Годишник на секция "Информатика" **Annual of "Informatics" Section**
Съюз на учените в България **Union of Scientists in Bulgaria**
Том 6, 2013, 175–184 **Volume 6, 2013, 175–184**

WS-BPEL ЗА СПЕЦИФИКАЦИЯ НА БИЗНЕС ПРОЦЕСИ

Владимир Димитров

Факултет по математика и информатика
Софийски университет „Свети Климент Охридски"
e-mail: cht@fmi.uni-sofia.bg

Резюме: Представена е XML-базираната нотация на WS-BPEL като средство за детайлна спецификация на бизнес процеси. Представени са основните елементи на нотацията. Примерите са приведени и във вид на диаграми от архитектурата за компоненти на услугите (*Service Component Architecture*), тъй като XML нотацията затруднява възприятието. Разгледани са основните елементи на нотацията.WS-BPEL е основно средство за изграждане на композитни уеб услуги, чрез които се представят бизнес процесите. WS-BPEL е език за интерпретация на бизнес процеси в машините на основните доставчици на такъв вид софтуер.
Ключови думи: Бизнес процеси, WS-BPEL.

1. Предназначение

WS-BPEL е предназначен за описание поведението на интерфейсите на услугите и за изпълнение на процеси, реализирани на услуги. Абстрактният процес (поведенческият интерфейс) е поведенческа спецификация на група услуги и включва ограничения в реда за обмен на съобщения между услугите. Изпълнимият процес определя реда на изпълнение в набора от дейности (основно комуникационни дейности); партньорите, участващи в процеса; обменяните съобщения; и обработката на събития и изключения.

2. Елементи

Спецификацията на WS-BPEL процеса съдържа набор от дейности. Последните са разпределени в две категории: основни и структуриращи. Основните дейности са атомарни. Това са:

- *invoke* – извиква операция на Уеб услуга;
- *receive* – очаква съобщение от външен партньор;
- *reply* – отговаря на външен партньор;
- *assign* – присъединява стойност на променлива;
- *throw* – известява за грешка при изпълнението;

- *compensate* – отменя ефекта на вече изпълнени дейности;
- *exit* – прекратява изпълнението на екземпляра на услугата;
- *empty* – не прави нищо.

Структуриращите дейности задават ограничения върху поведението на изпълнението на съдържащите се в тях дейности:

- *sequence* – задава последователно изпълнение;
- *switch* – превключва реда на изпълнението по алтернативни пътища;
- *pick* – управлява състезанието по получаване на събития и тези по таймер;
- *while* – организира цикъл;
- *scope* – групира дейностите в блок с единна обработка на изключения, аварии и компенсации.

Структуриращите дейности могат да бъдат влагани една в друга и да бъдат комбинирани по произволен начин. Това позволява да бъдат изграждани сложни WS-BPEL процеси.

Конструкциите *sequence*, *pick* и *while* се използват за представяне на зависимостите на структурирания поток на управление. Освен тях, в WS-BPEL със същата цел се използва конструкцията на контролните връзки, която заедно с нотацията на условието за съединение и условието за преход, определят реда, синхронизацията и условните зависимости на дейностите, обхванати от тази конструкция.

Наличието на контролна връзка от дейността А към дейността Б, означава че Б може да започне изпълнението си само след завършването на А или след нейното отклонение. Освен това, Б може да бъде изпълнена, само ако е истинно нейното условие за съединение, иначе Б се отклонява. Условието за съединение се задава с маркери, които се пренасят по контролните връзки – тези, които водят към Б. Тези маркери могат да имат положителна стойност (истина) или отрицателна стойност (лъжа). Дейността Х генерира положителен маркер в изходящата връзка С, ако Х е изпълнена (не е отклонена) и условието за преход, асоциирано с връзката С се изчислява до истина.

Условията на преход са логически изрази върху променливите на процеса (както и условието на дейността *switch*). Процес (обработка), който разпространява положителни и отрицателни маркери в контролните връзки, и който определя изпълнението или отклонението на дейности, се определя като „отстранител на мъртви хватки“.

Контролните връзки могат да пресичат границите на почти всички структуриращи дейности. При това, те не бива да създават циклични контролни зависимости и не трябва да пресичат границите на дейностите *while* и сериализуем *scope*. Взаимодействието между структуриращите дейности и контролните връзки не се възприемат добре от хората и това води до противоречиви WS-BPEL спецификации.

Независимо от това, че конструкциите на потока на управление в WS-BPEL са разработени така, че да не може процесът да влезе в мъртва хватка, някои структуриращи конструкции (например *switch* и *pick*) заедно с контролните връзки могат да доведат до това, че някои дейности да са недостижими.

Друга група от конструкции на потока на управление в WS-BPEL са обработчиците на събития, аварии и компенсации.

Обработчикът на събитие е правило от вида събитие-действие в даден обхват. Обработчикът на събитие е активен, когато неговият обхват се изпълнява, тогава

обработчикът може да се изпълнява заедно с основната дейност на обхвата. Когато се появи събитие, асоциирано с обработчика (това може да бъде събитие по таймер или получаване на съобщение) започва изпълнението на тялото на обработчика и продължава изпълнението на основната дейност на обхвата.

Някои аварии могат явно да бъдат изхвърлени с дейността *throw*. Обработчиците на аварии не се изпълняват конкурентно с основната дейност на обхвата. Основата дейност на обхвата се прекратява преди да започне изпълнението си обработчика на авариятa.

Обработчиците на компенсации са свързани с дейността compensateи позволяват процесът да отмени работата на вече завършен обхват. Когато дейността се изпълнява в даден обхват, тя активира обработчика на компенсации в този обхват, ако там има такъв. Това може да включва изпълнението на обработчици на компенсации, които са асоциирани само под обхватите на разглеждания обхват.

За повече детайли [1].

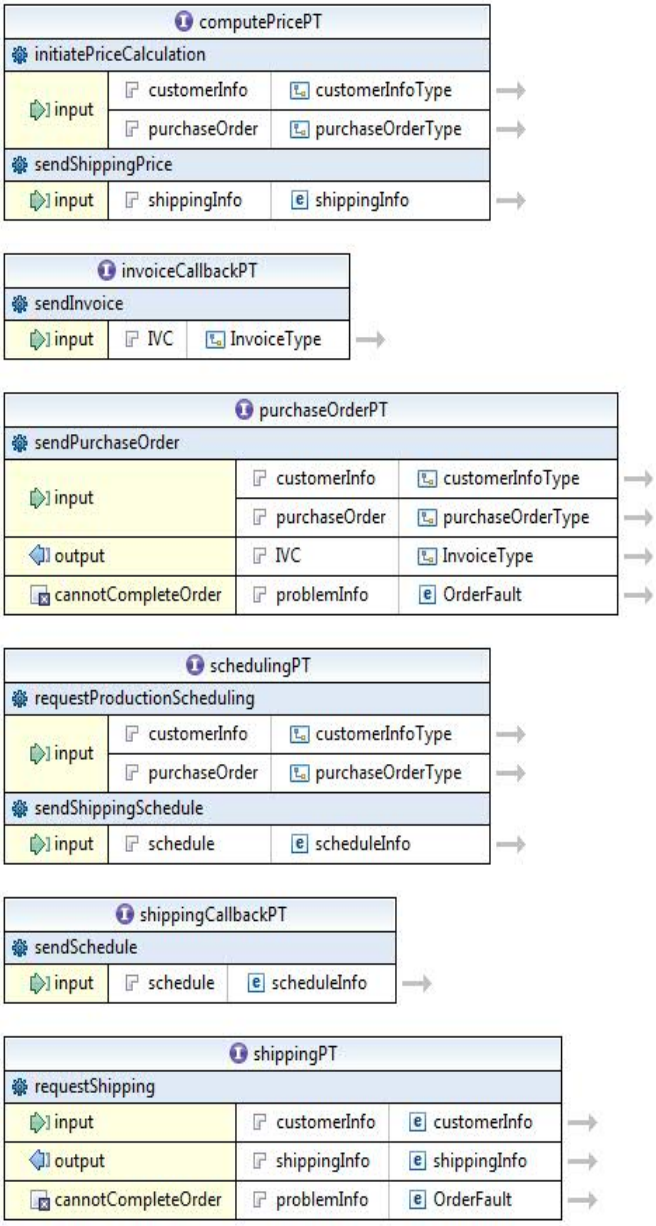
3. Пример

WSDL определенията, използвани в примера:

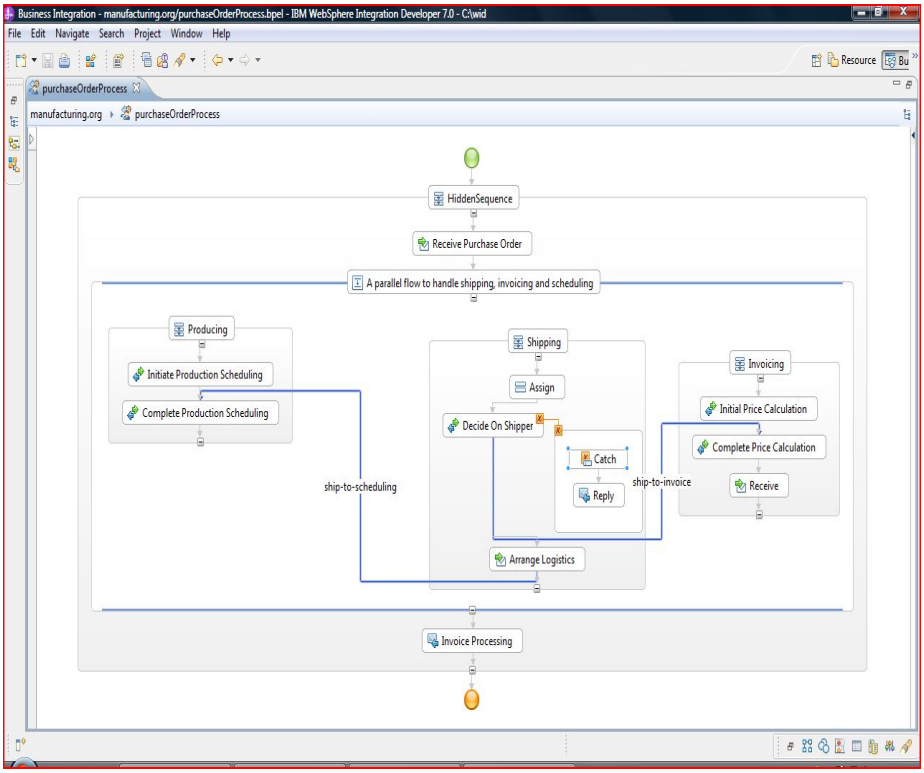
```
<wsdl:definitions
targetNamespace=
    "http://manufacturing.org/wsdl/purchase"
xmlns:sns="http://manufacturing.org/xsd/purchase"
xmlns:pos="http://manufacturing.org/wsdl/purchase"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:plnk=
    "http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
<xsd:schema>
<xsd:import
namespace="http://manufacturing.org/xsd/purchase"
schemaLocation=
          "http://manufacturing.org/xsd/purchase.xsd" />
</xsd:schema>
</wsdl:types>
<wsdl:messagename="POMessage">
<wsdl:partname="customerInfo"
type="sns:customerInfoType" />
<wsdl:partname="purchaseOrder"
type="sns:purchaseOrderType" />
</wsdl:message>
<wsdl:messagename="InvMessage">
<wsdl:partname="IVC" type="sns:InvoiceType" />
</wsdl:message>
<wsdl:messagename="orderFaultType">
<wsdl:partname="problemInfo"
element="sns:OrderFault" />
</wsdl:message>
```

```
<wsdl:messagename="shippingRequestMessage">
<wsdl:partname="customerInfo"
element="sns:customerInfo" />
</wsdl:message>
<wsdl:messagename="shippingInfoMessage">
<wsdl:partname="shippingInfo"
element="sns:shippingInfo" />
</wsdl:message>
<wsdl:messagename="scheduleMessage">
<wsdl:partname="schedule"
element="sns:scheduleInfo" />
</wsdl:message>
<!-- portTypessupportedbythepurchaseorder
process -->
<wsdl:portTypename="purchaseOrderPT">
<wsdl:operationname="sendPurchaseOrder">
<wsdl:inputmessage="pos:POMessage" />
<wsdl:outputmessage="pos:InvMessage" />
<wsdl:faultname="cannotCompleteOrder"
message="pos:orderFaultType" />
</wsdl:operation>
</wsdl:portType>
<wsdl:portTypename="invoiceCallbackPT">
<wsdl:operationname="sendInvoice">
<wsdl:inputmessage="pos:InvMessage" />
</wsdl:operation>
</wsdl:portType>
<wsdl:portTypename="shippingCallbackPT">
<wsdl:operationname="sendSchedule">
<wsdl:inputmessage="pos:scheduleMessage" />
</wsdl:operation>
</wsdl:portType>
<!-- portTypesupportedbytheinvoiceservices -->
<wsdl:portTypename="computePricePT">
<wsdl:operationname="initiatePriceCalculation">
<wsdl:inputmessage="pos:POMessage" />
</wsdl:operation>
<wsdl:operationname="sendShippingPrice">
<wsdl:inputmessage="pos:shippingInfoMessage" />
</wsdl:operation>
</wsdl:portType>
<!-- portTypesupportedbytheshippingservice -->
<wsdl:portTypename="shippingPT">
<wsdl:operationname="requestShipping">
<wsdl:inputmessage="pos:shippingRequestMessage" />
<wsdl:outputmessage="pos:shippingInfoMessage" />
<wsdl:faultname="cannotCompleteOrder"
message="pos:orderFaultType" />
</wsdl:operation>
</wsdl:portType>
<!-- portTypesupportedbytheproduction
schedulingprocess -->
<wsdl:portTypename="schedulingPT">
```

```
<wsdl:operationname="requestProductionScheduling">
<wsdl:inputmessage="pos:POMessage" />
</wsdl:operation>
<wsdl:operationname="sendShippingSchedule">
<wsdl:inputmessage="pos:scheduleMessage" />
</wsdl:operation>
</wsdl:portType>
<plnk:partnerLinkTypename="purchasingLT">
<plnk:rolename="purchaseService"
portType="pos:purchaseOrderPT" />
</plnk:partnerLinkType>
<plnk:partnerLinkTypename="invoicingLT">
<plnk:rolename="invoiceService"
portType="pos:computePricePT" />
<plnk:rolename="invoiceRequester"
portType="pos:invoiceCallbackPT" />
</plnk:partnerLinkType>
<plnk:partnerLinkTypename="shippingLT">
<plnk:rolename="shippingService"
portType="pos:shippingPT" />
<plnk:rolename="shippingRequester"
portType="pos:shippingCallbackPT" />
</plnk:partnerLinkType>
<plnk:partnerLinkTypename="schedulingLT">
<plnk:rolename="schedulingService"
portType="pos:schedulingPT" />
</plnk:partnerLinkType>
</wsdl:definitions>
```



Фигура 1. Услугите използвани в диаграма на IBM Rational Software Architect



Фигура 2. WS-BPEL кодът представен с диаграма на IBM WebSphere Integration Developer

WS-BPEL код:

```
<processname="purchaseOrderProcess"
targetNamespace="http://example.com/ws-bp/purchase"
xmlns="http://docs.oasis-open.org/wsbpel/2.0/
process/executable"
xmlns:lns="http://manufacturing.org/wsdl/purchase">
<documentationxml:lang="EN">
A simple example of a WS-BPEL process for handling a purchase order.
</documentation>
<partnerLinks>
<partnerLinkname="purchasing"
partnerLinkType="lns:purchasingLT"
myRole="purchaseService" />
<partnerLinkname="invoicing"
partnerLinkType="lns:invoicingLT"
myRole="invoiceRequester"
partnerRole="invoiceService" />
<partnerLinkname="shipping"
partnerLinkType="lns:shippingLT"
```

```
myRole="shippingRequester"
partnerRole="shippingService" />
<partnerLinkname="scheduling"
partnerLinkType="lns:schedulingLT"
partnerRole="schedulingService" />
</partnerLinks>
<variables>
<variablename="PO" messageType="lns:POMessage" />
<variablename="Invoice"
messageType="lns:InvMessage" />
<variablename="shippingRequest"
messageType="lns:shippingRequestMessage" />
<variablename="shippingInfo"
messageType="lns:shippingInfoMessage" />
<variablename="shippingSchedule"
messageType="lns:scheduleMessage" />
</variables>
<faultHandlers>
<catchfaultName="lns:cannotCompleteOrder"
faultVariable="POFault"
faultMessageType="lns:orderFaultType">
<replypartnerLink="purchasing"
portType="lns:purchaseOrderPT"
operation="sendPurchaseOrder"
variable="POFault"
faultName="cannotCompleteOrder" />
</catch>
</faultHandlers>
<sequence>
<receivepartnerLink="purchasing"
portType="lns:purchaseOrderPT"
operation="sendPurchaseOrder"
variable="PO"
createInstance="yes">
<documentation>
ReceivePurchaseOrder
</documentation>
</receive>
<flow>
<documentation>
A parallel flow to handle shipping, invoicing and scheduling
</documentation>
<links>
<linkname="ship-to-invoice" />
<linkname="ship-to-scheduling" />
</links>
<sequence>
<assign>
<copy>
<from>$PO.customerInfo</from>
<to>$shippingRequest.customerInfo</to>
</copy>
</assign>
```

```
<invokepartnerLink="shipping"
portType="lns:shippingPT"
operation="requestShipping"
inputVariable="shippingRequest"
outputVariable="shippingInfo">
<documentation>
DecideOnShipper
</documentation>
<sources>
<sourcelinkName="ship-to-invoice" />
</sources>
</invoke>
<receivepartnerLink="shipping"
portType="lns:shippingCallbackPT"
operation="sendSchedule"
variable="shippingSchedule">
<documentation>
Arrange Logistics
</documentation>
<sources>
<sourcelinkName="ship-to-scheduling" />
</sources>
</receive>
</sequence>
<sequence>
<invokepartnerLink="invoicing"
portType="lns:computePricePT"
operation="initiatePriceCalculation"
inputVariable="PO">
<documentation>
InitialPriceCalculation
</documentation>
</invoke>
<invokepartnerLink="invoicing"
portType="lns:computePricePT"
operation="sendShippingPrice"
inputVariable="shippingInfo">
<documentation>
CompletePriceCalculation
</documentation>
<targets>
<targetlinkName="ship-to-invoice" />
</targets>
</invoke>
<receivepartnerLink="invoicing"
portType="lns:invoiceCallbackPT"
operation="sendInvoice"
variable="Invoice" />
</sequence>
<sequence>
<invokepartnerLink="scheduling"
portType="lns:schedulingPT"
operation="requestProductionScheduling"
```

```
inputVariable="PO">
<documentation>
InitiateProductionScheduling
</documentation>
</invoke>
<invokepartnerLink="scheduling"
portType="lns:schedulingPT"
operation="sendShippingSchedule"
inputVariable="shippingSchedule">
<documentation>
CompleteProductionScheduling
</documentation>
<targets>
<targetlinkName="ship-to-scheduling" />
</targets>
</invoke>
</sequence>
</flow>
<replypartnerLink="purchasing"
portType="lns:purchaseOrderPT"
operation="sendPurchaseOrder"
variable="Invoice">
<documentation>InvoiceProcessing</documentation>
</reply>
</sequence>
</process>
```

4. Заключение

В този материал е направен кратък обзор на основните конструкции на WS-BPEL без да се навлиза в излишни подробности. Примерът и диаграмите нагледно илюстрират изпълнимите процеси в WS-BPEL. Направеният тук обзор е отправна точка за по-нататъшните изследвания върху нотацията.

Благодарности

Изследванията, представени в този доклад са по проекта „Съвременни езици, среди и технологии за програмиране и прилагането им при подготовка на софтуерни специалисти“ финансиран от Фонд „Научни изследвания“ по Национален конкурс "Финансиране на фундаментални научни и научно-приложни изследвания в приоритетните области" – 2012г., ДФНИ-И01/12.

Литература

[1] OASIS, Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html