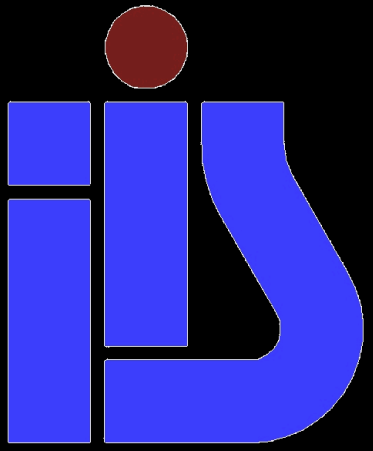


BPMN

Przenośność i wykonywalność definicji procesu

Legenda

- Serializacja i wymiana modeli
  - BPMN 2.0 a BPEL i XPDL
- Poziom wykonywalny
  - Dane procesu
  - Usługi i wiadomości/komunikaty
  - Zadania użytkownika
  - Narzędzia workflow

BPMN

Serializacja i wymiana modeli

Problem wymiany danych modelu

- BPMN 1.x nie udostępniał standardu wymiany danych
  - Model utworzony w jednym narzędziu był nie do otworzenia w innym
    - Możliwe pośrednictwo innych standardów (BPEL (OASIS), XPDL (WfMC))
- BPMN 2.0 wprowadza standard formatu wymiany

Znaczenie wymiany modeli

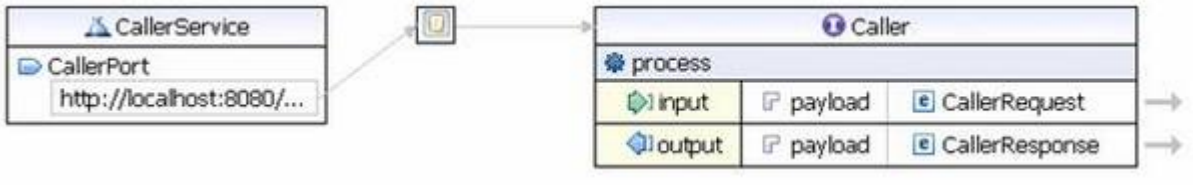
- Przypadek użycia 1:
  - Utworzenie niewykonywalnego modelu w narzędziu A i zaimportowanie go do narzędzia B
- Przypadek użycia 2:
  - Utworzenie niewykonywalnego modelu w narzędziu A i zaimportowanie go do nie BPMN-owego narzędzia B
- Przypadek użycia 3:
  - Utworzenie wykonywalnego modelu w narzędziu A i wykonanie go w środowisku B

BPEL jako format wymiany

1. BPEL to język wykonywalny procesów bazujących na usługach internetowych (ang. Web Sevices)
   - Przypadek użycia 2 i 3

- Problemy
  - BPEL -block-oriented a BPMN – graph-oriented
  - Wiele wzorców przepływów poprawnych w BPMN jest niepoprawna w BPEL
  - Brak natywnej konstrukcji dla zadań użytkownika
  - BPEL jest wykonywalny i nie ma standaryzowanej notacji graficznej (nie służy do analizy i dokumentowania)

BPEL przykład



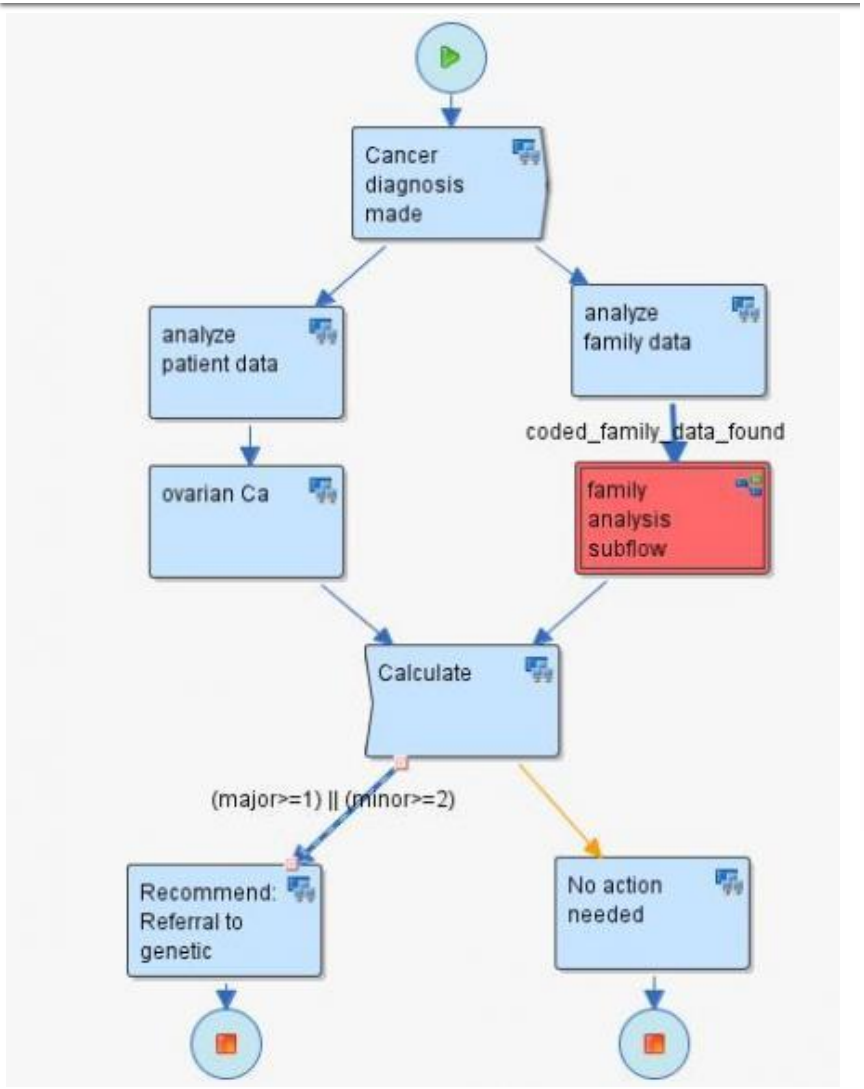

```
<?xml version="1.0" encoding="UTF-8"?>
<deploy xmlns="http://ode.fivesight.com/schemas/2006/06/27/dd"
    xmlns:pns="http://MyTest.com/Test"
    xmlns:wns="http://example.ws">

  <process name="pns:Caller">
    <active>true</active>
    <provide partnerLink="client">
      <service name="pns:CallerService" port="CallerPort"/>
    </provide>
    <!-- If there is a BPEL process which is also deployed under the same project-->
    <!--<provide partnerLink="DSLink">
      <service name="wns:DoSomethingService" port="DoSomething"/>
    </provide>-->
    <!-- DoSomethingService is an existing/external Web Service -->
     <invoke partnerLink="DSLink">
          <service name="wns:DoSomethingService" port="DoSomething"/>
     </invoke>
     <cleanup on="always" />
  </process>
  </deploy>
```

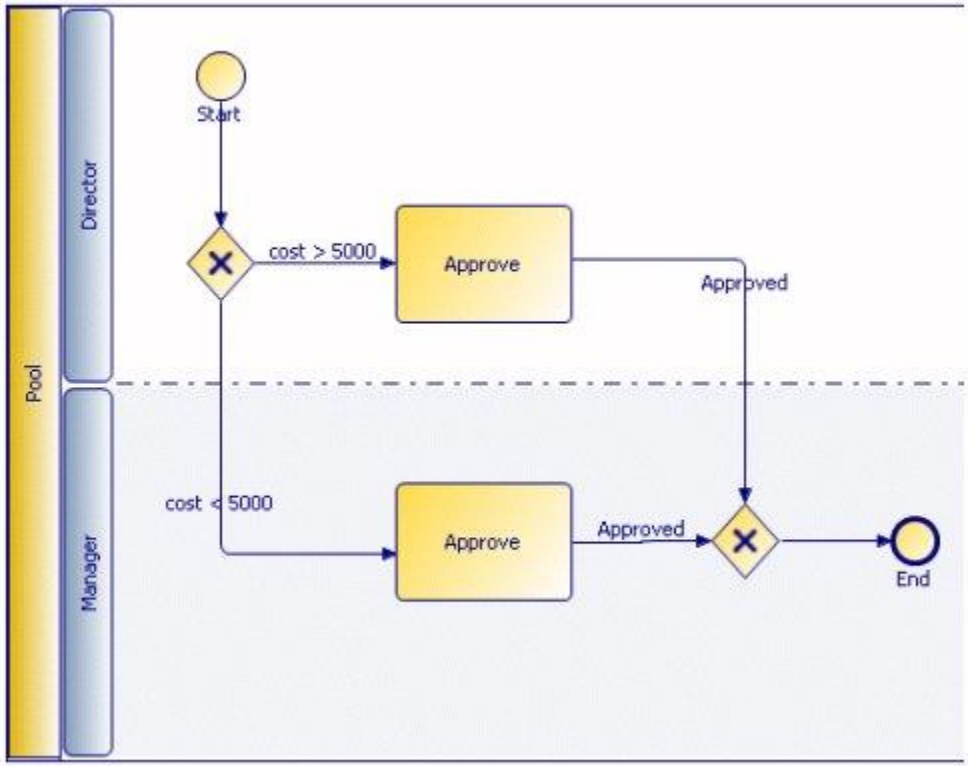
http://www.codeproject.com/Articles/33746/An-Example-Of-BPEL-invoking-WebService

XPDL jako format wymiany

- XPDL powstał jako format wymiany pomiędzy narzędziami do modelowania i narzędziami do wykonywania
  - Ogólny (dostawcy mogą wybrać co obsługują)
  - graph-oriented
- XPDL 2.0 –wsparcie BPMN 1.0
- XPDL 2.1 – wsparcie BPMN 1.1
- XPDL 2.2 (w planach) – wsparcie BPMN 2.0

XPDL przykład





BPMN 2.0 jako format wymiany

- Oficjalny format wymiany
  - XML Schema zapisany w xsd (generowany z bazującego na UML-u metamodelu BPMN)
  - Model semantyczny (ang. semantic model)
    - znaczenie
  - Model wymiany dla diagramu (ang. diagram interchange model)
    - Aranżacja elementów graficznych na stronie

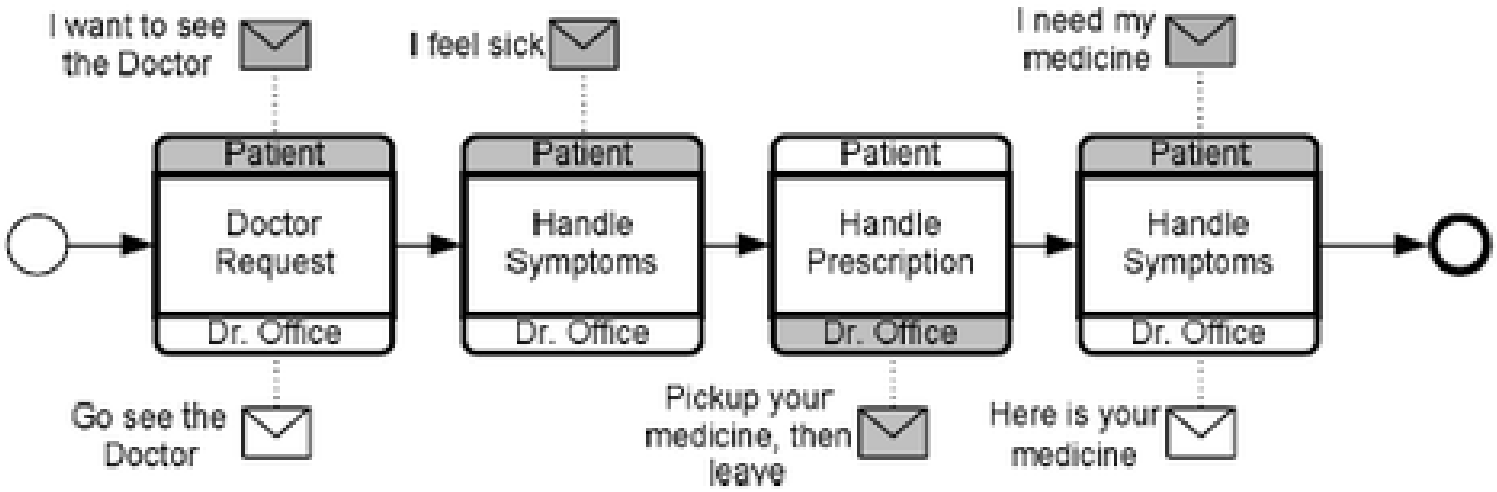
Przykładowa struktura dokumentu BPMN

```
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:activiti="http://activiti.org/bpmn" xmlns:bpmndi
    <process id="reviewSaledLead" isClosed="false" isExecutable="true" name="Review sales lead" processType="Non
      <startEvent activiti:initiator="initiator" id="theStart" isInterrupting="true" parallelMultiple="false">
      <sequenceFlow id="flow1" sourceRef="theStart" targetRef="provideNewSalesLead"/>
      <userTask activiti:assignee="${initiator}" activiti:exclusive="false" completionQuantity="1" id="provideNe
      <sequenceFlow id="flow2" sourceRef="provideNewSalesLead" targetRef="reviewSalesLeadSubProcess"/>
      <subProcess completionQuantity="1" id="reviewSalesLeadSubProcess" isForCompensation="false" name="Review s
      <sequenceFlow id="flow10" sourceRef="reviewSalesLeadSubProcess" targetRef="storeLeadInCrmSystem"/>
      <boundaryEvent attachedToRef="reviewSalesLeadSubProcess" cancelActivity="true" id="catchNotEnoughInformati
      <sequenceFlow id="flow11" sourceRef="catchNotEnoughInformationError" targetRef="provideAdditionalDetails"/
      <userTask activiti:assignee="${initiator}" activiti:exclusive="false" completionQuantity="1" id="provideAd
      <sequenceFlow id="flow12" sourceRef="provideAdditionalDetails" targetRef="reviewSalesLeadSubProcess"/>
      <task activiti:exclusive="false" completionQuantity="1" id="storeLeadInCrmSystem" isForCompensation="false
      <sequenceFlow id="flow13" sourceRef="storeLeadInCrmSystem" targetRef="processEnd"/>
      <endEvent id="processEnd">
    </process>
    <bpmndi:BPMNDiagram documentation="background=#FFFFFF;count=1;horizontalcount=1;orientation=0;width=597.6;heig
      <bpmndi:BPMNPlane bpmnElement="reviewSaledLead">
        <bpmndi:BPMNShape bpmnElement="theStart" id="Yaoqiang-theStart">
          <dc:Bounds height="32.0" width="32.0" x="75.0" y="300.0"/>
          <bpmndi:BPMNLabel>
            <dc:Bounds height="32.0" width="32.0" x="-1.0" y="-1.0"/>
          </bpmndi:BPMNLabel>
        </bpmndi:BPMNShape>
    </bpmndi:BPMNPlane>
  </bpmndi:BPMNDiagram>
</definitions>
```

BPMN 2.0 model semantyczny

- Trzy podstawowe elementy:
  - Process (orchiestration)
    - Opis pojedycznego procesu (orkiestracja - dyrygowanie)
  - Współpraca (collaboration)
    - Interakcja pomiędzy pulami za pośrednictwem przepływów wiadomości
  - Choreografia (choreography)
    - Interakcja pomiędzy uczestnikami na poziomie procesów (proces procesów) – istotna wymiana komunikatów

Choreografia - przykład



Pojedynczy dokument XML dla BPMN

- Może zawierać wiele procesów i potencjalnie wiele kolaboracji

- Może posiadać odniesienia do zewnętrznych definicji procesu (znacznik *import*)
  - Modularność
  - Możliwość opisu procesu od początku do końca

BPMN a przenośność modelu

- Wymiana (ang. interchange)
  - Standardowa serializacja
- Przenośność (ang. portability)
  - Zapewnianie wymiany (zazwyczaj na określonym poziomie)
    - Wymaga określenia poziomu zgodności (ang. conformance)

Poziomy zgodności

- Process Modeling Conformance,
  - Opisowy
  - Analityczny
  - Wspólny wykonywalny
- Process Execution Conformance,
  - Nie muszą wspierać modelowania (diagramów)
- BPEL Process Execution Conformance
- Choreography Modeling Conformance

BPMN

Poziom wykonywalny

Co to jest wykonywalny BPMN? (1)

- BPMN 1.x nie jest wykonywalny
  - Definicja abstrakcyjnego przepływu czynności - diagram
  - Brak jest możliwości specyfikacji m.in.:
    - Danych, wiadomości, interfejsów usług,
  - Szczegóły te traktowane są jako implementacyjne
    - Wyrażane w języku specyficznym dla rozwiązania workflow.

Co to jest wykonywalny BPMN? (2)

- BPMN 2.0 jest wykonywalny
  - Wszystko co potrzebne do wykonania definiowane w standardzie BPMN.
  - Technicznie jest to dodawanie dodatkowych informacji (metadanych).
    - Zostawiamy to co jest na diagramie
      - poziom analityczny
    - Na poziomie XML'a definiujemy szczegóły związane z wykonaniem - dla każdego elementu modelu (bramek, czynności, zdarzenia, przepływy danych)
      - **poziom wykonywalny**

Szczegóły związane z wykonaniem

- Definicje danych (zmiennych procesu)
- Definicje interfejsów usług
- Definicje wiadomości/komunikatów
- Wyrażenia dla bramek
- Reguły przypisywania zadań

Modelowanie danych

- Konstrukcje modelujące elementy (fizyczne lub informacyjne), które są tworzone, zmieniane i wykorzystywane w trakcie realizacji procesu
  - **Data Objects**, ItemDefinition, Properties, Data Inputs, Data Outputs, **Messages**, Input Sets, Output Sets, **Data Associations**
- Item-Aware Elements
  - Elementy BPMN'a przechowujace i przenoszące dane
    - **Data Objects, Data Object References, Data Stores**, Properties, DataInputs, DataOutputs

Obiekt danych (ang. data object)

- Podstawowa konstrukcja modelująca dane w procesie (kontener na dane)
  - Jest częścią procesu (obiekt globalny) lub podprocesu (obiekt lokalny)
    - Miejsce ogranicza zakres widoczności oraz czas życia (instancja procesu lub jego czynności)
  - Posiada reprezentację graficzną
    - Data object vs data object reference

Skład danych (ang. Data Store)

- Element powalający czynnościom procesu pobieranie i zapis informacji w sposób trwały (poza zakresem procesu)

Własność (ang. property)

- Nie ma reprezentacji graficznej
- Kontener na dane powiązane z
  - Procesem, czynnością, zdarzeniem
- Cykl życia
  - Zgodny z cyklem życia właściciela
- Dostęp
  - Z poziomy właściciela oraz bezpośrednich dzieci (dla procesu i podprocesu)

Dane wejściowe i dane wyjściowe

- Czynności i procesy często potrzebują danych do działania. Mogą je również produkować
  - Dane wejściowe (ang. Data input)
    - Deklaracja danych wejściowych (opcjonalnych lub wymaganych)
  - Dane wyjściowe (ang. Data output)
    - Proces, czynność produkuje dane
- Mogą być nie reprezentowane na diagramie


Data Input


Collection Data Input


Data Output


Collection Data Output

Powiązania danych

- Powiązanie danych (ang. Data association)
  - Powiązanie pomiędzy obiektem danych, składem danych lub własnością z jednej strony a danymi wejściowymi lub wyjściowymi z drugiej strony
- Aktywności maja dwa typy powiązań
  - dataInputAssociation, dataOutputAssociation
- Zdarzenia mają jedno powiązanie
  - dataInputAssociation – zdarzenia rzucająca
  - dataOutputAssociation – zdarzenia przechwytujące

Mapowanie danych

- Obiekty danych, wejścia i wyjścia to struktury danych
  - Definiowane za pomocą XML schema
- Elementy struktur
  - Dostęp za pomocą wyrażeń Xpath
    - Specjalne funkcje (np. , getDataObject, getDataInput)

```
<resourceparameterBinding parameterRef=”parmCountry”>
        <formalExpression>
            getDataInput(”revlewLoan—inputb”)/address/country
        </formalExpression>
</resourceParameterBinding>
```

BPMN

Usługi i wiadomości/komunikaty

Zadania zautomatyzowane

1. Reprezentowane jako usługi w metamodelu BPMN.
   - A **Service Task** is a **Task** that uses some sort of service, which could be a Web service or an automated application (BPMN, str. 158)
   - Atrybuty:
     - **implementation**: string = ##webService (##unspecified|URI)
     - operationRef: Operation [0..1]

Usługa – Interfejs

- Zestaw operacji implementowany przez usługę

Attribute Name	Description/Usage
name: string	The descriptive name of the element.
operations: Operation [1..*]	This attribute specifies operations that are defined as part of the `Interface`. An `Interface` has at least one `Operation`.
callableElements: CallableElement [0..*]	The `CallableElements` that use this `Interface`.
implementationRef: Element [0..1]	This attribute allows to reference a concrete artifact in the underlying implementation technology representing that interface, such as a WSDL porttype.

- Definicja adresu usługi jest poza zakresem BPMN'a
  - Występuje tylko pojęcie EndPoint

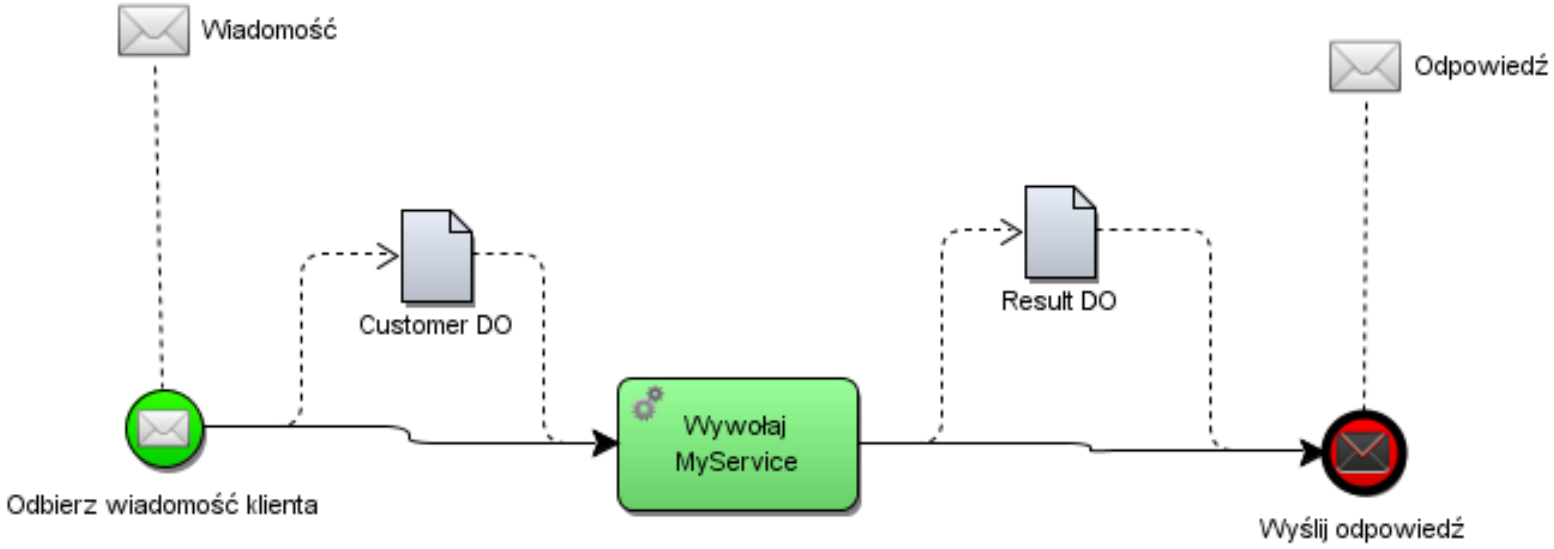
Usługa - Operacja

- Operacja definiuje komunikaty konsumowane i produkowane przez operację

Attribute Name	Description/Usage
name: string	The descriptive name of the element.
inMessageRef: Message	This attribute specifies the input **Message** of the `Operation`. An `Operation` has exactly one input **Message**.
outMessageRef: Message [0..1]	This attribute specifies the output **Message** of the `Operation`. An `Operation` has at most one input **Message**.
errorRef: Error [0..*]	This attribute specifies errors that the `Operation` may return. An `Operation` MAY refer to zero or more `Error` elements.
implementationRef: Element [0..1]	This attribute allows to reference a concrete artifact in the underlying implementation technology representing that operation, such as a WSDL operation.

Usługa – semantyka wykonywalna

1. Dane wejściowe zadania (Data Input) są przypisywane komunikatowi wejściowemu (inMessage) operacji (Operation)
2. Operacja jest wywoływana
3. Komunikat wyjściowy (outMessage) operacji (Operation) jest przypisywany danym wyjściowym (Data Output) zadania
4. Jeżeli usługa zwróci błąd jest on traktowany jako zdarzenie przerywające typu błąd.


Wiadomość
Odpowiedź
Customer DO
Result DO
Wywołaj
MyService
Odbierz wiadomość klienta
Wyślij odpowiedź

BPMN

Zadania użytkowników

Zadania wykonywane przez człowieka

- BPMN rozróżnia dwa typy:
  - Zadanie użytkownika (user task)

  - Zadanie ręczne (manual task)

- Założenie:
  - Zadanie może być zdefiniowane, przypisane i wykonane niezależnie od procesu biznesowego.
- Podstawowy cel specyfikacji zadania użytkownika:
  - Zdefiniowanie przypisania zadania do użytkownika lub roli

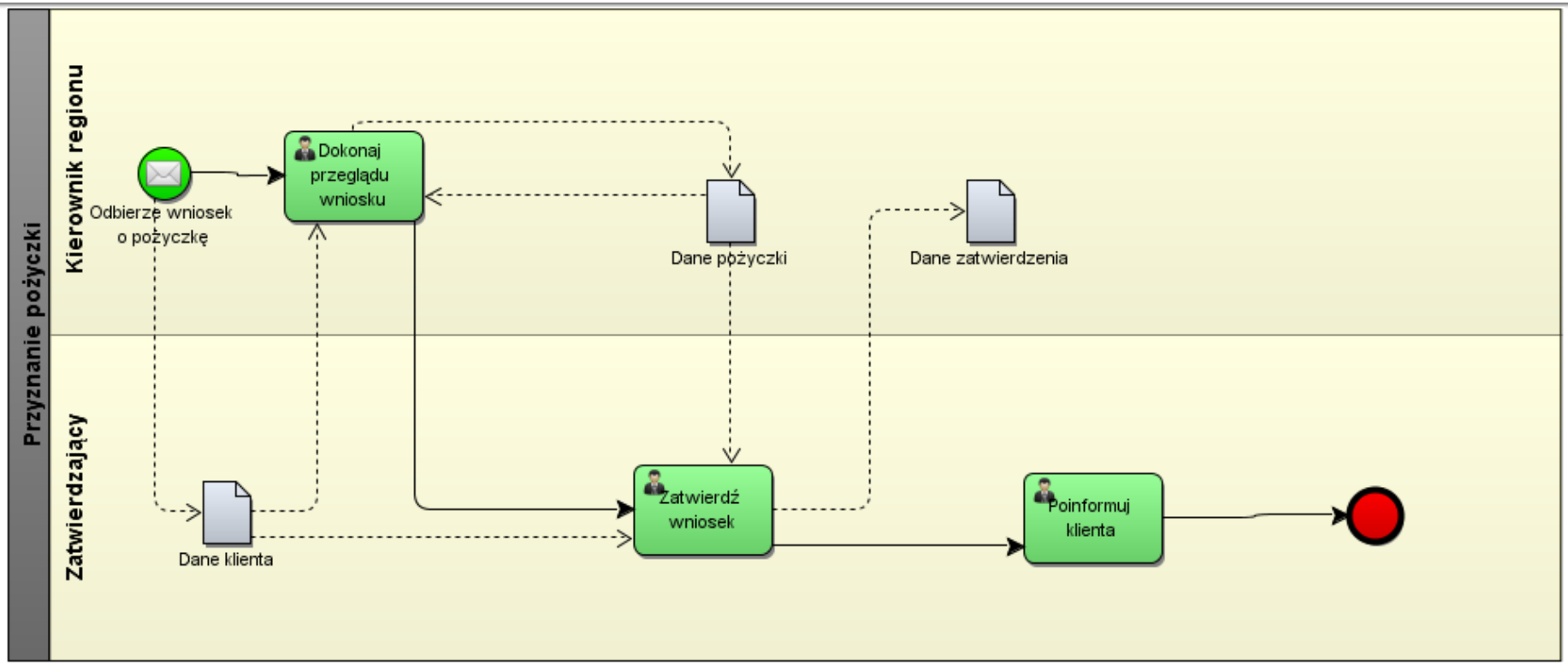
Wykonawca (ang. performer)

- Każda czynność procesu może być powiązana z zasobami
  - posiada referencję do zasobów zdefiniowanych w ramach modelu
  - W szczególności do wykonawcy (ang. performer)
  - Dla zdań „ludzkich” model może definiować specyficzne typy wykonawców
    - Potencjalny wykonawca – ma prawo do wykonania zadania i może z niego skorzystać (ang. claim)
    - Supervisor, quality control

Przypisanie wykonawcy do zadania

- W czasie definicji procesu (ang. *design time*)
  - Określony użytkownik lub grupa
- W czasie wykonania na podstawie wartości parametrów (ang. *parameter-based query evaluated at runtime*)
  - np. inicjator procesu, zatwierdzający

Wykonawcy a linie



- Linie pozwalają na organizowanie przepływu w dowolnej kategorii (niekoniecznie wykonawców)

Wykonawca a zadanie (1)

- Wykonawca (performer) jest modelowany jako zasób procesu.

```
<resource id="RS_1" name="Kierownik regionu">
  <resourceParameter id="RS_1_P_1" name="paramCity" type="xsd:string"/>
  <resourceParameter id="RS_1_P_2" isRequired="true" name="paramCountry" type="xsd:string"/>
</resource>
<resource id="RS_2" name="Zatwierdzający pożyczkę">
  <resourceParameter id="RS_2_P_1" isRequired="true" name="paramAmount" type="xsd:long"/>
</resource>
```

Wykonawca a zadanie (2)

```
<userTask completionQuantity="1" id="_6" implementation="##unspecified" isForCompensation="false" name="Dokonaj &#10;przeglądu &#10;wnios
  <incoming>_7</incoming>
  <outgoing>_9</outgoing>
  <ioSpecification>
    <dataInput id="Din_6_14" isCollection="false" name="Dane klienta"/>
    <dataInput id="Din_6_15" isCollection="false" name="Dane pożyczki"/>
    <dataInput id="Din_6_1" isCollection="false" name="Dane wykonawcy"/>
    <dataOutput id="Dout_6_15" isCollection="false" name="Dane pożyczki"/>
    <inputSet>
      <dataInputRefs>Din_6_14</dataInputRefs>
      <dataInputRefs>Din_6_15</dataInputRefs>
      <dataInputRefs>Din_6_1</dataInputRefs>
    </inputSet>
    <outputSet>..
  </ioSpecification>
  <dataInputAssociation id="_18">
    <sourceRef>_14</sourceRef>
    <targetRef>Din_6_14</targetRef>
  </dataInputAssociation>
  <dataInputAssociation id="_22">
    <sourceRef>_15</sourceRef>
    <targetRef>Din_6_15</targetRef>
  </dataInputAssociation>
  <dataOutputAssociation id="_20">..
  <humanPerformer id="_6_RES_1" name="Kierownik regionu">
    <resourceRef>RS_1</resourceRef>
    <resourceParameterBinding id="_6_RES_1_RS_1_B_1" parameterRef="RS_1_P_1">
      <formalExpression><![CDATA[getDataInput("Din_6_1")/address/city]]></formalExpression>
    </resourceParameterBinding>
    <resourceParameterBinding id="_6_RES_1_RS_1_B_2" parameterRef="RS_1_P_2">
      <formalExpression><![CDATA[getDataInput("Din_6_1")/address/country]]></formalExpression>
    </resourceParameterBinding>
  </humanPerformer>
</userTask>
```

Dane wejściowe nt. wykonawcy

Referencja do zasobu

Wiązanie danych wejściowych zadania z parametrami wykonawcy

BPMN

Narzędzia workflow

Wykonywalny BPMN 2.0 w narzędziach

- Standard jest dosyć młody więc ekosystem narzędzi nie jest mocno rozbudowany
  - Silniki Open Source
    - *JBoss BPM (jBPM)*
    - *BonitaSoft*
    - *Activiti*
    - Camunda BPM platform (+ wersja komercyjna)
  - Silniki komercyjne
    - ActiveVOS
    - Roubroo

JBoss BPM (jBPM)

- Pierwotnie wspierał język jPDL,
  - od wersji 5.0 wspiera BPMN 2.0
- Developer-oriented
- Maszyna stanów
- Integracja z Drools (silnikiem reguł biznesowych)
- jBPM Designer (Eclipse)

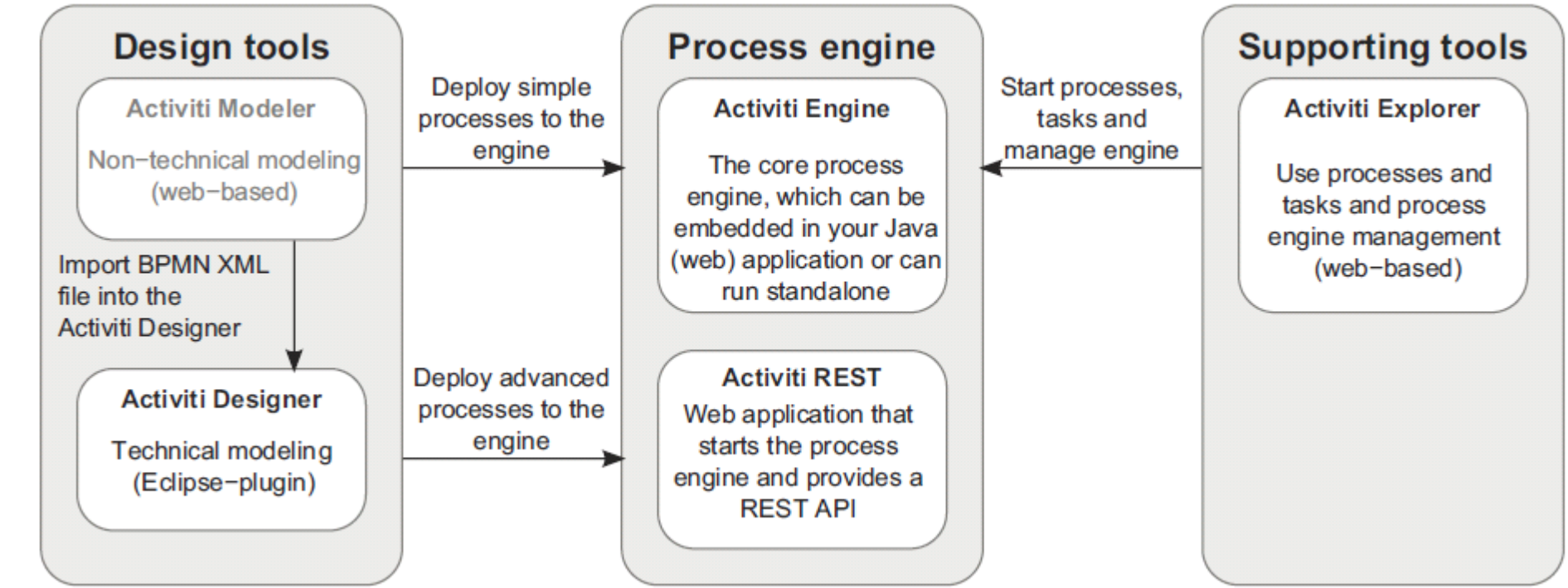
Bonitasoft

- Zorientowana na narzędzia generujące kod
  - nie wymagające pisania kodu
- Duże możliwości integracyjne
  - Ponad 100+ konektorów, możliwość budowani własnych
- Generacja aplikacji opartych na silniku procesu

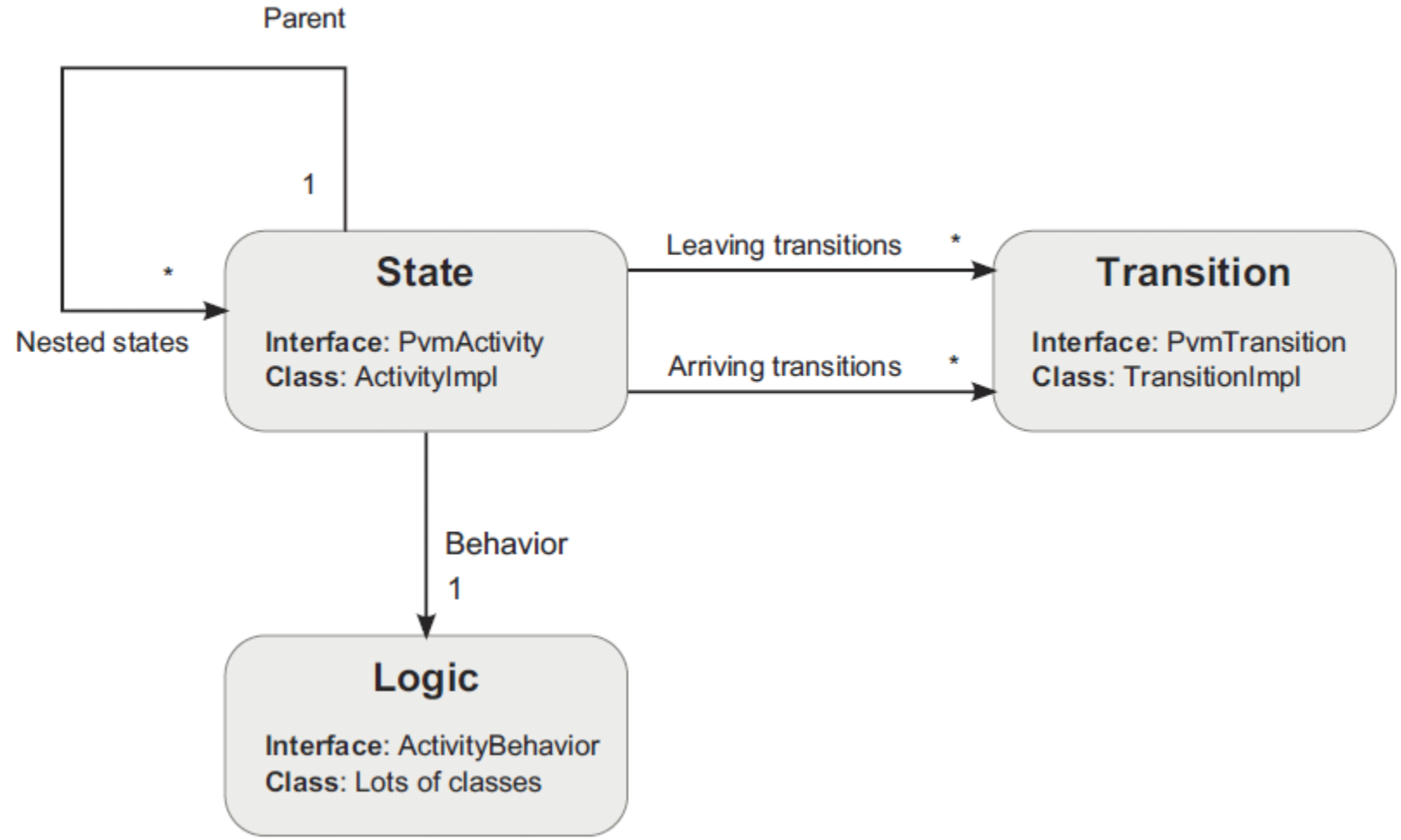
Activiti

- Uruchomienie projektu – 2010
  - Wersja produkcyjna - Grudzień 2010
- Społeczność developerów
  - SpringSource, FuseSource, Mulesoft
- Developer-oriented
- Maszyna stanów
- Implementacja specyfikacji BPMN 2.0
  - Umieszczenie (deploy) definicji procesu
  - Uruchamianie instancji
  - Wykonywanie zadań użytkowników
  - Wykonywanie semantyki BPMN 2.0

Narzędzia platformy



Maszyna stanów



Testy jednostkowe (1)

```
public class MyBusinessProcessTest {

  @Rule
  public ActivitiRule activitiRule = new ActivitiRule();

  @Test
  @Deployment
  public void ruleUsageExample() {
    RuntimeService runtimeService = activitiRule.getRuntimeService();
    runtimeService.startProcessInstanceByKey("ruleUsage");

    TaskService taskService = activitiRule.getTaskService();
    Task task = taskService.createTaskQuery().singleResult();
    assertEquals("My Task", task.getName());

    taskService.complete(task.getId());
    assertEquals(0, runtimeService.createProcessInstanceQuery().count());
  }
}
```

Testy jednostkowe (2)

```
@Test
public void createTask() {
  TaskService taskService = activitiRule.getTaskService();
  Task task = taskService.newTask();
  task.setName("Test task");
  task.setPriority(100);
  taskService.saveTask(task);
  assertNull(task.getAssignee());

  IdentityService identityService =
      activitiRule.getIdentityService();
  User user = identityService.newUser("JohnDoe");
  identityService.saveUser(user);

  taskService.addCandidateUser(task.getId(), "JohnDoe");
  task = taskService.createTaskQuery()
      .taskCandidateUser("JohnDoe")
      .singleResult();
  assertNotNull(task);
  assertEquals("Test task", task.getName());
  assertNull(task.getAssignee());

  taskService.claim(task.getId(), "JohnDoe");
  task = taskService.createTaskQuery()
      .taskAssignee("JohnDoe")
      .singleResult();
  assertEquals("JohnDoe", task.getAssignee());
```

⇩

⇩

```
  taskService.complete(task.getId());
  task = taskService.createTaskQuery()
      .taskAssignee("JohnDoe")
      .singleResult();
  assertNull(task);
}
```

Rozszerzenia

- Execution Listeners
  - Rozpoczynanie i kończenia instancji procesu, czynności, bramki, zdarzenia pośredniego.
  - Rozpoczynanie transakcji
- Email task
- Formularze dla zadań użytkownika

```
<startEvent ... >
  <extensionElements>
    <activiti:formProperty id="numberOfDays" name="Number of days" value="${numberOfDays}" type="long" required="true"/>
    <activiti:formProperty id="startDate" name="First day of holiday (dd-MM-yyy)" value="${startDate}" datePattern="dd-MM-yyyy hh:mm"
    <activiti:formProperty id="vacationMotivation" name="Motivation" value="${vacationMotivation}" type="string" />
  </extensionElements>
</userTask>
```