Yet Another BPEL Extension for User Interactions*

Mohamed Boukhebouze, Waldemar Pires Ferreira Neto, and Lim Erbin

PReCISE Research Center, University of Namur, 5000, Belgium
{mohamed.boukhebouze,waldemar.neto,lim.erbin}@fundp.ac.be

Abstract. In this paper, we propose a BPEL extension that deals with the user interactions expression in a Web service composition. This extension defines new data interaction activities to allow user providing, selecting or getting data. Moreover, the extension proposes a new type of interaction events that allow processing of the user interaction. The user interaction specification helps to generate a user interface for the Web service composition which is made by transforming the described user interactions to user interface components.

Keywords: BPEL extension, User Interaction, User Interface.

1 Introduction

A Web service is an autonomous software application that can be described, published, discovered, and invoked across the Web [6] by using a set of XML based standards such as UDDI, WSDL, and SOAP. Web services can be combined together in order to fulfill the user request that a single Web service cannot satisfy [10]. This mechanism is known as Web service composition. A Web service composition consists of several Web services orchestration or Web services choreography processes that deal with a functional need which is unsatisfied by a single Web service [10].

Several initiatives have been conducted to provide languages such as WS-BPEL (Web Services Business Process Execution Language) [5] that allow the description of Web service composition execution. WS-BPEL (BPEL for short) is an XML based standard that provides a syntax to define the Web service composition behavior (control flow) and mechanisms that enable data manipulation (data flow).

This language expresses the Web service composition process in a fully automated way. Users are not able to interact with the Web services until the end of the process execution. For example, users are not able to provide the input to a Web service at runtime, they are not able to cancel the process execution, or they are not able to have some intermediary output from a Web service. However, many Web service composition scenarios require user interactions [3]. These user interactions can be classified into four types [8]:

* Research supported by la Wallonie.

O. De Troyer et al. (Eds.): ER 2011 Workshops, LNCS 6999, pp. 24–33, 2011.

- *Data input interaction* represents the fact that the user provides data to a Web service at runtime. For example, a user provides a licence Number to a car renting Web service;
- *Data output interaction* represents the fact that a Web service composition makes data available to the user. For example, the trip scheduling Web service composition presents the car renting price to the user;
- *Data selection* represents the fact that the user can select a data from a set of data. For example, a user can make a choice between flight or driving as a means of transportation;
- *Interaction event* represents an indicator that a user interaction has been carried out. For example, a cancellation is done by a user during the Web service composition execution (by pressing a button for example).

The difference between the data interaction types (input, output and selection) and the interaction event is that the data interaction types allow changing the data flow of the composition, while an interaction event changes only the control flow of the composition regardless of the data (e.g. the cancelling process is done regardless of the data).

Unfortunately, the BPEL language does not support such types of user interaction. For this reason, BPEL meta-model needs to be extended to express user interactions in a Web service composition. In addition, a user interface for the composition needs to be developed in order to help the user to interact with the Web services at runtime.

In this work, we propose a BPEL extension for the user interactions expression. This extension is called UI-BPEL (User Interaction Business Process Execution Language). UI-BPEL supports the expression of the four types of the user interactions explained above by introducing new BPEL elements: (1) a new BPEL activities (*DataInputUI*, *DataOutputUI*, *DataSelectionUI*) to express the user data interactions; (2) a new type of BPEL event (*InteractionEventUI*) to express the interaction event; (3) an extension of the BPEL's *Pick* and *Scope* activities that support the new *InteractionEventUI*. The main objective of this extension is to allow the generation of a user interface for the Web service composition based on the described user interactions. This generation is performed by transforming the UI-BPEL user interactions elements to specific user interface components. For example, transforming a *DataInputUI* to a text box, transforming a *DataOutputUI* to a label, and transforming a *DataSelectionUI* to a combo box.

UI-BPEL is part of the several initiative efforts of BPEL extension to address the user interaction expression in a Web service composition. An example of such extensions is BPEL4People [3], which introduces user actors into a Web service composition by defining a new type of BPEL activities to specify user tasks. However, this extension focuses only on the user task and does not deal with the design of a user interface for the Web service composition. Another example of BPEL extensions that addresses the user interaction is BPEL4UI (Business Process Execution Language for User Interface) [1]. BPEL4UI extends the *Partner Link* part of BPEL in order to allow the definition of a binding

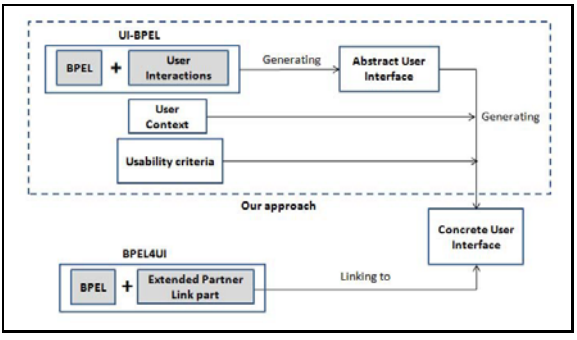

Fig. 1. UI-BPEL Vs. BPEL4UI

between BPEL activities and an existing user interface. This user interface is developed separately from the composition instead to be generated.

Figure 1 shows the difference between our approach (UI-BPEL) and BPEL4UI approach. The top side of the figure depicts our approach that proposes to extend BPEL with user interactions so that a user interface for the Web service composition can be generated. The generated user interface is described in two abstraction levels: first, we propose to generate, from a UI-BPEL process, an abstract user interface, which is independent to any interaction modality (e.g. graphical input, vocal output) and computing platform (e.g. PC, smart phone) [4]. We then generate a concrete user interface based on the user context and by taking into account a set of user interface usability criterions [7]. This approach is compliant with the existing user interface description languages (like UsiXML [4]), which describe a user interface at different abstraction levels. The bottom side of Figure 1 shows the BPEL4UI approach that proposes to extend the *Partner Link* of BPEL in order to link the composition with an existing concrete user interface (HTML/JavaScript). This approach does not allow the generation of a user interface adapted to the user context (user preference, user environment, and user platform) and the usability criteria (e.g the interface should respect the size of the device screen).

In the remainder of this paper, we focus on the description of the UI-BPEL. In section 2, we show a Web service composition scenario that requires user interactions. We present, in Section 3, an overview of the UI-BPEL meta-model with an illustrative example. Next, in Section 4, we present an extension of the Eclipse BPEL editor that supports the UI-BPEL. Finally, we conclude this paper with a discussion about the generation of the user interface from the proposed extension and our future works.

2 Scenario

In this section we introduce the scenario of the purchase order process, which requires user interactions. A customer requests a purchase order by providing

the necessary data such as the item, and the customer address (user interaction: data input interaction). Upon receiving a request from the customer, the initial price of the order, initial price of the order is calculated and a shipper is selected simultaneously. The shipper selection needs to be made by the customer who selects a shipper from a list of available shippers (user interaction: data selection). When the two activities are completed, the final price is calculated and a purchase order needs to be presented to the customer (user interaction: data output interaction). If the customer accepts his purchase order, she/he selects a payment method either by cash or by credit card (user interaction: data selection). There is a discount if the customer pays by cash. Next, a bill is shown to the customer (user interaction: data output interaction). Finally, the user the customer could cancel her/his request for an purchase order before receiving the receipt of the payment (user interaction: interaction event).

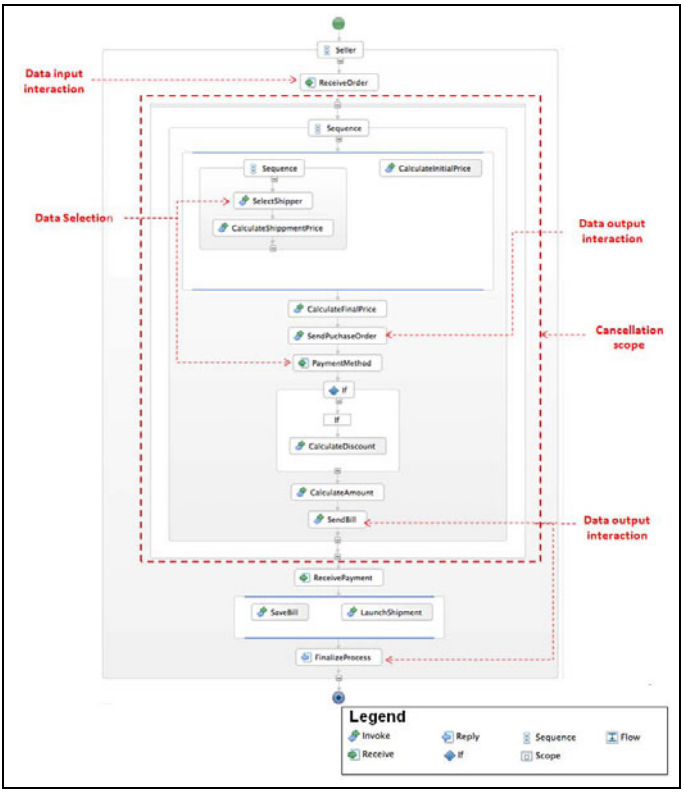

Fig. 2. Purchase Order Process expressed in BPEL

Figure 2 illustrates the purchase order process expressed using the graphical representation of Eclipse BPEL Designer graphical notations [2]. BPEL does not support any type of user interaction required by the scenario. For example, BPEL cannot express the fact that the customer needs data interaction to provide

the order. Moreover, BPEL cannot express the fact that the process can make the purchase order available to the customer using data output interaction. In addition, BPEL cannot describe the data interaction that allows the customer to choose a shipper and a payment method. Finally, BPEL does not support the fact that the customer has the ability to cancel the process before the payment is received.

In the next section, we propose an extension of BPEL that deals with user interactions required by the purchase order process scenario. The user interaction specification helps to generate a user interface for the purchase order process.

3 UI-BPEL

In this section, we present our BPEL extension (called UI-BPEL) that addresses the BPEL user interaction expression issue.

3.1 UI-BPEL Meta-Model

UI-BPEL extends the BPEL [5] by adding new data interaction activities and a new type of interaction events. Figure 3 presents an overview of the UI-BPEL meta-model. It shows the classes that model the user interactions and their connections with some relevant classes of the BPEL meta-model.

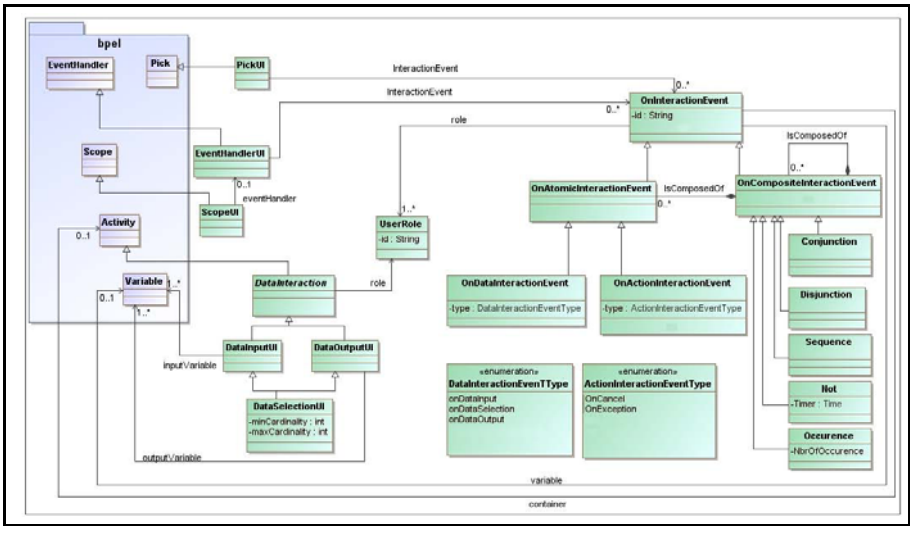

Fig. 3. Overview of the UI-BPEL Meta-model

In the following, we present the main classes of the UI-BPEL meta-model.

– ***DataInteraction*** is a generalization class that represents the new UI-BPEL activities.

- ***DataInputUI*** is a class that represents the data input activity. This activity is similar to the BPEL *Receive* activity, since it suspends the composition precess execution while waiting for an input from the user. However, unlike the BPEL *Receive* activity, the process waits for an event where data is provided, instead of a message from another Web service. This type of event will be explained afterwards.
- ***DataOutputUI*** is a class that represents the data output activity. This activity specifies which variable contains data to be presented to the user.
- ***DataSelectionUI*** is a class that represents the data selection activity. This activity allows the user to select one value (or a subset of values) from a set of values of a specific variable. The number of selectable values can be defined by using the minCardinality property (how many values must be selected at least) and/or the maxCardinality property (the maximum number of elements have to be selected). Like ***DataInputUI***, the ***DataSelectionUI*** activity suspends the execution of the process until receiving an event of selecting data as we will explain afterwards.

UI-BPEL proposes a new type of event to process the user interaction. These events help to indicate when a user interaction is carried out, so that a specific action can be launched. UI-BPEL defines the user interaction events as following:

- ***OnInteractionEvent*** is a generalization class that represents a new type of BPEL event. Such an event defines the occurrence of a user interaction. An interaction event can be either an atomic event that designates a predefined event*OnAtomicInteractionEvent* or a composite event that combines atomic and/or composite events *OncompositeInteractionEvent*.
- ***OnAtomicInteractionEvent*** is a generalization class that represents the occurrence of a data interaction or an interaction action done by user.
- ***OnDataInteractionEvent*** is a class that indicates the fact that a Data interaction has been done. An OnDataInteractionEvent can be:
  - *onDataInput* indicates that new data has been entered by the user. This event allows the process to resume its execution when it was being suspended by a *DataInputUI* activity.
  - *onDataSelection* indicates that some data has been selected by the user. This event allows to resume the process execution when it was being blocked by a *DataSelectionUI* activity.
  - *onDataOutput* indicates that new data has been presented through the user interface. This event is used to notify that the user validates a data output. For example, an onDataOutput event can be the confirmation that the user accepts the copyright licence of a Web service.
- ***OnActionInteractionEvent*** is a class that represents the notification of the occurrence of a predefined interaction action. These predefined interaction actions are:
  - *OnCancel* indicates that a cancellation action has been performed by the user.

- *OnException* indicates that an exception has occurred in the user interface (e.g. the User Interface Failure). This event is used by the BPEL *Fault handler* part.

OnActionInteractionEvent is used to handle the cancellation or the exception that can happen within a scope of the Web service composition.

– ***OnCompositeInteractionEvent:*** is a class that expresses a combination of atomic and/or composite events. According to [9], a composite event can be:

- *Disjunction(e1, e2):* specifies that at least one of the two interaction events is detected;
- *Conjunction (e1, e2):* specifies that interaction events take place without taking into account of their occurrence order;
- *Occurrence (e1, number of occurrence):* specifies multiple occurrences of the same interaction event;
- *Sequence (e1, e2):* specifies the sequence of interaction events
- *Not (e1, t):* characterizes the fact that an interaction event has not happened at time t.

OnCompositeInteractionEvent can be used to process the user interactions aspect in the Web service composition. For example, if a user does not provide an input data during time t (*Not (onDataInput, t)*), the process should be canceled. *OnCompositeInteractionEvent* can also be used to process the grouping of a set of data interactions on the same User interface container. For example, if we group the data input of the customer information, and the data selection of a shipper on a same user interface component, the event *Conjunction (on customer information Input, on Shipper Selection)* models the fact that the customer information is provided and a shipper Selected. This could resume the execution of a suspended process by the data input and the data selection activities. Note that, grouping the data interaction on a same user interface component is handled by the user interface generation method. The data interaction grouping is not in the scope of this paper.

UI-BPEL defines also defines some useful new classes, for example:

– ***UserRole:*** is a class that represents which user is responsible of *DataInteraction*. *UserRole* is an important element for the user interface generation process since a user interface should be generated for each different role.
– ***PickUI:*** is a class that extends the BPEL *Pick* activity in order to take into account *onInteractionEvent* listening.
– ***ScopeUI:*** is a class that extends the BPEL *Scope* activity in order to add on its *EventHandler* the *onUserEvent* listening.

3.2 UI-BPEL Example

We now illustrate the UI-BPEL of the purchase order scenario presented in Section 2.

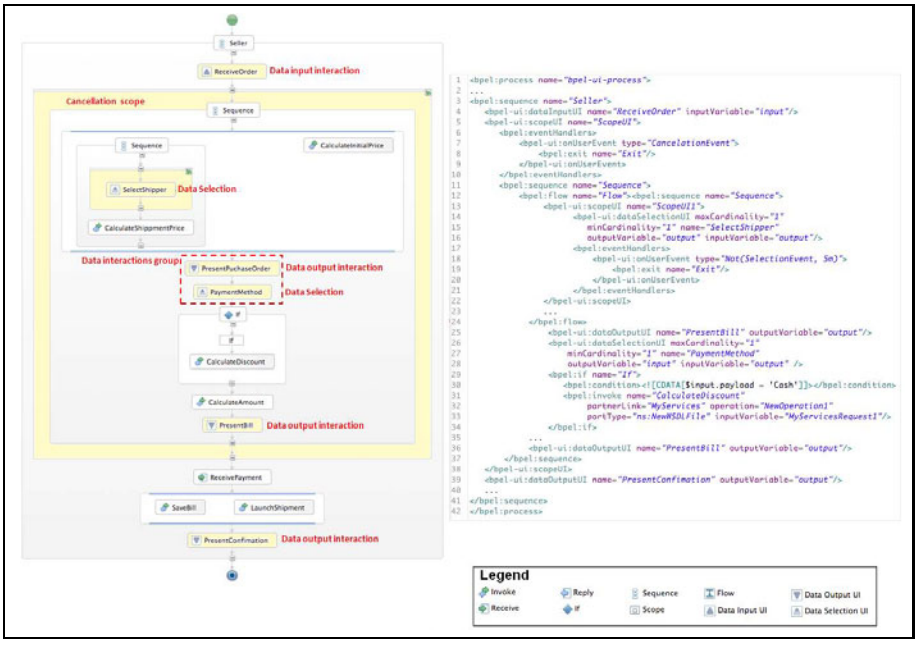

Fig. 4. Purchase Order Process expressed in UI-BPEL

Figure 4 depicts the purchase order process expressed using both graphical representation (extended Eclipse BPEL Designer graphical notations [2]) and XML representation of UI-BPEL. The figure shows that UI-BPEL supports the user interaction types required by the scenario. For example:

- UI-BPEL expresses the fact that customer needs data input interaction to provide the data order by using the *DataInputUI* activity (Figure 4, line 4). This activity launches the process execution when the input is provided;
- UI-BPEL expresses the fact that the customer can select one shipper by using *DataSelectionUI* (Figure 4, line 14). In order to process this data interaction, the composed event *Not (onShipperSelection, 5 min)* is listened. So that, if no shipper is selected in 5 minutes, then the process is cancelled (Figure 4, line 18);
- UI-BPEL expresses the fact that the customer can cancel the request for an purchase order by using a *ScopeUI* with an *OnCancelEvent* (Figure 4, line 5-11);
- The *DataOutputUI* "presenting an order" and The *DataSelectionUI* "PaymentMethod" can be gathered on the same user interface component so that the event *Conjunction (OnDataOutput, on OnDataSelection)* will be raised. This event notifies that user has confirmed the order (*OnDataOutput*) and leads to unblock the process that waits for a payment method selection (*OnDataSelection*).

4 UI-BPEL Designer Tool

This section describes the UI-BPEL Designer Tool that is dedicated to edit a UI-BPEL process that conforms to the UI-BPEL Meta-model. The tool is an Eclipse plug-in based on the Eclipse BPEL Designer [2]. The Eclipse BPEL Designer is chosen since it has an extensible framework that allows not only the WS-BPEL 2.0 specification, but also the deployment and the execution of BPEL processes from the tool into a BPEL engine. Figure 5 shows a screenshot of the UI-BPEL Designer Tool.

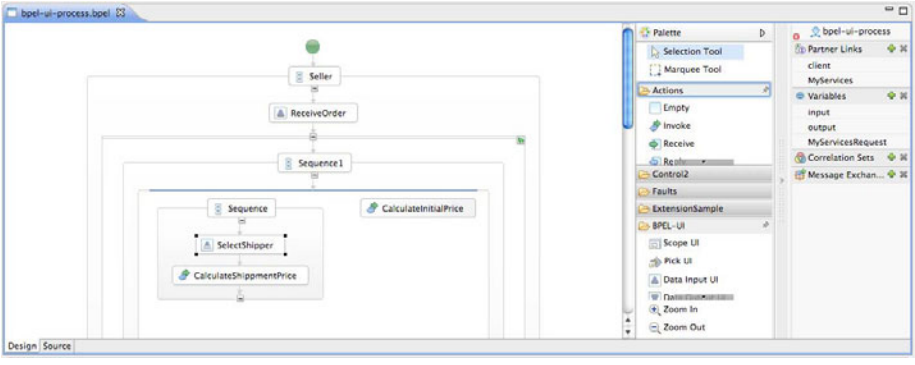

Fig. 5. UI-BPEL Designer Tool

The tool is available at the following address: http://webapps.fundp.ac.be/wse

5 Conclusion and Discussions

In this paper, we presented a BPEL extension, called the UI-BPEL. This extension addresses the user interaction in a Web service composition by defining new data interaction activities to allow user providing, selecting or getting data. The extension also proposes new types of interaction events to react when user interaction is carried out. The main goal of UI-BPEL is to allow the generation of a user interface for the Web service composition based on the described user interactions. This generation can be done in two steps:

- ***Step1*:** an abstract user interface [7] is generated from a UI-BPEL model. This user interface is described independent of any interaction modality (e.g. graphical modal, vocal modal) and computing platform (e.g. PC, smart phone). For this reason, each specified user interaction will be transformed into an abstract component, which can be an input abstract component, a selection abstract component, an output abstract component or a control abstract component. A control abstract component allows to process one or a set of data interactions. It also allows to model action interaction (e.g. cancel action). In addition, the control abstract component is also responsible of triggering the interaction events.

– ***Step2*:** a concrete user interface [7] is generated from the abstract user interface. The concrete user interface is adapted to a specific user context (user preference, user environment, and user platform). For example, for a visually handicapped person, the output abstract component will be transformed to vocal output. The concrete user interface takes also into account a set of user interface usability criterions. For example, the interface should respect the size of the device screen.

Our future work includes the development of two transformation methods for the two steps of the user interface generation described above.

References

1. Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., Yan, L.: From people to services to ui: Distributed orchestration of user interfaces. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 310–326. Springer, Heidelberg (2010)
2. Eclipse, B.: Project. Eclipse BPEL Designer (2011), `http://www.eclipse.org/bpel/`
3. Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., Trickovic, I.: Ws-bpel extension for people–bpel4people. Joint White Paper, IBM and SAP (2005)
4. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: USIXML: A language supporting multi-path development of user interfaces. In: Bastide, R., Palanque, P.A., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 200–220. Springer, Heidelberg (2005)
5. OASIS, B.: Web Services Business Process Execution Language (2007)
6. Rao, J., Su, X.: A survey of automated web service composition methods. In: Cardoso, J., Sheth, A. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 43–54. Springer, Heidelberg (2005)
7. Seffah, A., Gulliksen, J., Desmarais, C.M. (eds.): Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle. HCI Series, vol. 8 ch. 6, pp. 109–140. Springer, Heidelberg (2005)
8. Tofan, S., Pradais, A., Buraga, S.: A study regarding the abstract specification of the user interface by using USIXML and UIML languages. Romanian Journal of Human-Computer Interaction (RoCHI 2009), 31–34 (2009)
9. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: Proceedings of the 2006 ACM SIGMOD 2006, pp. 407–418 (2006)
10. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Trans. Software Eng. 30(5), 311–327 (2004)