Figure 7 illustrates a business process for handling a purchase order. On receiving the purchase order from a customer, the process initiates and invokes the enter process order from CRM to store the purchase and payment information, invoke the schedule work from ERP for production and shipment the order and finally the Send Ack task is invoked to notify the customer.

#### 3.1.2 Utilizing the modelling approach

In figure 8 the architecture for the business process is illustrated. A role may be assigned to an organization entity. This assignment may be constrained by a condition that holds on the Organization Entity. The role, in the example process, is the customer that is responsible for initiate the process. A process activity may invoke a component (this case a web service that is responsible for calculate the total price and others) using an interface and according to the behaviour specified in the BPMN model that will be mapped to BPEL.

![Figure 8. Architecture for the purchase order process](image)

Figure 8. Architecture for the purchase order process

The software architecture in Figure 8 defines the modelling environment for the given example. The software process establishes when and what to model in the analysis and design phases. In the analysis phase the process is modelled and role is assigned. In design phase the generated code in BPEL must be refined to consume the interfaces of the services it must invoke.

The next section is specific to show an example of the BPEL generated code.

### 3.2 BPEL Generated Code

The Figure 9 shows an example of code (fragmentary code for better readability) generation in BPEL that corresponds to the elements in the BPMN model.

```xml
<process>
  <sequence>
    <receive name="ReceivePO" partnerLink="Customer"/>
    <invoke name="EnterPO" partnerLink="CRM"/>
    <invoke name="ScheduleWO" partnerLink="ERP"/>
    <invoke name="SendAck" partnerLink="Customer"/>
  </sequence>
</process>
```

Figure 9. BPEL Process

To execute the BPEL process, a BPEL engine is required which parses the BPEL code and executes the contained instructions. Examples of existing BPEL engines are ActiveVOS [7] by Active Endpoints and the Oracle BPEL Process Manager [8]. All engines have in common that the BPEL process, which has to be deployed itself as well, needs to be supplemented. A BPEL package usually comprises either the WSDL documents for the involved Web services themselves or the respective URLs where they can be found. At this point it should be mentioned that because BPEL needs partnerLinkType tags for each involved Web service that are not included in a WSDL file by default. Some engines even require more supplementary files in the packages that contain information about partners.
4. Related Work

The work of White [13] was the first research to use the BPMN notation to model a BPEL process. The relationship is directly obtained between elements of the two notations. Ouyang [14,15] proposes the relationship through translation standards-based BPMN process models to BPEL models. In the work of Benatallah [18], it was proposed to define the relationship, using finite state automata.

Current techniques of the relationship between BPMN and BPEL focus on a core set of BPMN elements [19] or specific elements such as translation between element ‘OR’ of BPMN to ‘LINK’ in BPEL [20]. The techniques also impose restrictions on the structure of BPMN diagrams - links must have an entry point and a single exit point [21]. Gond and Xiong [22] deal with the problem of translating BPMN to BPEL processes and sorts them into deadlock, synchronization and communication. Götz et al. [23] addresses the issue through token analysis to identify the components automatically with the disadvantage that there is not a formal method of verification.

The work of Balouki and Hafid et al [24,25], specify the behaviour of executable system in order to make the process manageable and executable. For this, it was used the BPEL language to conceptualize the behaviour in the reference model of ODP [27]. In the work of [26] was applied the modelling services with MDA (Model Driven Architecture) to generate the BPEL language. Although these researches do not use the concept of abstraction levels to relate elements of different viewpoints, they were the first papers to relate BEPL concepts with MDA for transformation and generation of BPEL models.

5. Discussion and Results

The main objective of this work is to demonstrate that software architectural views and viewpoints can be used to relate elements that are common used in the software development process. Using the software architecture, the problems of relationship can be treat in a higher level of abstraction. And utilizing different views and viewpoints, it is easier to find the relationship and the problems that may emerge than looking only through the software development process.

With a simple example, the results are: the BPMN and BPEL elements mapped in the software architecture that drives the way the execution environment is implemented. The software architecture emphasizes that BPMN and BPEL are elements in different views and viewpoints, but can be related each other.

6. Conclusion

This paper presents the relationship between the BPMN 2.0 specification and BPEL. The relationship is obtained thought the software architecture perspective. To achieve this, first was proposed a software process in a business oriented manner and an architecture with different views that drives how the specification in BPMN could be mapped to BPEL.

The main contribution of this paper is to show that it is possible to raise the level of abstraction of the languages BPMN and BPEL to the perspective of software architecture. And through the views and viewpoints obtain the relationship between the elements. This capability is used to highlight some of the benefits of the Object Management Group (OMG) [2], Model Driven Architecture (MDA) [1] initiative: “raising the level of abstraction at which development occurs, which, in turn, will deliver greater productivity, better quality, and insulation from underlying changes in technology”.

The proposed work also explains the concepts of the data object elements in BPMN because it reveals a mechanism to define the state of the process in a structured way, this is important in the relationship with the variables in BPEL. The BPMN it self is not sufficient to model a complete process oriented architecture, but it makes clear where are the hooks that can be extended by other architectural elements that must be specified, like semantic of data and the monitoring and auditing properties.

References


