Język BPEL

Bussiness Process Execution Language

Język BPEL

- BPEL jest (Web Services) Business Process Execution Language, standaryzowany przez OASIS
- BPEL jest językiem bazującym na XML służącym do definiowania interakcji usług sieciowych
- **BPEL** opisuje interakcje między usługami sieciowymi tworzącymi razem przepływ pracy
  - przekazywanie pracy
  - przekazywanie danych
  - obsługa błędów

Historia

- IBM - 2001 - WSFL
- Microsoft - 2001 - XLANG
- W roku 2002 - połączenie dwóch języków w jeden: BPEL4WS
- W 2003 kilka firm (w tym IBM i Microsoft) przesyłają BPEL4WS 1.1 do standaryzacji (OASIS)
- W 2007 nowa wersja standardu BPEL4WS 2.0

Cechy języka BPEL

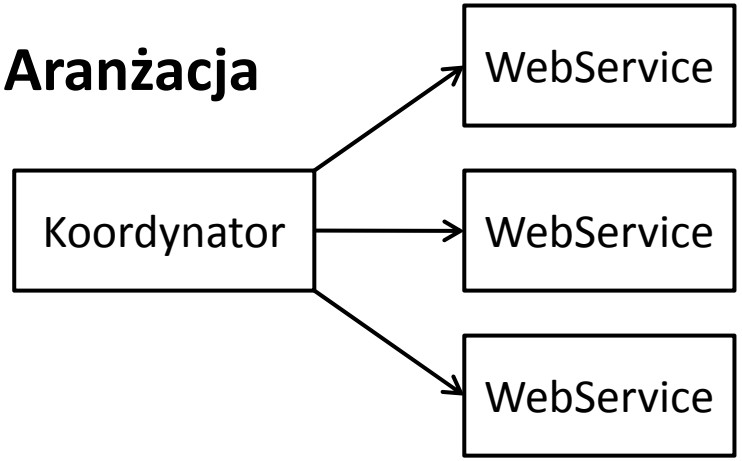
- Służy do opisu interakcji między usługami sieciowymi - Web Services, poprzez opis:
  - przepływu pracy
  - przepływu danych
  - obsługi błędów
- Zbudowany na bazie języka XML
- Wspiera architekturę SOA za pomocą aranżacji lub choreografii usług sieciowych
- Procesy zaimplementowane za pomocą języka BPEL są koordynowany przez serwer BPEL

Architektury przepływów pracy

- Aranżacja (orkiestracja) – logika procesów jest scentralizowana; elementarne aplikacje nie muszą być świadome uczestnictwa w przepływie pracy
- Choreografia – logika procesów jest rozproszona między aplikacje realizujące przepływ pracy

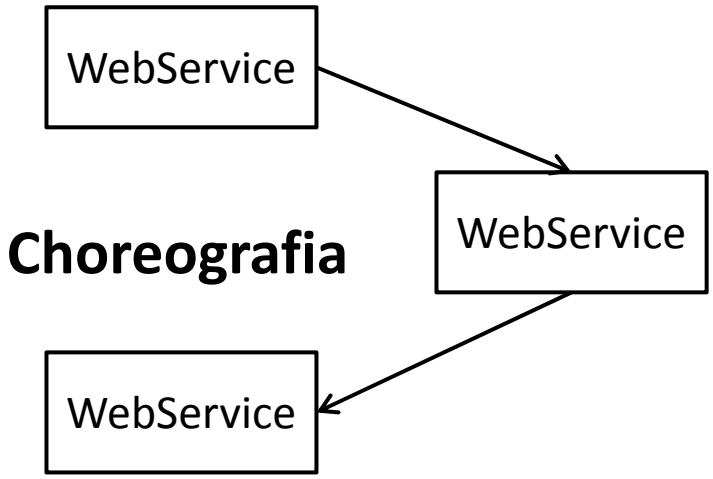
Aranżacja usług

- Aranżacji usług polega na zastosowaniu centralnego komponentu odpowiedzialnego za koordynację wywołania usług składowych
- Usługi składowe nie mają świadomości uczestnictwa w złożonym procesie biznesowym
- Aranżacja jest uważana za rozwiązanie bardziej elastyczne:
  - usługi składowe mogą uczestniczyć w złożonym procesie biznesowym bez konieczności modyfikowania ich kodu źródłowego
  - możliwa jest realizacja scenariuszy alternatywnych na wypadek awarii usługi składowej



Choreografia usług

- Nie korzysta z koncepcji centralnego koordynatora usług
- Każda z usług składowych posiada fragment wiedzy na temat realizowanego procesu biznesowego
- Komunikuje się samodzielnie z usługami zależnymi
- Konieczność dostosowywania kodu źródłowego istniejących usług do struktury procesu biznesowego
- Awaria którejkolwiek usługi składowej pociąga za sobą odłączenie usług zależnych
- Przepływ sterowania odbywa się poprzez wzajemnie wywołanie usług składowych



Własności BPEL

BPEL modeluje i implementuje podstawowe elementy procesów biznesowych

- kształt przepływów pracy: rozgałęzienia, iteracje, współbieżność
- asynchroniczną komunikację i korelację długotrwałych , zagnieżdżonych elementarnych jednostek pracy
- obsługę błędów i kompensację
- przekazywanie danych

Podstawowe konstrukcje

Główny zbiornik – znacznik **<process>**

```
<process name="PrimerProcess"
        targetNamespace="http://oasis-open.org/WSBPEL/Primer/"
        xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable" />
```

Przewidziano istnienie dwóch typów procesów:

- **Proces wykonywalny** (Executable process) – definiuje strukturę procesu i zewnętrzne interfejsy jego składowych usług. Zawiera kompletny opis procesu umożliwiający jego wykonanie:
  xmlns=http://docs.oasis-open.org/wsbpel/2.0/process/**executable**

- **Proces abstrakcyjny** (Abstract process) – nie jest kompletnie zdefiniowany. Definicja procesów abstrakcyjnych nie służy ich wykonaniu. Procesy abstrakcyjne mogą być analizowane i służyć za podstawę do implementacji różnych procesów wykonywalnych.
  xmlns=http://docs.oasis-open.org/wsbpel/2.0/process/**abstract**

Podstawowe konstrukcje

Atrybuty znacznika <*process*>:

- *targetNamepsace* – docelowa przestrzeń nazw, do której należeć będzie definiowany proces – atrybut obligatoryjny
- *name* – nazwa procesu – atrybut obligatoryjny;
- *queryLanguage* – język który jest wykorzystywany do wykonywania zapytań, domyślnym językiem jest Xpath;
- *expressionLanguage* – język wykorzystywany do wartościowania wyrażeń, domyślnym językiem jest Xpath;
- *supressJoinFailure* – flaga określająca sposób obsługi oczekiwania na przychodzące połączenia, wartość *yes* oznacza oczekiwanie do momentu pojawienia się ostatniego połączenia;
- *exitOnStandardFault* – flaga określająca sposób obsługi wyjątku, wartość *yes* oznacza przerwania wykonywania procesu w momencie pojawienia się wyjątku.

Podstawowe konstrukcje

Usługi sieciowe będące składowymi procesu, są nazywane partnerami. Do ich definiowania służy znacznik **<partnerLinks>.**

```
<partnerLinks>
        <partnerLink name="NCName"
                partnerLinkType="QName"
                myRole="NCName"?
                partnerRole="NCName"?
                initializePartnerRole="yes|no"?/>+
</partnerLinks>
```

Komunikacja między partnerami

W BPEL istnieją trzy podstawowe znaczniki odpowiedzialne za komunikacje między *partnerami* procesu: <***receive***>, <***replay***> i <***invoke***>.

- znacznik **<invoke>** - służy do wywołania usługi sieciowej partnera w celu przeprowadzenia operacji, umożliwia przekazywanie parametrów wywołania:

```
<invoke name="RequestResponseInvoke"
    partnerLink="BusinessPartnerServiceLink"
    operation="RequestResponseOperation"
    inputVariable="Input"
    outputVariable="Output" />
```

Komunikacja między partnerami

W BPEL istnieją trzy podstawowe znaczniki odpowiedzialne za komunikacje między *partnerami* procesu: <***receive***>, <***replay***> i <***invoke***>.

- znacznik **<receive>** - reprezentuje port wejściowy danej usługi sieciowej, nasłuchujący na otrzymanie żądania

```
<receive name="ReceiveRequestFromPartner"
        createInstance="yes"
        partnerLink="ClientStartUpPLT"
        operation="StartProcess" ... />
```

- znacznik **<reply>** - wykorzystywany w połączeniu z **<receive>**, służy zwróceniu komunikatu do partnera wywołującego żądanie

```
<reply name="ReplyResponseToPartner"
        partnerLink="ClientStartUpPLT"
        operation="StartProcess" ... />
```

Zmienne

BPEL pozwala definiować zmienne które mogą być wykorzystywane do kontrolowania przebiegu procesu, lub gromadzenia informacji o jego stanie. Zmienne definiowane są za pomocą znacznika **<variable>** umieszczanego na poziomie znacznika **<process>**

```
<variables>
        <variable name="myVar1" type="xsd:string" />
        <variable name="myVar2" type="xsd:string" />
        <variable name="myVar3" type="myNS:myComplexType" />
</variables>
```

Przypisywanie zmiennych odbywa się poprzez znacznik **<assign>**

```
<assign>
        <copy>
                <from variable="myVar1" />
                <to variable="myVar2" />
        </copy>
</assign>
```

Zmienne

BPEL pozwala definiować zmienne które mogą być wykorzystywane do kontrolowania przebiegu procesu, lub gromadzenia informacji o procesie. Zmienne definiowane są w znaczniku **<variable>** umieszczanym na poziomie znacznika **<process>.**

Można wykorzystywać XPath do przeliczania wartości lub przeglądania typów złożonych:

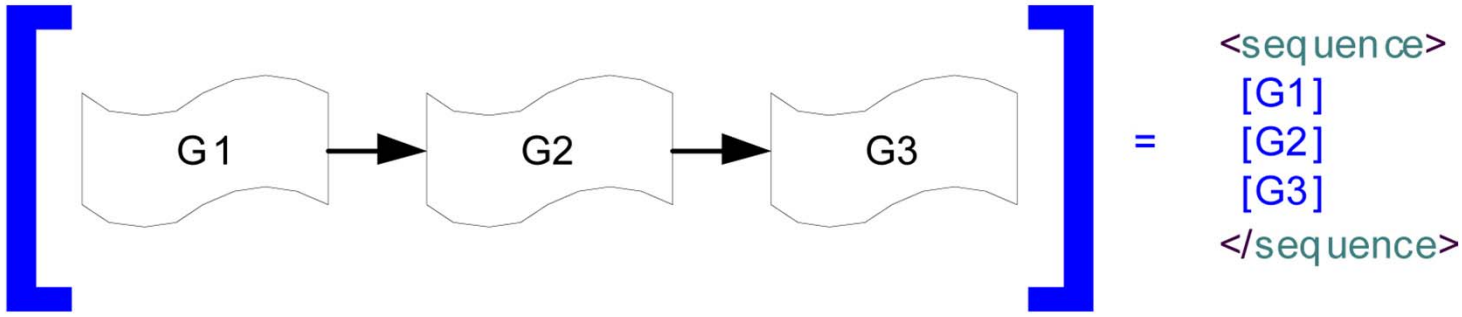
```
<from>count($zamówienia/zamówienie)</from>
<from variable="Input" part="operation1Parameter">
        <query>
                creditCardInformation
        </query>
</from>
```

Podstawowe konstrukcje sterujące

BPEL udostępnia podstawowe konstrukcje sterujące

- Znacznik <**sequence>** modeluje sekwencyjne wykonanie zbioru akcji
- Znacznik <**switch>** modeluje wybór dokładnie jednej z wielu alternatywnych akcji, na podstawie wartości wyrażeń przypisanych poszczególnym akcjom za pomocą znacznika <**case>** uzupełnionych opcjonalnym znacznikiem <**otherwise>**
- Znacznik <**while>** modeluje wielokrotne wykonanie aktywności tak długo jak skojarzone z nim wyrażenie logiczne jest spełnione
- Znacznik <**flow>** modeluje zrównoleglenie aktywności
- Podstawowa synchronizacja współbieżnych działań jest realizowana za pomocą powiązań <**link>** w równoległych wątkach

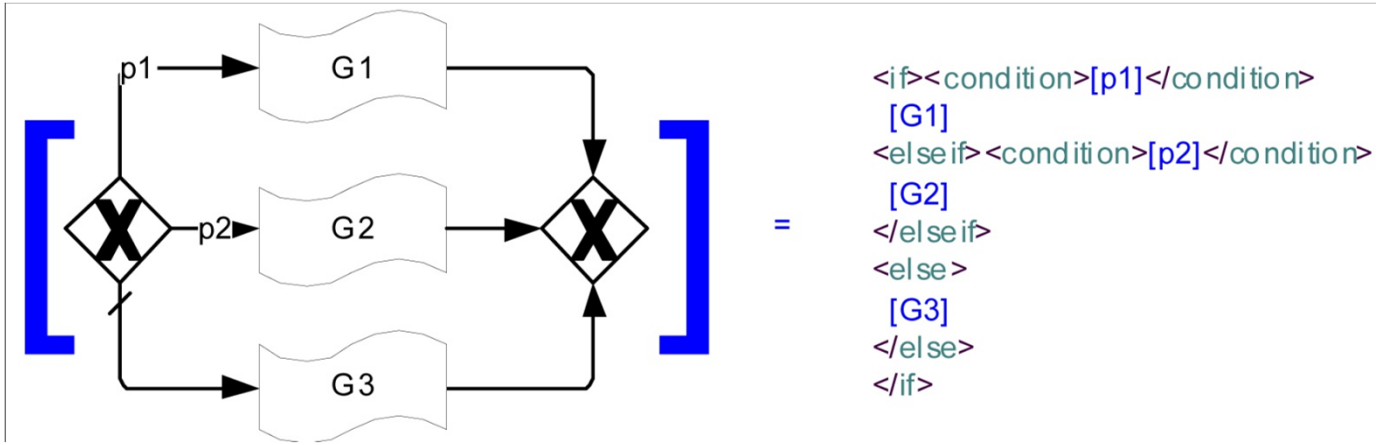
Sekwencja



- znacznik **<sequence>** - modeluje sekwencyjne wykonanie zbioru akcji. Operacje są wykonywane jedna po drugiej, w kolejności występowania na liście

```
<sequence name="InvertMessageOrder">
    <receive name="receiveOrder" ... />
    <invoke name="checkPayment" ... />
    <invoke name="shippingService" ... />
    <reply name="sendConfirmation" ... />
</sequence>
```

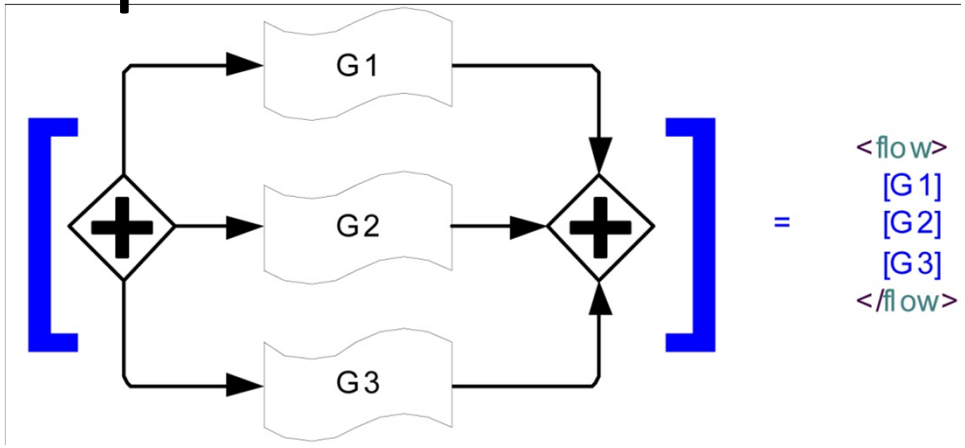
Alternatywa – bramka XOR



Znacznik **<if-else>** - modeluje rozgałęzienie w przebiegu wykonywania procesu bazując na warunkach definiowanych przy użyciu Xpath.

```
<if name="isOrderBiggerThan5000Dollars">
    <condition>  $order &gt; 5000  </condition>
    <invoke name="calculateTenPercentDiscount" ... />
    <elseif>
        <condition>  $order &gt; 2500  </condition>
        <invoke name="calculateFivePercentDiscount" ... />
    </elseif>
    <else>
        <reply name="sendNoDiscountInformation" ... />
    </else>
</if>
```

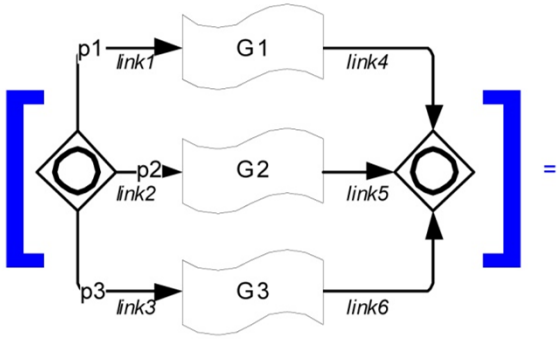
Współbieżność – bramka AND



Znacznik **<flow>** służy do zrównoleglenia wątków w procesie. Do synchronizacji połączenia równoległych wątków służy znacznik **<link>**.

```
<flow>
    <links>  <link name="checkFlight-To-BookFlight" />  </links>
    <invoke name="checkFlight" ...>
        <sources>  <source linkName="checkFlight-To-BookFlight" />  </sources>
    <invoke name="checkHotel" ... />
    <invoke name="checkRentalCar" ... />
    <invoke name="bookFlight" ...>
        <targets>
            <target linkName="checkFlight-To-BookFlight" />
        </targets>
    </invoke>
</flow>
```

Bramka OR




```
<flow>
  <links>
    <link name="[link1]"/>
    ...
    <link name="[link6]"/>
  </links>

  <empty>
    <sources>
      <source linkName="[link1]">
        <transitionCondition>[p1])</transitionCondition>
      </source>
      <source linkName="[link2]">
        <transitionCondition>[p2])</transitionCondition>
      </source>
      <source linkName="[link3]">
        <transitionCondition>[p3])</transitionCondition>
      </source>
    </sources>
  </empty>

  <flow>
    <targets><target linkName="[link1]"/></targets>
    <sources><source linkName="[link4]"/></sources>
    [G1]
  </flow>

  <flow>
    <targets><target linkName="[link2]"/></targets>
    <sources><source linkName="[link5]"/></sources>
    [G2]
  </flow>

  <flow>
    <targets><target linkName="[link3]"/></targets>
    <sources><source linkName="[link6]"/></sources>
    [G3]
  </flow>

  <empty>
    <targets>
      <target linkName="[link4]"/>
      <target linkName="[link5]"/>
      <target linkName="[link6]"/>
    </targets>
  </empty>
</flow>
```

Obsługa zdarzenia

Dotychczas omówiono jedynie znaczniki powodujące zablokowanie wykonania procesu do czasu obsługi pewnego bloku operacji. Nie zawsze pożądane jest wstrzymywanie bądź ingerowanie w główny tor wykonywania procesu. Czasami warto przenieść obsługę pewnych opertacji poza tor główny. Takie zachowanie może zostać zdefiniowane przez blok obsługi zdarzeń definiowany przez znacznik **<eventHandlers>**.

```
<process name="purchaseOrderProcess" ...>
        ...
        <eventHandlers>
                <onEvent partnerLink="purchasing"
                        operation="queryOrderStatus" ...>
                        <scope>...</scope>
                </onEvent>
                <onEvent partnerLink="purchasing"
                        operation="cancelOrder" ...>
                        <scope>...</scope>
                </onEvent>
        </eventHandlers>
        ...
</process>
```

Znacznik **<eventHandlers>** musi być potomkiem znacznika **<process>**.

Obsługa wyjątków

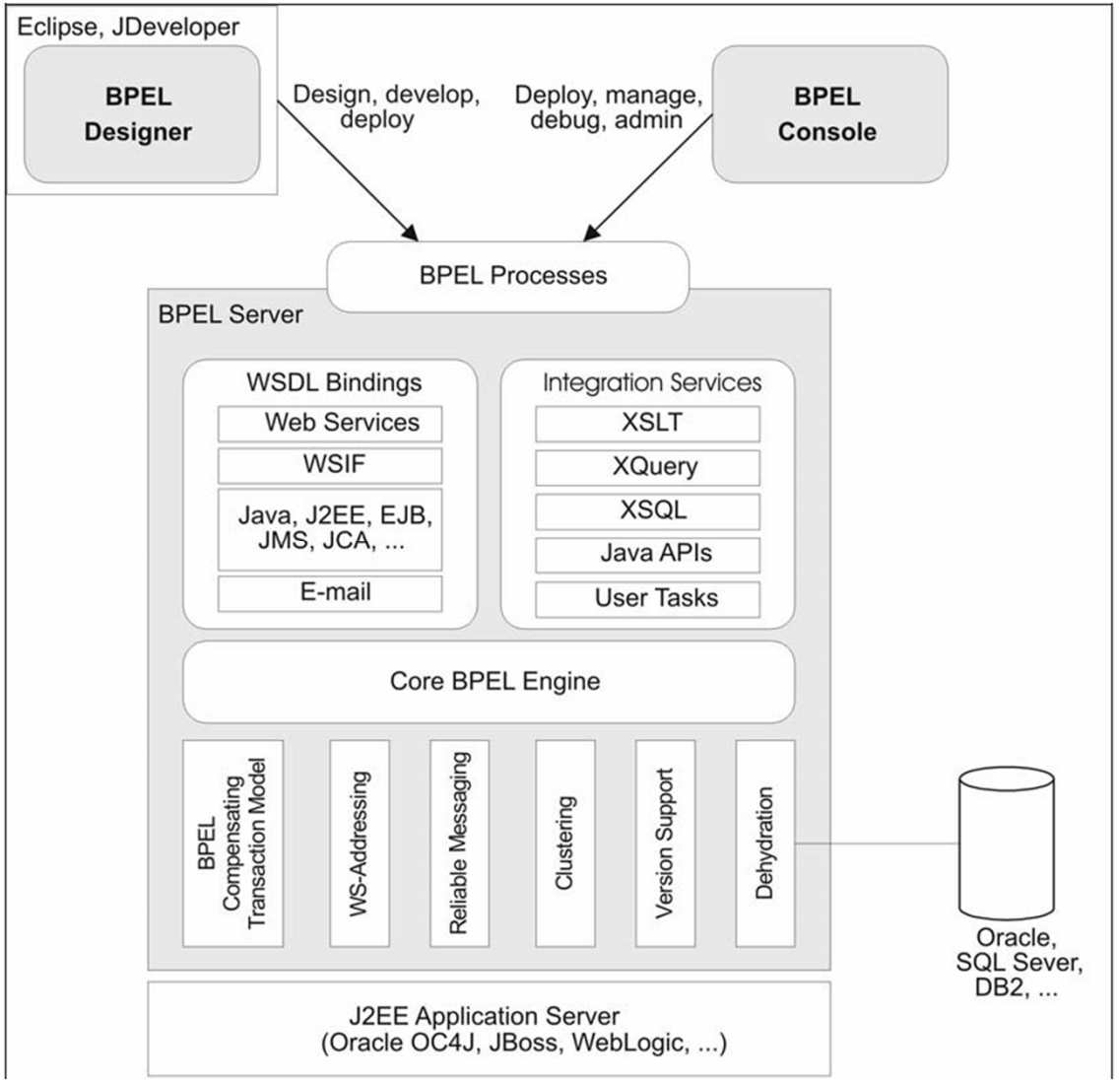
BPEL dostarcza mechanizmu do obsługi sytuacji wyjątkowych. Mechanizm ten jest modelowany poprzez znacznik **<faultHandlers>**. Znacznik ten musi być zdefiniowany jako dziecko znacznika **<scope>**. Wewnątrz definiowane są bloki obsługi wyjątków. Operacje wykonywane wewnątrz bloku **<scope>** mogą również podlegać kompensacji w razie wyjątku dzięki znacznikowi **<compensationHandler>**.

```
<scope name="S1">
        <faultHandlers>
                <catch xmlns:flt="http://example/faults" faultName="flt:OutOfStock" >...</catch>
                <catchAll>
                        <compensateScope target="S2" />
                </catchAll>
        </faultHandlers>
        <sequence>
                <scope name="S2">
                        <compensationHandler>
                                <!-- undo work -->
                        </compensationHandler>
                        <!-- do some work -->
                </scope>
                <!-- do more work -->
                <throw xmlns:flt="http://example/faults" faultName="flt:OutOfStock" />
        </sequence>
</scope>
```

Środowisko wykonywania BPEL Oracle

Środowisko do zarządzania przepływami pracy w Oracle BPEL składa się z czterech podstawowych komponentów:

- BPEL Designer
- BPEL Server
- BPEL Console
- Baza danych



Oracle BPEL Server

Oracle BPEL Server działa w środowisku serwera aplikacji i składa się z trzech podstawowych części:

- **Core BPEL Engine** – środowisko wykonywania modelu procesów zapisanych w języku BPEL; do składowania informacji o stanie wystąpień procesów wykorzystywana jest baza danych
- **Wiązania WSDL** – umożliwiają integrację usług WebServices implementujących składowe zadania procesów
- **Usługi integracyjne** – transformacja dokumnetów XML i integracja z językiem Java