Годишник на секция "Информатика" **Annual of "Informatics" Section**
Съюз на учените в България **Union of Scientists in Bulgaria**
Том 6, 2013, 185–191 **Volume 6, 2013, 185–191**

WS-CDL ЗА СПЕЦИФИКАЦИЯ НА БИЗНЕС ПРОЦЕСИ

Владимир Димитров

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

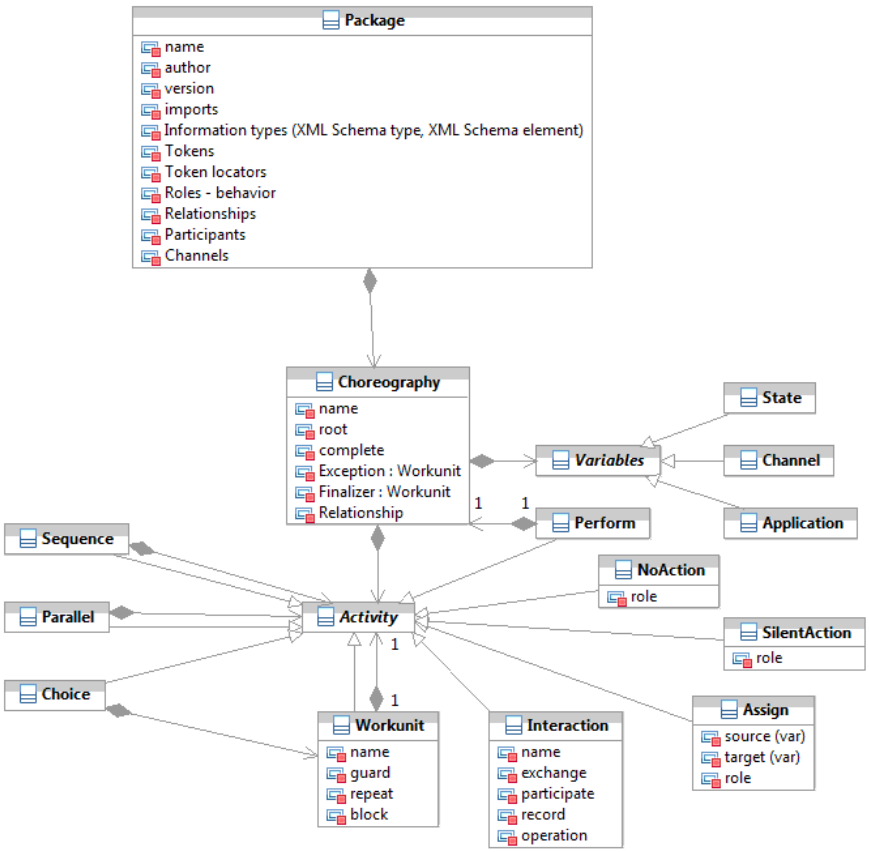
Резюме: WS-CDL е средство за спецификация на взаимодействието между няколко уеб услуги. С него се описва взаимодействието от най-високо без да се навлиза в детайли по реализацията. WS-CDL за сега е единствената спецификация, преминала успешно процедурите на Международната организация по стандартизация. Представени са основните конструктивни елементи – дейностите на хореографията. Формализацията на средството е доведено до край, което позволява автоматизация на изследването на спецификациите написани на WS-CDL.
Ключови думи: Бизнес процеси, WS-CDL.

1. Предназначение

Структурата на WS-CDL, от концептуална гледна точка, е представена на Фиг. 1.

Хореографията с WS-CDL е глобално и безпристрастно описание на взаимодействието на между няколко участника. Уеб услугите се разработват от различни доставчици и е необходимо да се представи съвместното им използване, което не може да стане с WSDL. Работната група W3C Choreographye създадена да разработи средства за описание на съвместното използване на уеб услуги за постигане на зададена цел.

Назначението на WS-CDL спецификациите е да описва контрактите между участниците, като се представя наблюдаваното отвън поведение на Уеб услугите и на техните клиенти (най-често това са други уеб услуги), и да описва обмена на съобщения между тях. По идея, би трябвало от описанието на WS-CDL хореографията да се генерира скелетът на код за уеб услуги или абстрактни WS-BPEL процеси.



Фигура 1. Пакетът на WS-CDL.

2. Елементи

Описанието на хореографията е пакет, който е контейнер на набора от дейности, изпълнявани от участниците. Основните типове дейности са: дейностите на потока на управление (*control flow*), работните единици (*work unit*) и основните дейности (*basic activities*).

Първата категория включва дейностите *Sequence*, *Parallel* и *Choice*. Дейността *Choice* избира изпълнението на някоя дейност от определено множество конкурентни дейности. Тази дейност извършва два вида избори:

- Избор, управляван от данните (непосредствен избор). В този случай, изборът се извършва с изчисляване на логически условия върху променливите за данни. Изчисляването се извършва в момента, когато дейността *Choice* получи управлението.
- Избор, управляван от събития (отложен избор). Изборът в този случай се определя от сбъдването на някое събитие от определено множество конкурентни

събития. Събитието може да бъде генерирано от изменението на някои променливи в резултат на взаимодействие или изпълнение на някои дейности от хореографията. Изменението на променливите води до преизчисляване на условията. Изборът се отлага когато дейността *Choice* поема управлението и изпълнението спира в очакване на събития.

Видът на дейността *Choice* се определя от природата на дейностите от множеството на избора. Ако всички дейности имат защитни неблокиращи условия, тогава изборът се управлява от данните. Иначе, изборът може изцяло да се управлява от събитията или да е комбинация и от двата вида. По това WS-CDL се различава от WS-BPEL, където има само два вида избор в отделните конструкции (*Switch* и *Pick*).

Вторият вид дейност в WS-CDL е *WorkUnit*. Тази дейност описва условно или възможно циклично изпълнение на някоя дейност. Нейният синтаксис има няколко части, включващи референции към изпълнявана дейност, защитно условие на блока и условие за цикъл. Условията са логически и определят колко пъти ще се изпълни дейността (възможно е и нито веднъж). Изпълнението на дейността *WorkUnit* започва с пресмятане на защитното условие. Ако резултатът е „истина“, тогава дейността се изпълнява, в противен случай се пропуска. След изпълнението на дейността се пресмята условието на цикъла и в зависимост от получения резултат се изпълнява дейността или не. С влагането на дейности *WorkUnit* може да се организира влагане на цикли. Условието на блока се използва за задържане на пресмятането на защитното условие и условието на цикъла докато не станат достъпни всички използвани от него променливи. Обикновено, *WorkUnit* се използва заедно с дейността *Choice* и в този случай изборът може да е непосредствен или отложен в зависимост от условието на блока на *WorkUnit*, който е обхванат от дейността Choice.

Дейностите *Sequence*, *Parallel*, *Choice* и *WorkUnit* са типични конструкции от процедурните езици за програмиране. Тези дейности съответстват на дейностите *Sequence*, *Flow*, *While* и *Pick* на WS-BPEL. Изобразяването на *Choice* и *WorkUnit* не е тривиално, тъй като трябва да се направи избор между *Switch* и *Pick* по условието на блока.

Третият вид дейности в WS-CDL са основните дейности: *Interaction*, *NoAction*, *SilentAction*, *Assign* и *Perform*.

Дейностите *NoAction* и *SilentAction* задават тези места в хореографията, където се изпълнява някоя именувана роля или се изпълнява някакво „задкулисно“ действие, което не влияе върху хореографията.

Дейността *Assign* се използва за задаване стойност на променлива, например от една променлива на друга, когато и двете променливи са в една и съща роля.

Дейността *Perform* се използва за извикване на друга хореография за изпълнение в контекста на текущата. Извикваната хореография може да започне от пакета, който я извиква, или в съвсем друг пакет, импортиран в текущия пакет. Дейността *Perform* има механизми за обвързване на променливите на извикваната хореография с променливите на викащата хореография.

Най-важният елемент на WS-CDL е дейността *Interaction*. Тази дейност описва обмена на информация между участниците, като задава получателя. Взаимодействията могат да бъдат следните: заявка (*request*), отговор (*respond*) или заявка-отговор (*request-respond*). Описанието на взаимодействието се състои три части: участници, обменяна информация и канал за обмен на информацията. По-долу е

приведен пример за взаимодействие. Трябва да се отбележи, че описанието е асиметрично – основно внимание се отделя на получателя. По-точно, WS-CDL описва какви операции се предприемат за получаване на информация, а не какво се прави преди нейното изпращане.

```
<?xmlversion="1.0" encoding="utf-8"?>
<interactionname="QuoteResponse" initiate="false" align="false"
       operation="ReceiveQuote"
       channelVariable="receiveQuoteChannel">
<participatetoRole="customer" fromRole="supplier"
   relationship="GetQuote" />
<exchangename="e2" action="respond"
   informationType="quoteDocument">
<sendvariable="quote" />
<receivevariable="quote" />
</exchange>
</interaction>
```

Ролите *from* (изпращач), *to* (получател) и отношенията между тях явно се задават в раздела *participate*. Независимо от това, че отношението е зададено с ролите, за някое поведение е възможно операцията, която се изпълнява за приемане на информация, да е зададена на друго място в определението на взаимодействието, а не в раздела *participate*.

Информацията, която се изпраща и получава, се задава с променливи. Всяка променлива описва своите елементи *informationType* и *recordReference*. Променливите съдържат три вида информация: приложна информация, информация за състоянието, канална информация. Съдържанието на променливите е достъпна чрез функциите на WS-CDL разширението с XPath 1.0, например *setVariable* (*name*, *path*, *role*).

Променливите от канален тип задават роля на получателя (при съобщенията от вида заявка или отговор) и поведението на получателя. Поведението на ролята може да се задава и с WSDL интерфейси:

```
<channelTypename= "requestQuoteChannel" action= "request"
usage= "unlimited">
    <roletype= "supplier" behavior= "ReceiveRFQ"/>
    <reference>
          <tokenname= "supplierRef"/>
    </reference>
    <identity>
          <tokenname= "quoteId"/>
    </identity>
    <passingaction= "respond" new= "true"
channel= "receiveQuoteChannel"/>
</channelType>
<channelTypename= "receiveQuoteChannel" action= "respond"
usage= "once">
    <roletype= "customer" behavior= "ReceiveQuote"/>
    <reference>
          <tokenname= "customerRef"/>
    </reference>
    <identity>
          <tokenname= "quoteId"/>
    </identity>
</channelType>
```

Каналите могат да са двустранни или едностранни. Всеки канал е от вида заявка, отговор или заявка-отговор. По принцип, връзката между тези три вида канали и *WSDLMessageExchangePatterns* се нуждае от доработка.

Каналите са връзката между хореографиите и WSDL интерфейсите. Всяко поведение и хореография се описва с канален тип. Променливите от канален тип имат само едно поведение. По каналите се предават променливи на състоянието, приложни променливи и канални променливи. В описанието на каналния тип се задават имената на каналните типове, които могат да бъдат предавани в канала от разглеждания тип.

Маркерът *reference* се задава за задаване на получателя, а маркерът *identity* – за корелация на обменяните съобщения. Маркерът се представя като информационен тип. Локаторите на маркери дават възможност да се намират маркерите с XPath изрази.

Маркерите и информационните типове се задават на ниво пакет и са достъпни за всички хореографии и дейности на пакета. Информационните типове могат да са XMLSchema елементи, WSDL 1.1 типове, съобщения, XMLSchema типове или WSDL 1.0 типове съобщения. Всяка променлива има определен тип. Променливите, които се използват от дейностите на хореографията, се определят на нивото на цялата хореография. Тези променливи са достъпни за съвместно използване с извикваните хореографии като се прилага механизма на обвързване на дейностите *perform*.

Променливите могат да принадлежат едновременно на няколко роли от хореографията. Предаването на стойностите на променливите между ролите може да става със съобщения на дейността на взаимодействието:

```
<informationTypename= "address"type= "xsd:anyURI"/>
<informationTypename= "correlationId"type= "xsd:string"/>
<informationTypename= "quoteRequest"type= "xsd:anyURI"/>
<informationTypename= "quoteDocument"type= "xsd:anyURI"/>
<tokenname= "customerRef"informationType= "address"/>
<tokenname= "supplierRef"informationType= "address"/>
<tokenname= "quoteId"informationType= "correlationId"/>
<tokenLocatortokenName= "quoteId"query= "quoteRequest/docId"
       informationType= "quoteRequest">
</tokenLocator>
```

Описанието на хореографията съдържа дейности от най-високо ниво. Този контейнер може да има обработчик на изключения и компенсатор. Обработчикът на изключения се активира, когато възникне изключение. Компенсаторът се активира, когато разглежданата хореография е завършила работата си в контекста на някоя друга хореография, и в последната е възникнала компенсация, тогава компенсаторът отменя ефекта от работата на разглежданата хореография.

```
<choreographyname= "QuoteAndOrder"isolation= "dirty-write"
       root= "true">
       <relationshiptype= "GetQuote"/>
       <relationshiptype= "PlaceOrder"/>
       <variableDefinitions>
              <variablename= "rfq"mutable= "false"free= "false"
                     informationType= "quoteRequest"
                     silentAction= "false"/>
              <variablename= "quoteForm"mutable= "true"
                     free= "false"informationType= "quoteDocument"
```

```
                    silentAction= "true"role= "supplier"/>
             <variablename= "quote"mutable= "true"free= "false"
                    informationType= "quoteDocument"
                    silentAction= "false"/>
      </variableDefinitions>
      <sequence> ... activities ... </sequence>
      <exceptionname= "NoAction">
             <workunitblock= "false"repeat= "false"
                    name= "NoAction"guard= "true">
                    <noActionrole= "customer"/>
             </workunit>
      </exception>
      <finalizername= "None">
             <workunitblock= "false"repeat= "false"name= "None"
                    guard= "true">
                    <noActionrole= "customer"/>
             </workunit>
      </finalizer>
</choreography>
```

Пакетът съдържа няколко хореографии и се използва за обединяване на общата информация на съдържащите се в него хореографии. Пакетът описва типовете роли, които задават поведението, и типовете отношения между два типа роли. Типовете отношения задават подмножество от роли на поведение, което те проявят в разглежданото отношение.

Пакетът може да съдържа описание на типовете участници, т.е. логически да групира ролите. Всяка роля има поне една поведение. Отношението има две роли и подмножество от поведения за всяка от тези роли. Типовете участници се определят на ниво пакет, но не се използват в хореографиите и дейностите.

WS-CDL не разглежда разгръщането на статичния проект (роли, поведение, отношения и участници) по време на изпълнение на доставчиците на услуги и на конкретните услуги.

```
<packagename= "Buying"
      xmlns= "http://www.w3.org/2004/10/ws-chor/cdl/"
...
      <roleTypename= "supplier">
             <behaviorinterface= "Supplier.wsdl"name=
      "ReceiveRFQ"/>
             <behaviorinterface= "Supplier.wsdl"name=
      "ReceivePO"/>
      </roleType>
      <roleTypename= "customer">
             <behaviorinterface= "Customer.wsdl"
                    name= "ReceiveQuote"/>
             <behaviorinterface= "Customer.wsdl"
                    name= "ReceivePaymentRequest"/>
      </roleType>
      <relationshipTypename="GetQuote">
             <roletype= "supplier"behavior= "ReceiveRFQ"/>
             <roletype= "customer"behavior= "ReceiveQuote"/>
      </relationshipType>
      <relationshipTypename="PlaceOrder">
             <roletype= "supplier"behavior= "ReceivePO"/>
             <roletype= "customer"behavior=
      "ReceivePaymentRequest"/>
```

```
        </relationshipType>
        <participantTypename= "CustomerA">
              <roletype= "customer"/>
        </participantType>
        <participantTypename= "SupplierB">
              <roletype= "supplier"/>
              <roletype= "customer"/>
        </participantType>
...
</package>
```

Независимо от това, че всички елементи на WS-CDL имат уникални имена, много от тях (хореографии, дейности, роли, отношения и канали) се нуждаят от уникални идентификатори по време на изпълнение. Например, ако в хореографията се изпълнява взаимодействие с паралелна дейност за стартиране на поръчки към няколко доставчика, то трябва да има механизъм на обвързване на съобщенията-отговори с екземплярите на паралелните дейности.

3. Заключение

Тук беше представен накратко WS-CDL. Неговите конструкции са доста близки с тези на WS-BPEL, но неговото назначение е друго – да описва взаимодействието отвън, а не както WS-BPEL – отвътре. За повече информация може да се види [1].

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] W3C, Web Services Choreography Description Language Version 1.0, http://www.w3.org/TR/ws-cdl-10/
.