

## Exercise series 2

*Service orchestration with WS-BPEL:*

*Patient Treatment Flow*

Date	:	<i>April 17, 2012</i>
Course	:	Service-Oriented Architectures with Web Services
Course code	:	192652150
Group	:	<i>011</i>
Students	:	<i>Mark Wienk, 1238531</i> <i>Vincent van Donselaar, 0115282</i>

# Table of contents

[Table of contents](#)

[Introduction](#)

[1. Business Process Specification](#)

[2. WSDL Specification](#)

[3. Orchestration Architecture](#)

[4. WS-BPEL Process](#)

[Assignments](#)

[Physician](#)

[5. Process Deployment](#)

[6. Testing](#)

[7. Analysis of WS-BPEL code](#)

[8. Multiple Process Instances](#)

[References](#)

[Appendix A - BPEL specification](#)

# Introduction

In this exercise series, a total number of four web services are combined in a WS-BPEL process. The radiology service which we created during exercise one is one of them. Other services were made by fellow students. Since all of these implementations were built according to the WSDL-file delivered, the BPEL process should support these services no matter what.

Our plan was to design the BPEL process in advance, adding an arbitrary service implementation later on. In reality, we faced problems in the assign activity of the process, which took us hours to discover. An additional obstacle was that some projects of other groups were database driven, requiring external libraries and/or database management systems. Although it was not a problem to download and install both Apache Derby and MySQL, it distracted our attention from the real assignment.

Considering the Eclipse BPEL designer and Apache ODE, we must say that we were surprised by the effort it required to get things running. First, Eclipse was very picky on the version number of the Tomcat6 server. The standard Tomcat6 shipping with Ubuntu Linux didn't work at all. A manual installation did the trick eventually. Other hurdles emerged while deploying the several services on the server. Frequently, the deployment didn't succeed. Either it was not published on the server, or we faced an error. Starting each server individually from Eclipse worked sometimes, but there were scenarios in which we had to undeploy-stop-start-deploy the services. Looking back, we think that we spent around 90% of our time on the tooling, getting things to work. Both having practical experience with web services (not WS-BPEL though), we were unpleasantly surprised by this.

One last remark we had on the tooling is the lack of debugging functionality of the BPEL process. Running both servers in debugging mode didn't give the opportunity to place breakpoints, investigating the inner workings of the process. Having no other option, we decided to write some debug messages to the standard output of the server. Despite of the disappointment of not having a debugger tool, we were however very delighted about the existence of a unit testing framework. We will elaborate on this tool in section 6.

For completeness, our whole project is available on GitHub on the address <https://github.com/mwienk-school/SOA-S2> where it is possible to browse the code in a browser. Also, checking out the repository from there might be easier than downloading the zip from Blackboard.

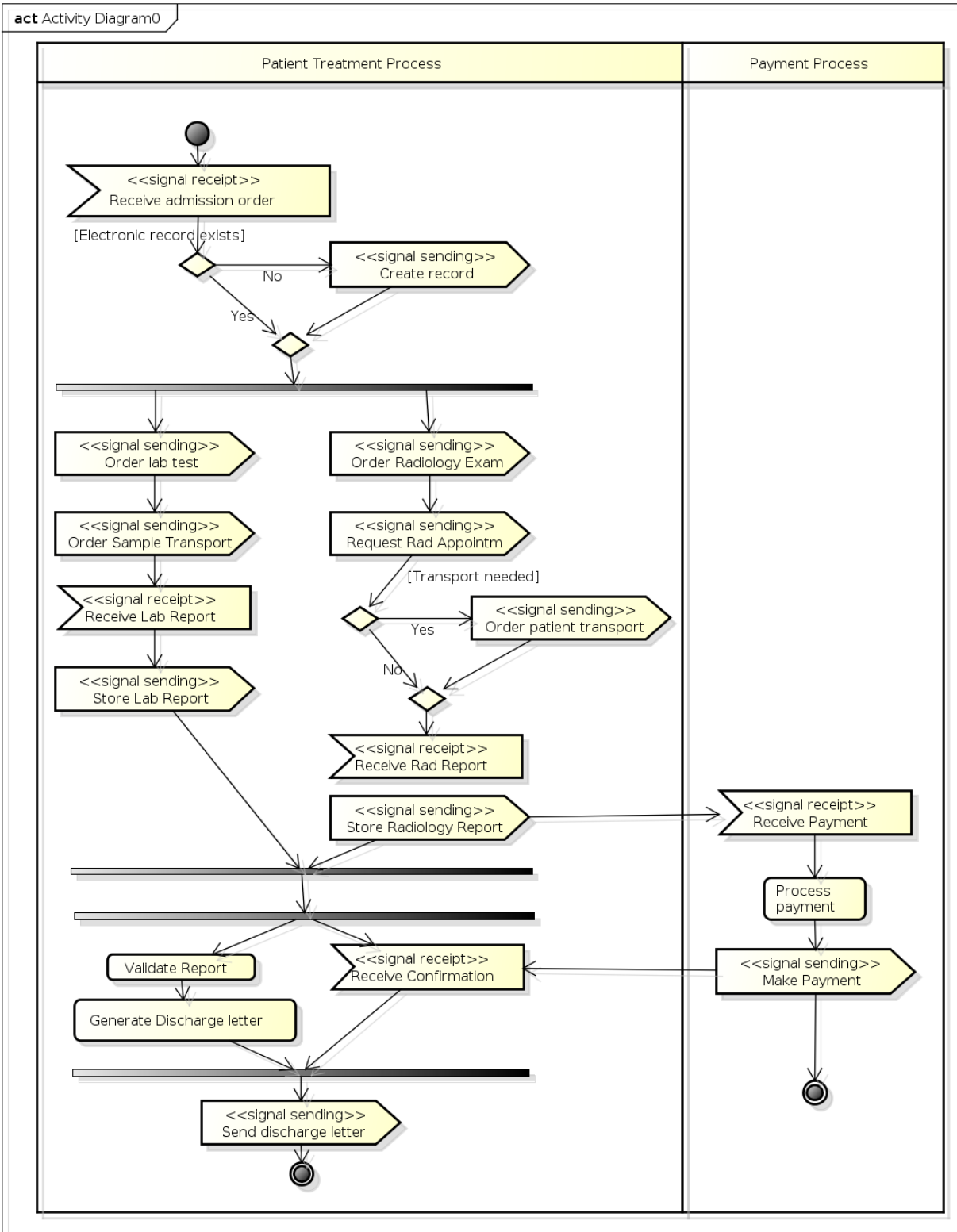
A digital version of this report is available on Google Docs for online viewing:  
<http://goo.gl/JIEhP>

# 1. Business Process Specification

The UML activity diagram below (*figure 1*) depicts the business process, showing the ward physician's actions. The major part happens in the left swim lane. The other interacting services are not displayed in their own swim lane to save some space in the diagram. Only the asynchronous payment service is displayed.

The physician's job starts as soon an admission order is received. The creation of a patient record is done automatically if the record does not exist.

The radiology examination and the lab test process run in parallel. The signal actions should wait for response, therefore the actions of the individual process threads are connected in series. One exception is the MakePayment operation. The ward's physician does not have to wait for this operation to finish. Therefore this one runs in a second swim lane: the Payment Process. As soon as the radiology reports are in, the physician validates the reports and writes a discharge letter, which is the end of the process.



powered by astah®

Figure 1: BPEL process in UML

## 2. WSDL Specification

The WSDL specification (*listing 1*) for the business process defines one service: the *Ward Service*. This service has just one operation: *OrderPatientTreatment*. This operation accepts a *PatientTreatmentOrder* message, which consists of a *PatientID* and the name of the patient (both *xsd:string* elements). This *PatientTreatmentOrder* message is the admission message which starts the entire patient treatment process.

The *OrderPatientTreatment* operation returns a *DischargeLetter* message, which is a *complexType* holding just one *xsd:string* element (body) that can be entered by a physician. We chose to use a *complexType* element so the addition of elements to the discharge letter (like lab results) can easily be done. This return message marks the end of the operation.

All other functionality is hidden from end users. This way, only the BPEL engine can control the business process (i.e. controlling the order of processing steps). In this specification we have used the *PhysicianService* for the physician's interaction (generating a discharge letter).

```
<types>
  <xsd:schema targetNamespace="http://www.PAHospital.org/WardService/">
    <xsd:element name="PatientTreatmentOrder">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="PatientID" type="xsd:string"/>
          <xsd:element name="PatientName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="DischargeLetter">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="body" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
</types>

<message name="WardOrderPatientTreatmentRequest">
  <part element="s0:PatientTreatmentOrder" name="PatientTreatmentOrder"/> </
message>
<message name="WardOrderPatientTreatmentResponse">
  <part element="s0:DischargeLetter" name="DischargeLetter"/>
</message>

<portType name="Ward">
  <operation name="OrderPatientTreatment">
```

```

        <input message="s0:WardOrderPatientTreatmentRequest"/>
        <output message="s0:WardOrderPatientTreatmentResponse"/>
    </operation>
</portType>

```

*Listing 1: key parts of WSDL specification (WardArtifacts.wsdl)*

### 3. Orchestration Architecture

In the *WardArtifacts.wsdl* file we added the namespace declaration for partnerLinkTypes `xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype".`

After we declared this namespace, the necessary `<partnerLinkType>` elements can be added to the WSDL file. The `<partnerLinkType>` elements are defined as *listing 2*. In this listing, the PartnerLinkTypes have quite straightforward names:

- PhysicianPLT                      -> PhysicianService
- WardPLT                            -> WardService
- TransportPLT                      -> TransportService
- LaboratoryPLT                    -> LaboratoryService
- PatientPLT                        -> ElectronicPatientRecord
- RadiologyPLT                     -> RadiologyService

```

<plnk:partnerLinkType name="PhysicianPLT">
    <plnk:role name="PhysicianService" portType="phys:PhysicianService"/>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="WardPLT">
    <plnk:role name="WardService" portType="s0:Ward"/>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="TransportPLT">
    <plnk:role name="TransportService" portType="ts:Transportation"/>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="LaboratoryPLT">
    <plnk:role name="LaboratoryService" portType="lab:Laboratory"/>
    <plnk:role name="LaboratoryRequester" portType="labcb:LabCallback"/>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="PatientPLT">
    <plnk:role name="ElectronicPatientRecordService"
        portType="pat:ElectronicPatientRecord"/>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="RadiologyPLT">
    <plnk:role name="RadiologyService" portType="rad:Radiology"/>
    <plnk:role name="RadiologyRequester" portType="radcb:RadiologyCallback"/>
</plnk:partnerLinkType>

```

*Listing 2: PartnerLinkTypes in WardArtifacts.wsdl*

This only defines the partnerLinkTypes. They connect the porttypes to a role a partnerLink can have. The callback services are also implemented as roles, these are the roles the *Ward*

business process will have (besides the *WardService* role). To use these types as partnerLinks in our BPEL process, the <partnerLink> elements must be defined in the BPEL process itself. We have defined these links as shown in *listing 3*.

```
<bpel:partnerLinks>
  <bpel:partnerLink name="WardService" partnerLinkType="ward:WardPLT"
    myRole="WardService" />
  <bpel:partnerLink name="TranspService" partnerLinkType="ward:TransportPLT"
    partnerRole="TransportService" />
  <bpel:partnerLink name="LabService" partnerLinkType="ward:LaboratoryPLT"
    partnerRole="LaboratoryService" myRole="LaboratoryRequester" /
>
  <bpel:partnerLink name="PatService" partnerLinkType="ward:PatientPLT"
    partnerRole="ElectronicPatientRecordService" />
  <bpel:partnerLink name="RadService" partnerLinkType="ward:RadiologyPLT"
    partnerRole="RadiologyService" myRole="RadiologyRequester" />
  <bpel:partnerLink name="PhysicianService"
    partnerLinkType="ward:PhysicianPLT"
    partnerRole="PhysicianService" />
</bpel:partnerLinks>
```

*Listing 3: PartnerLink implementation in ward.bpel file*

The partnerlinks carry the same name as the web service they call. The BPEL process has three roles (myRole attributes in the partnerlinks): *WardService*, *RadiologyRequester* and *LaboratoryRequester*. The *WardService* role is the role which is needed to be able to receive and reply requests for patient treatments. The other two roles, *RadiologyRequester* and *LaboratoryRequester* are both required to enable the BPEL process to asynchronously receive the Radiology and Laboratory reports via the callback services.

At this point in development, the services WSDL files still contain enough information. The complete business process can be designed by using the information from these WSDL files. However, when the system would be deployed, unreachable service errors would occur. For the deployment of the system we need the services to be implemented and fully operational. For now, the services can contain just some empty skeleton interface files.



## 4. WS-BPEL Process

The business process of the hospital ward is implemented in BPEL Designer. *Figure 2* shows the BPEL Designer process design. The design is pretty straightforward.

1. First, an admission order is received by the BPEL process.
2. Then the patient's ID is retrieved from the *ElectronicPatientRecord* service.  
This process is designed as an IF activity. It has two options: either the patient exists or patient doesn't exist. When the *ElectronicPatientRecord* service doesn't return an ID for a patient name, a new patient record is created. Then, the retrieved ID is associated with rest of the process.
3. After the patient ID is received, the orders for a radiology and a laboratory exam are respectively given to the *RadiologyService* and *LaboratoryService*. These processes can run independently, therefore a flow control type is appropriate. Within this flow, the required goods/patients are transported. The radiology process needs to be paid, therefore the *makePayment* process is invoked in parallel. Then both sequences wait for a report of the examination.
4. When these reports are received, they are stored at the patients record in the *ElectronicPatientRecord* service.
5. After that, the physician can examine these reports, this physician generates a discharge letter, which is stored in the reply (output) variable.
6. When both the discharge letter and the payment for the radiology examination are received. The discharge letter is returned by the process.

The BPEL Designer was actively used while designing the process. After manually adding the partner links, the activities could easily be selected for inclusion in the process. A consequence of using the designer tool is that every request and response is stored in another variable, which leads to some unnecessary allocation of memory (e.g. the returned report could be stored in the storage request message instantly). The allocation of this many variables does however attribute to the simplicity of the application. Now every activity has a well defined set of variables which it uses, see *listing 4*.

The complete BPEL specification can be found in *Appendix A*.

Note: in the UML activity diagram (*Figure 1*), a check whether a patient needs to be transported is proposed. However, there are no fields that indicate whether a patient needs to be transported. We chose to exclude this check in the process, as it could only have been implemented as a dummy.

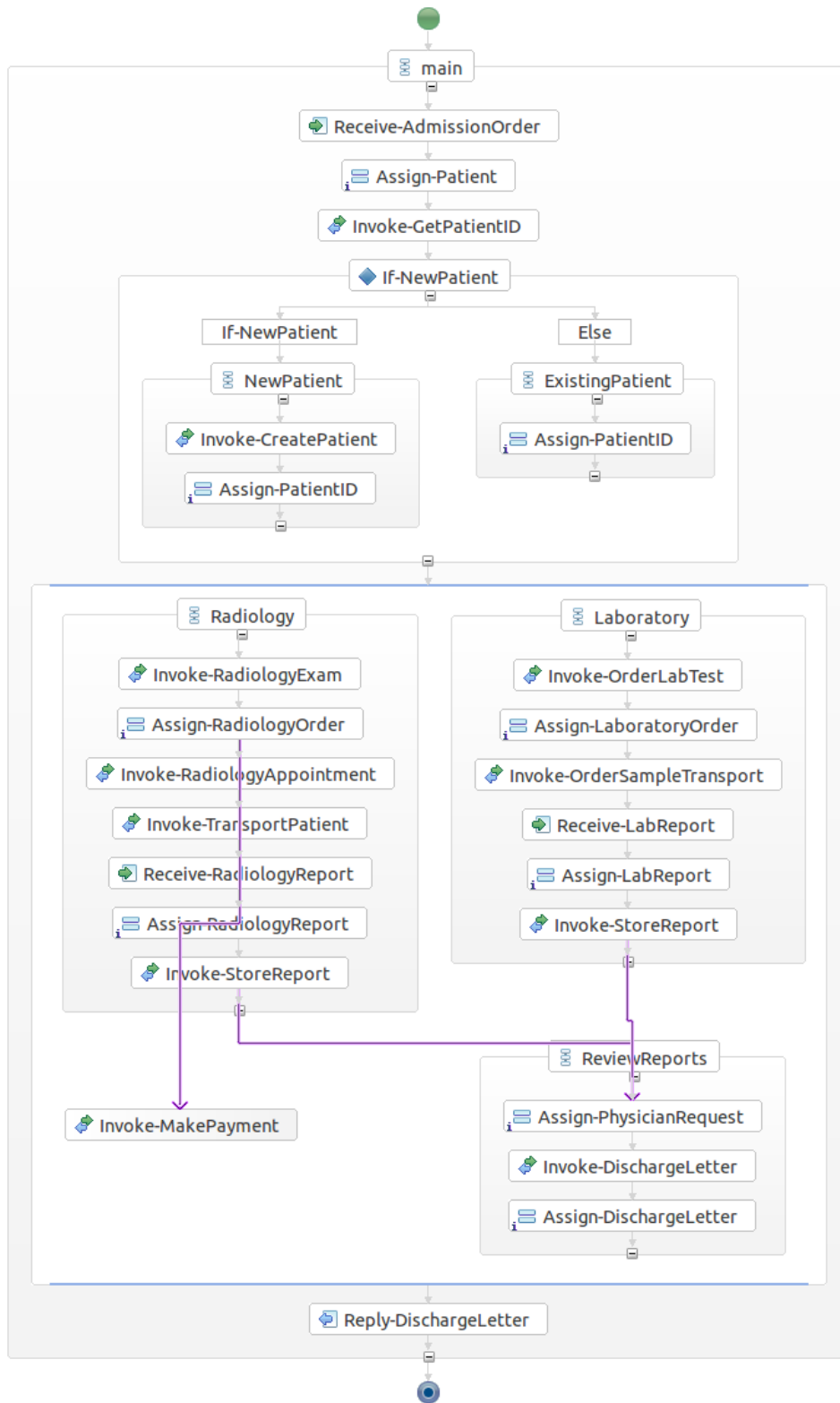


Figure 2: BPEL process design

```

<bpel:variables>
  <bpel:variable name="WardServiceRequest"
    messageType="ward:WardOrderPatientTreatmentRequest" />
  <bpel:variable name="WardServiceResponse"
    messageType="ward:WardOrderPatientTreatmentResponse" />
  <bpel:variable name="GetPatientIDRequest"
    messageType="pat:GetPatientIDsRequest" />
  <bpel:variable name="GetPatientIDResponse"
    messageType="pat:GetPatientIDsResponse" />
  <bpel:variable name="CreatePatientRequest"
    messageType="pat:CreatePatientRecordRequest" />
  <bpel:variable name="CreatePatientResponse"
    messageType="pat:CreatePatientRecordResponse" />
  <bpel:variable name="OrderRadiologyExamRequest"
    messageType="rad:RadiologyOrderRadExaminationRequest" />
  <bpel:variable name="OrderRadiologyExamResponse"
    messageType="rad:RadiologyOrderRadExaminationResponse" />
  <bpel:variable name="OrderLabTestRequest"
    messageType="lab:LaboratoryOrderLabTestRequest" />
  <bpel:variable name="OrderLabTestResponse"
    messageType="lab:LaboratoryOrderLabTestResponse" />
  <bpel:variable name="RadiologyAppointmentRequest"
    messageType="rad:RadiologyRequestAppointmentRequest" />
  <bpel:variable name="RadiologyAppointmentResponse"
    messageType="rad:RadiologyRequestAppointmentResponse" />
  <bpel:variable name="RadPatientTransportRequest"
    messageType="ts:OrderPatientTransportRequest" />
  <bpel:variable name="LabSampleTransportRequest"
    messageType="ts:OrderSampleTransportRequest" />
  <bpel:variable name="RadiologyReportResponse"
    messageType="radcb:RadiologyReportRequest" />
  <bpel:variable name="StoreRadiologyReportRequest"
    messageType="pat:StoreRadiologyReportRequest" />
  <bpel:variable name="LabReportResponse"
    messageType="labcb:SendLabReportResponse" />
  <bpel:variable name="StoreLabReportRequest"
    messageType="pat:StoreLabReportRequest" />
  <bpel:variable name="RadiologyMakePaymentRequest"
    messageType="rad:RadiologyMakePaymentRequest" />
  <bpel:variable name="PhysicianServiceResponse"
    messageType="phys:DischargeLetterResponse" />
  <bpel:variable name="PhysicianServiceRequest"
    messageType="phys:DischargeLetterRequest" /> </
bpel:variables>

```

*Listing 4: variable assignments in ward.bpel file*

## Assignments

Some of the assignments of fields could not be done. In the case of the transport service, a CaseID or SampleID is required. These are both undefined. To define these variables, either a more specific admission order or a samples (and cases) register would be required.

The Date field of the RadiologyAppointmentRequest is also undefined. The process now assigns a 'dummy' text to the requests, after which this generated text is overwritten with the necessary variable assignments.

In the process, a correlation between the invoked laboratory and radiology orders and the received reports for both was needed. These correlation consists of correlationsets which keep track of the received RadiologyOrderID and LabOrderID. The invoke and receive operations for both processes make use of these correlationsets.

## Physician

In order to be able to receive a discharge letter, a *PhysicianService* was implemented. This service now always returns the dummy text 'Patient seems healthy'. A real life deployment would implement a client to this service, in which the incoming reports are shown to a physician, after which this physician can enter the discharge letter. However, for the purpose of this process, this client isn't required. The dummy text returns the required information and a simple implementation as is suffices.

# 5. Process Deployment

We have implemented the processes of the other groups by dropping in the <Service>Skeleton.java files and fixing the dependencies like databases and 3rd party classes. This process did cost some time. Normally the web services one would combine into a business process would be already deployed on some server. In that case, only the WSDL and endpoints of the service would be required, relieving an outsider of the implementation of the service.

We have used Apache Tomcat 6 with Axis2 for the deployment of the web services. For the deployment of the business process, we used Apache ODE on a Tomcat 6 server. These servers were accessed in the Eclipse J2EE IDE, which makes for easy deployment. The deployment however didn't give much feedback, so debugging the process was pretty hard. In designing the process, we weren't able to simulate it, so debugging had to be done by trial and error at the deployment stage of the project. After spending a lot of time restarting every server and service, a working business process was realised.

To get the deployment working, first it was necessary to change the endpoints to the right location. Since eclipse doesn't give any information on which address it publishes the services, this was quite a puzzle (nowhere does it say the project name is used). After that, the real problems began, failing assignments, uninitialized partnerroles, selectionfault were some SOAP errors indicating what went wrong, but not where it went wrong. After a lot of time, we figured it all out.

On the ODE BPEL engine localhost website, the published processes are shown, but the buttons didn't work as expected. Pushing the details button didn't give any response, so that site could only be used to see if a process was started. We have tested the process with the Eclipse Web Services explorer tool using the *WardArtifacts.wsdl* file, which didn't give any problems.

Though the tools did cause quite some irritation, they did give us a better understanding of the workings of the BPEL engine.

## 6. Testing

A traditional way of testing software is unit-testing. One important aspect of unit testing is that the test subject is under full control of the testing framework, allowing in-depth testing of variables and expected return values. Testing web services makes things already slightly more complicated. Unit testing a web service involves both client and server. The server might hide its internals from the client. One option is to test both client and server separately, which requires mocking of the actual server's and client's behaviour respectively.

Testing a BPEL process, being a composition of multiple services, the problem becomes even harder. Looking for ways to test a BPEL process, we found the BPELUnit testing framework [1]. This framework allows (unit) testing of each operation involved in the process. Additionally, BPELUnit can run a coverage analysis and it allows built-in mocking of services.

In our case, we made a new test suite based on our BPEL process. For the sake of demonstration, we added one single test using some arbitrary input values as input message. The configuration is done using the interface depicted in the figure below.

### Test Suite

#### Test Suite

Enter a name and a base URL for this suite.

Suite Name

Base URL

[Configure Namespace Prefixes...](#)

#### Process Under Test

Enter a name, type, and WSDL file for the PUT.

PUT Name

PUT Type

WSDL

[Configure Deployment Options...](#)

#### Test Cases and Tracks

Manage test cases and partner tracks.



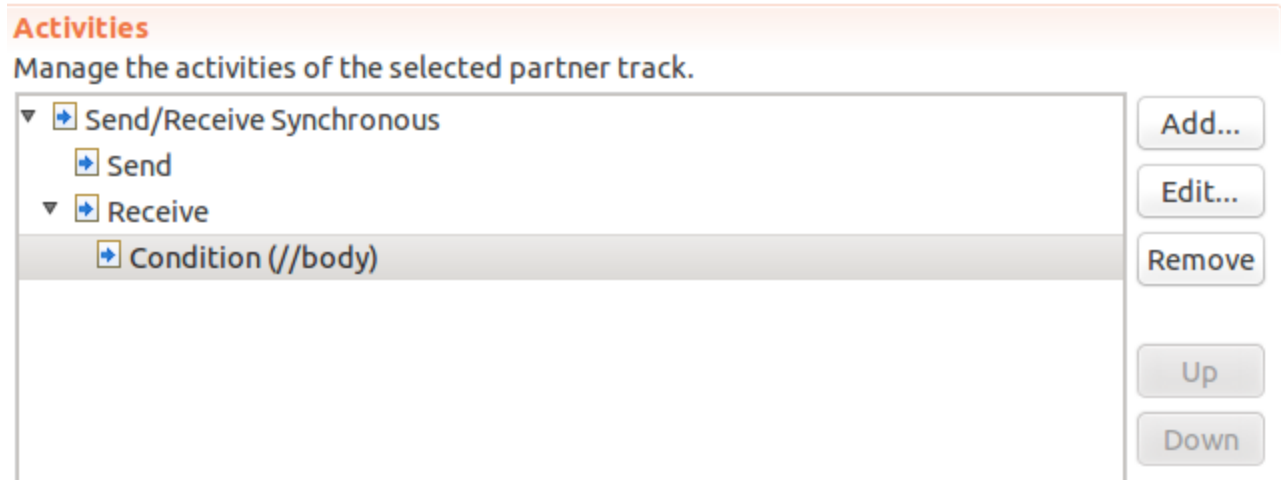
▼  FirstTest  
     client

Figure 3: Test suite settings

The test suite is configured by choosing a PUT (process under test), which was the Ward BPEL process in this assignment (*figure 3*). The suite can contain multiple test cases. In the figure, one test case is defined. This test consists of activities, each with expected input and output based on our single scenario. *Figure 4* gives an example of such an activity test.



*Figure 4: Test case activities*

In this case the activity is a synchronous send and receive activity. Editing the Send and Receive children of this test, a test XML file and expected response can be defined respectively. In our case we set the send to some arbitrary values (id: 123 and name: Vincent). The expected result is a soap message with an element body, which contains the text 'patient seems healthy'.

Once the test suite is run, BPELUnit shows a user friendly user interface in which it reports which tests have passed and which have failed, comparable to the way in which JUnit does (Figure 5).

The test result yields PASSED, the received message complies with the expected result (Figure 5).

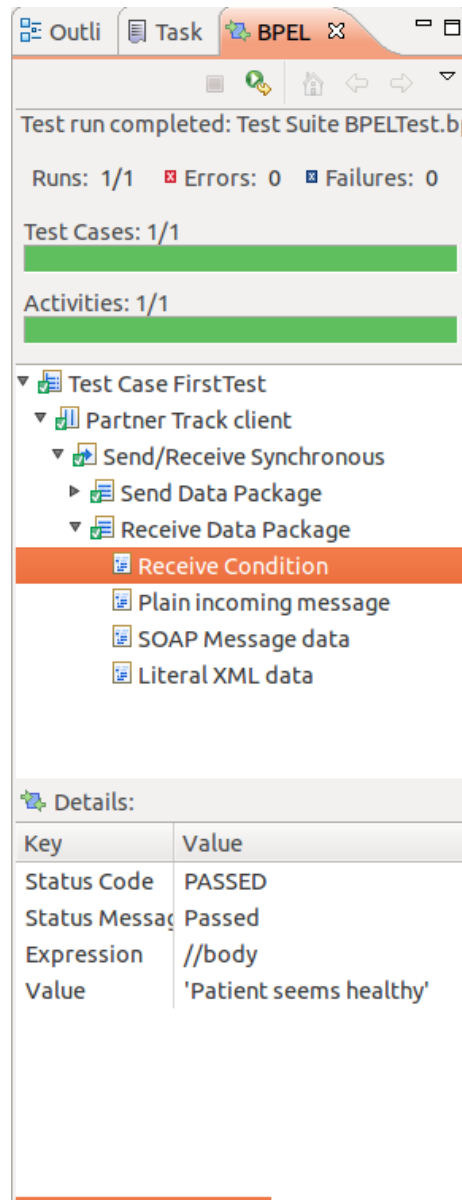


Figure 5: Test results

## 7. Analysis of WS-BPEL code

The BPEL source file is attached in Appendix A. Having a look at the generated code, we were not really surprised by the output, since the BPEL designer is almost a one to one representation of the actual file.

We faced a couple of design decisions which we could have done in multiple ways. We tried to follow the description as good as possible. Our design from *figure 1* resulted in the process shown in *figure 2*. The two threads were explicitly mentioned in the description. The generation of the discharge letter was however less clear to us. After asking some help, we decided that this was the way to interpret the last part of the process.

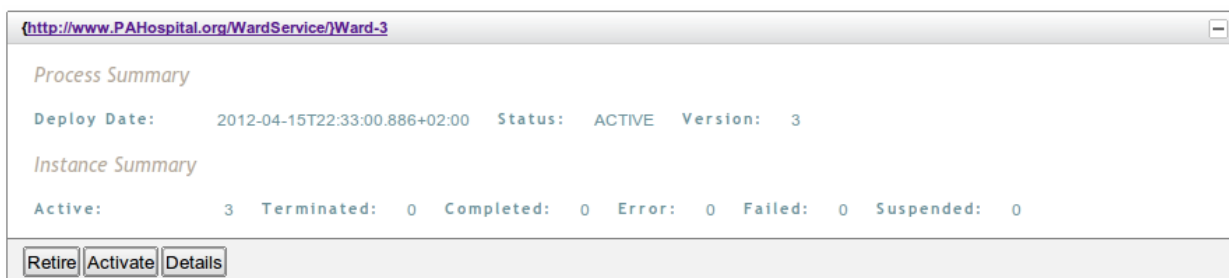
One thing we were asking ourselves was whether it is necessary to introduce this much variables in the process. Sometimes we only wanted to pass the outcome of one process as input to the other process, without storing the intermediate data in a variable. This could be done, but to keep a good overview of the process, we decided not to do it. Now any transport of data between variables happens in assign activities. An alternative would be for instance to receive the DischargeLetter and assign it to the input variable for the Reply-DischargeLetter activity.

We have used links for the transition from the radiology and laboratory process flows, this could probably also have been achieved differently. The assignment stated that the makePayment and generation of a discharge letter should be able to run in parallel. An alternative therefore could have been to put the *ReviewReports* sequence and the *makePayment* activity in a separate flow control outside of the current flow. The new flow would be implemented after the reports are received, so after the examination processes flow and before the Reply-DischargeLetter activity.

In our current process, the patient ID is assigned manually. If the patient name is found, the real ID is assigned to the required variables. This is actually a pretty weird situation, why supply an ID when the system will find it's own ID? This workflow however was necessary to support the patient creation activity (when the name isn't known to the system). The patient creation activity needs a patient ID parameter. Normally, the service would return the ID, but in the supplied WSDL, the *CreatePatientRecordRequest* message is a complete *Patient* type, with an ID element in it. An alternative is to change this specification to a *NewPatient* type (without the ID field).

## 8. Multiple Process Instances

While Apache ODE takes care of obvious tasks in managing the WS-BPEL processes, it is indeed a legitimate question how messages are assigned to the right flow instance. Important to know is that the ODE server keeps a state. Different process instances are identified by their unique process number, which is assigned to every new process created. Each deployment is able to spawn multiple instances. Using the ODE web interface, an overview of these deployments and processes is displayed. The screenshot below is an example of a deployment of the ward BPEL process.



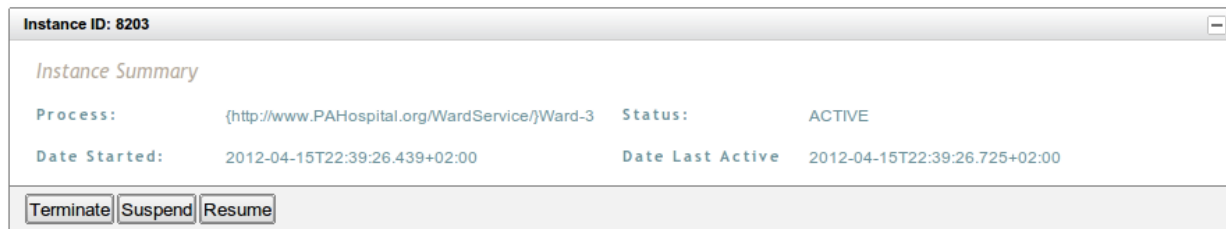
( <a href="http://www.PAHospital.org/WardService/Ward-3">http://www.PAHospital.org/WardService/Ward-3</a> )						
Process Summary						
Deploy Date:	2012-04-15T22:33:00.886+02:00	Status:	ACTIVE	Version:	3	
Instance Summary						
Active:	3	Terminated:	0	Completed:	0	Error: 0 Failed: 0 Suspended: 0
<a href="#">Retire</a> <a href="#">Activate</a> <a href="#">Details</a>						

Figure 6: ODE BPEL processes deployment overview

This deployment can have multiple process instances. The figure below shows one process



instance, having ID 8203.



*Figure 7: ODE BPEL instances overview*

The ability of having multiple conversations with multiple web services asynchronously is made possible by so called correlation sets. Correlation sets must be manually configured, this is not done 'automagically' by Apache ODE. Using the tutorial from [2] by Nguyen Ngoc Chan we figured out how to do this.

A 'normal' synchronous web service call doesn't need correlation sets to derive the correlation between the outgoing message and incoming response. Asynchronous calls however do need this aid. A correlation set is defined like in the example below:

```
<bpel:correlationSets>
  <bpel:correlationSet name="RadOrderID" properties="ward:RadiologyOrderID"/>
</bpel:correlationSets>
```

The invocation contains the correlation set defined above:

```
<bpel:correlations>
  <bpel:correlation set="RadOrderID" initiate="yes" pattern="response"/>
</bpel:correlations>
```

The receive activity has the counterpart of it:

```
<bpel:correlations>
  <bpel:correlation set="RadOrderID" initiate="no"/>
</bpel:correlations>
```

This links the receive activity to the right invoke activity (and therefore, the right process instance).

## References

- [1] <http://bpelunit.net/>
- [2] [http://www-inf.it-sudparis.eu/~nguyen\\_n/teaching\\_assistant/bpel/using\\_correlation\\_sets\\_in\\_bpel\\_processes](http://www-inf.it-sudparis.eu/~nguyen_n/teaching_assistant/bpel/using_correlation_sets_in_bpel_processes)

## Appendix A - BPEL specification

```
<?xml version="1.0"?>
<!-- Ward BPEL Process [Generated by the Eclipse BPEL Designer] -->
<!-- Date: Wed Mar 28 12:59:26 CEST 2012 -->
<bpel:process xmlns:ward="http://www.PAHospital.org/WardService/" xmlns:bpel="http://
/docs.oasis-open.org/wsbpel/2.0/process/executable" xmlns:ts="http://
www.PAHospital.org/TransportationService/" xmlns:lab="http://www.PAHospital.org/
LabService/" xmlns:labcb="http://www.PAHospital.org/LabCallbackService/"
xmlns:pat="http://www.PAHospital.org/PatService/" xmlns:rad="http://
www.PAHospital.org/RadiologyService/" xmlns:radcb="http://www.PAHospital.org/
RadiologyCallbackService/" xmlns:phys="http://www.PAHospital.org/PhysicianService/"
xmlns:ns1="http://www.w3.org/2001/XMLSchema" name="Ward" targetNamespace="http://
www.PAHospital.org/WardService/" suppressJoinFailure="yes">
  <!-- Import the service WSDL -->
  <bpel:import location="WardArtifacts.wsdl" namespace="http://www.PAHospital.org/
WardService/" importType="http://schemas.xmlsoap.org/wsdl/" />
  <!-- Import the PartnerLink WSDLs -->
  <bpel:import location="TranspService.wsdl" namespace="http://www.PAHospital.org/
TransportationService/" importType="http://schemas.xmlsoap.org/wsdl/" />
  <bpel:import location="LabService.wsdl" namespace="http://www.PAHospital.org/
LabService/" importType="http://schemas.xmlsoap.org/wsdl/" />
  <bpel:import location="LabService-callback.wsdl" namespace="http://www.PAHospital.org/
LabCallbackService/" importType="http://schemas.xmlsoap.org/wsdl/" />
  <bpel:import location="PatService.wsdl" namespace="http://www.PAHospital.org/
PatService/" importType="http://schemas.xmlsoap.org/wsdl/" />
  <bpel:import location="RadService.wsdl" namespace="http://www.PAHospital.org/
RadiologyService/" importType="http://schemas.xmlsoap.org/wsdl/" />
  <bpel:import location="RadService-callback.wsdl" namespace="http://www.PAHospital.org/
RadiologyCallbackService/" importType="http://schemas.xmlsoap.org/wsdl/" />
  <bpel:import location="PhysicianService.wsdl" namespace="http://www.PAHospital.org/
PhysicianService/" importType="http://schemas.xmlsoap.org/wsdl/" />
  <!-- ===== -->
  <!-- ORCHESTRATION LOGIC -->
  <!-- Set of activities coordinating the flow of messages across the -->
  <!-- services integrated within this business process -->
  <!-- ===== -->
  <bpel:partnerLinks>
    <bpel:partnerLink name="WardService" partnerLinkType="ward:WardPLT"
myRole="WardService"/>
    <bpel:partnerLink name="TranspService" partnerLinkType="ward:TransportPLT"
partnerRole="TransportService"/>
    <bpel:partnerLink name="LabService" partnerLinkType="ward:LaboratoryPLT"
partnerRole="LaboratoryService" myRole="LaboratoryRequester"/>
    <bpel:partnerLink name="PatService" partnerLinkType="ward:PatientPLT"
partnerRole="ElectronicPatientRecordService"/>
    <bpel:partnerLink name="RadService" partnerLinkType="ward:RadiologyPLT"
partnerRole="RadiologyService" myRole="RadiologyRequester"/>
    <bpel:partnerLink name="PhysicianService" partnerLinkType="ward:PhysicianPLT"
partnerRole="PhysicianService"/>
  </bpel:partnerLinks>
  <bpel:variables>
    <bpel:variable name="WardServiceRequest"
messageType="ward:WardOrderPatientTreatmentRequest"/>
    <bpel:variable name="WardServiceResponse"
messageType="ward:WardOrderPatientTreatmentResponse"/>
    <bpel:variable name="GetPatientIDRequest" messageType="pat:GetPatientIDsRequest"/>
    <bpel:variable name="GetPatientIDResponse" messageType="pat:GetPatientIDsResponse"/>
    <bpel:variable name="CreatePatientRequest"
messageType="pat:CreatePatientRecordRequest"/>
    <bpel:variable name="CreatePatientResponse"
```

```

messageType="pat:CreatePatientRecordResponse"/>
<bpel:variable name="OrderRadiologyExamRequest"
messageType="rad:RadiologyOrderRadExaminationRequest"/>
<bpel:variable name="OrderRadiologyExamResponse"
messageType="rad:RadiologyOrderRadExaminationResponse"/>
<bpel:variable name="OrderLabTestRequest"
messageType="lab:LaboratoryOrderLabTestRequest"/>
<bpel:variable name="OrderLabTestResponse"
messageType="lab:LaboratoryOrderLabTestResponse"/>
<bpel:variable name="RadiologyAppointmentRequest"
messageType="rad:RadiologyRequestAppointmentRequest"/>
<bpel:variable name="RadiologyAppointmentResponse"
messageType="rad:RadiologyRequestAppointmentResponse"/>
<bpel:variable name="RadPatientTransportRequest"
messageType="ts:OrderPatientTransportRequest"/>
<bpel:variable name="LabSampleTransportRequest"
messageType="ts:OrderSampleTransportRequest"/>
<bpel:variable name="RadiologyReportResponse"
messageType="radcb:RadiologyReportRequest"/>
<bpel:variable name="StoreRadiologyReportRequest"
messageType="pat:StoreRadiologyReportRequest"/>
    <bpel:variable name="LabReportResponse" messageType="labcb:SendLabReportResponse"/>
>
<bpel:variable name="StoreLabReportRequest" messageType="pat:StoreLabReportRequest"/>
<bpel:variable name="RadiologyMakePaymentRequest"
messageType="rad:RadiologyMakePaymentRequest"/>
<bpel:variable name="PhysicianServiceResponse"
messageType="phys:DischargeLetterResponse"/>
<bpel:variable name="PhysicianServiceRequest"
messageType="phys:DischargeLetterRequest"/>
    </bpel:variables>
    <bpel:correlationSets>
        <bpel:correlationSet name="RadOrderID" properties="ward:RadiologyOrderID"/>
        <bpel:correlationSet name="LabOrderID" properties="ward:LaboratoryOrderID"/>
    </bpel:correlationSets>
    <bpel:sequence name="main">
<bpel:receive name="Receive-AdmissionOrder" partnerLink="WardService"
operation="OrderPatientTreatment" variable="WardServiceRequest" createInstance="yes">
    </bpel:receive>
    <bpel:assign validate="no" name="Assign-Patient">
        <bpel:copy>
            <bpel:from>
                <bpel:literal xml:space="preserve">
<s0:Patient xmlns:s0="http://www.PAHospital.org/PatService/" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
                    <PatientID>PatientID</PatientID>
                                <PatientName>PatientName</PatientName>
                                <PatientStreet>PatientStreet</
PatientStreet>
                                <PatientZipCode>0</PatientZipCode>
                                <PatientCity>PatientCity</PatientCity>
                                <PatientBirthday>2001-01-01</
PatientBirthday>
                                </s0:Patient>
                </bpel:literal>
            </bpel:from>
            <bpel:to variable="CreatePatientRequest" part="Patient"/>
        </bpel:copy>
        <bpel:copy>
            <bpel:from>
                <bpel:literal>
                    <s0:PatientName xmlns:s0="http://www.PAHospital.org/PatService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

```

```

                                s0:PatientName
                                </s0:PatientName>

                                </bpel:literal>
                                </bpel:from>
                                <bpel:to variable="GetPatientIDRequest" part="PatientName"/>
                                </bpel:copy>
                                <bpel:copy>
                                    <bpel:from variable="WardServiceRequest" part="PatientTreatmentOrder">
                                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
                                [CDATA[PatientName]]></bpel:query>
                                    </bpel:from>
                                    <bpel:to variable="GetPatientIDRequest" part="PatientName"/>
                                    </bpel:copy>
                                    <bpel:copy>
                                        <bpel:from part="PatientTreatmentOrder" variable="WardServiceRequest">
                                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
                                [CDATA[PatientName]]></bpel:query>
                                        </bpel:from>
                                        <bpel:to part="Patient" variable="CreatePatientRequest">
                                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
                                [CDATA[PatientName]]></bpel:query>
                                        </bpel:to>
                                        </bpel:copy>
                                        <bpel:copy>
                                            <bpel:from part="PatientTreatmentOrder" variable="WardServiceRequest">
                                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
                                [CDATA[PatientID]]></bpel:query>
                                            </bpel:from>
                                            <bpel:to part="Patient" variable="CreatePatientRequest">
                                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
                                [CDATA[PatientID]]></bpel:query>
                                            </bpel:to>
                                            </bpel:copy>
                                        </bpel:assign>
                                <bpel:invoke name="Invoke-GetPatientID" partnerLink="PatService"
                                operation="GetPatientIDsByName" inputVariable="GetPatientIDRequest"
                                outputVariable="GetPatientIDResponse"/>
                                        <bpel:if name="If-NewPatient">
                                            <bpel:condition><![CDATA[$GetPatientIDResponse.PatientIDsList/PatientID=""]]></
                                bpel:condition>
                                            <bpel:sequence name="NewPatient">
                                <bpel:invoke name="Invoke-CreatePatient" partnerLink="PatService"
                                operation="CreatePatientRecord" portType="pat:ElectronicPatientRecord"
                                inputVariable="CreatePatientRequest" outputVariable="CreatePatientResponse"/>
                                            <bpel:assign validate="no" name="Assign-PatientID">
                                                <bpel:copy>
                                                    <bpel:from>
                                                        <bpel:literal>
                                                            <s0:LabOrder xmlns:s0="http://www.PAHospital.org/LabService/"
                                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                                                                <PatientID>PatientID</PatientID>
                                                                <CaseID>CaseID</CaseID>
                                                                <SampleID>SampleID</SampleID>
                                                                <SampleMaterial>BLOOD</SampleMaterial>
                                                                <LabParameter>LabParameter</LabParameter>
                                                            </s0:LabOrder>
                                                        </bpel:literal>
                                                    </bpel:from>
                                                    <bpel:to variable="OrderLabTestRequest" part="LabOrder"/>
                                                </bpel:copy>
                                                <bpel:copy>
                                                    <bpel:from>
                                                        <bpel:literal>

```

```

<s0:RadiologyOrder xmlns:s0="http://www.PAHospital.org/RadiologyService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <PatientID>PatientID</PatientID>
    <CaseID>CaseID</CaseID>
    <ExaminationType>XRAY</ExaminationType>
</s0:RadiologyOrder>
</bpel:literal>
</bpel:from>
<bpel:to variable="OrderRadiologyExamRequest" part="Order"/>
</bpel:copy>
<bpel:copy>
    <bpel:from part="PatientID" variable="CreatePatientResponse">
        </bpel:from>
        <bpel:to part="Order" variable="OrderRadiologyExamRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID]]></bpel:query>
        </bpel:to>
    </bpel:copy>
<bpel:copy>
    <bpel:from part="PatientID" variable="CreatePatientResponse"/>
    <bpel:to part="LabOrder" variable="OrderLabTestRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID]]></bpel:query>
        </bpel:to>
    </bpel:copy>
</bpel:assign>
</bpel:sequence>
<bpel:else>
    <bpel:sequence name="ExistingPatient">
        <bpel:assign validate="no" name="Assign-PatientID">
            <bpel:copy>
                <bpel:from>
                    <bpel:literal>
                        <s0:LabOrder xmlns:s0="http://www.PAHospital.org/LabService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                            <PatientID>PatientID</PatientID>
                            <CaseID>CaseID</CaseID>
                            <SampleID>SampleID</SampleID>
                            <SampleMaterial>BLOOD</SampleMaterial>
                            <LabParameter>LabParameter</LabParameter>
                        </s0:LabOrder>
                    </bpel:literal>
                </bpel:from>
                <bpel:to variable="OrderLabTestRequest" part="LabOrder"/>
            </bpel:copy>
            <bpel:copy>
                <bpel:from>
                    <bpel:literal>
                        <s0:RadiologyOrder xmlns:s0="http://www.PAHospital.org/RadiologyService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                            <PatientID>PatientID</PatientID>
                            <CaseID>CaseID</CaseID>
                            <ExaminationType>XRAY</ExaminationType>
                        </s0:RadiologyOrder>
                    </bpel:literal>
                </bpel:from>
                <bpel:to variable="OrderRadiologyExamRequest" part="Order"/>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="PatientIDsList" variable="GetPatientIDResponse">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID[1]]></bpel:query>
                    </bpel:from>

```

```

        <bpel:to part="Order" variable="OrderRadiologyExamRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID]]></bpel:query>
        </bpel:to>
    </bpel:copy>
    <bpel:copy>
        <bpel:from part="PatientIDsList" variable="GetPatientIDResponse">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID[1]]></bpel:query>
        </bpel:from>
        <bpel:to part="LabOrder" variable="OrderLabTestRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID]]></bpel:query>
        </bpel:to>
    </bpel:copy>
    </bpel:assign>
    </bpel:sequence>
</bpel:else>
</bpel:if>
<bpel:flow name="Flow-Review">
    <bpel:links>
        <bpel:link name="RadiologyReport"/>
        <bpel:link name="LaboratoryReport"/>
        <bpel:link name="link1"/>
    </bpel:links>
    <bpel:sequence name="Radiology">
<bpel:invoke name="Invoke-RadiologyExam" partnerLink="RadService"
operation="OrderRadiologyExamination" portType="rad:Radiology"
inputVariable="OrderRadiologyExamRequest" outputVariable="OrderRadiologyExamResponse">
        <bpel:correlations>
            <bpel:correlation set="RadOrderID" initiate="yes" pattern="response"/>
        </bpel:correlations>
    </bpel:invoke>
    <bpel:assign validate="no" name="Assign-RadiologyOrder">
        <bpel:copy>
            <bpel:from>
                <bpel:literal>
                    <s0:RadiologyOrderIDForPayment xmlns:s0="http://www.PAHospital.org/
RadiologyService/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                        s0:RadiologyOrderIDForPayment
                    </s0:RadiologyOrderIDForPayment>
                </bpel:literal>
            </bpel:from>
            <bpel:to variable="RadiologyMakePaymentRequest" part="RadiologyOrderID"/>
        </bpel:copy>
        <bpel:copy>
            <bpel:from>
                <bpel:literal>
<s0:PatientOrder xmlns:s0="http://www.PAHospital.org/TransportationService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    <PatientID>PatientID</PatientID>
                    <RequestingUnit>RequestingUnit</RequestingUnit>
                    <DestinationUnit>DestinationUnit</DestinationUnit>
                    <TransportationDate>2001-12-31T12:00:00</TransportationDate>
                </s0:PatientOrder>
            </bpel:literal>
        </bpel:from>
        <bpel:to variable="RadPatientTransportRequest" part="PatientTransportOrder"/>
    </bpel:copy>
    <bpel:copy>
        <bpel:from>
            <bpel:literal>
                <s0:Appointment xmlns:s0="http://www.PAHospital.org/RadiologyService/"

```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <OrderID>OrderID</OrderID>
    <Date>2001-01-01</Date>
  </s0:Appointment>
</bpel:literal>
</bpel:from>
<bpel:to variable="RadiologyAppointmentRequest" part="RadiologyAppointmentRequest"/>
</bpel:copy>
<bpel:copy>
  <bpel:from part="RadiologyOrderID" variable="OrderRadiologyExamResponse"/>
<bpel:to part="RadiologyAppointmentRequest" variable="RadiologyAppointmentRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[OrderID]]></bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:copy>
  <bpel:from part="Order" variable="OrderRadiologyExamRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID]]></bpel:query>
  </bpel:from>
<bpel:to part="PatientTransportOrder" variable="RadPatientTransportRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID]]></bpel:query>
  </bpel:to>
</bpel:copy>
<bpel:sources>
  <bpel:source linkName="link1"/>
</bpel:sources>
<bpel:copy>
  <bpel:from part="RadiologyOrderID" variable="OrderRadiologyExamResponse"/>
  <bpel:to part="RadiologyOrderID" variable="RadiologyMakePaymentRequest"/>
</bpel:copy>
</bpel:assign>
<bpel:invoke name="Invoke-RadiologyAppointment" partnerLink="RadService"
operation="RequestAppointment" portType="rad:Radiology"
inputVariable="RadiologyAppointmentRequest"
outputVariable="RadiologyAppointmentResponse">
  </bpel:invoke>
  <bpel:invoke name="Invoke-TransportPatient" partnerLink="TranspService"
operation="OrderPatientTransport" inputVariable="RadPatientTransportRequest">
  </bpel:invoke>
  <bpel:receive name="Receive-RadiologyReport" partnerLink="RadService"
operation="SendRadiologyReport" variable="RadiologyReportResponse">
    <bpel:correlations>
      <bpel:correlation set="RadOrderID" initiate="no"/>
    </bpel:correlations>
  </bpel:receive>
  <bpel:assign validate="no" name="Assign-RadiologyReport">
    <bpel:copy>
      <bpel:from>
        <bpel:literal>
          <s0:RadiologyReport xmlns:s0="http://www.PAHospital.org/PatService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <PatientID>PatientID</PatientID>
            <RadiologyOrderID>RadiologyOrderID</RadiologyOrderID>
            <DateOfExamination>2001-01-01</DateOfExamination>
            <ReportText>ReportText</ReportText>
          </s0:RadiologyReport>
        </bpel:literal>
      </bpel:from>
      <bpel:to variable="StoreRadiologyReportRequest" part="RadiologyReport"/>
    </bpel:copy>
    <bpel:copy>

```

```

        <bpel:from part="RadiologyReport" variable="RadiologyReportResponse"/>
        <bpel:to part="RadiologyReport" variable="StoreRadiologyReportRequest"/>
    </bpel:copy>
</bpel:assign>
<bpel:invoke name="Invoke-StoreReport" partnerLink="PatService"
operation="StoreRadiologyReport" portType="pat:ElectronicPatientRecord"
inputVariable="StoreRadiologyReportRequest">
    <bpel:sources>
        <bpel:source linkName="RadiologyReport"/>
    </bpel:sources>
</bpel:invoke>
</bpel:sequence>
<bpel:invoke name="Invoke-MakePayment" partnerLink="RadService"
operation="MakePayment" portType="rad:Radiology"
inputVariable="RadiologyMakePaymentRequest">
    <bpel:targets>
        <bpel:target linkName="link1"/>
    </bpel:targets>
</bpel:invoke>
<bpel:sequence name="Laboratory">
<bpel:invoke name="Invoke-OrderLabTest" partnerLink="LabService"
operation="OrderLabTest" portType="lab:Laboratory" inputVariable="OrderLabTestRequest"
outputVariable="OrderLabTestResponse">
    <bpel:correlations>
        <bpel:correlation set="LabOrderID" initiate="yes" pattern="response"/>
    </bpel:correlations>
</bpel:invoke>
<bpel:assign validate="no" name="Assign-LaboratoryOrder">
    <bpel:copy>
        <bpel:from>
            <bpel:literal>
<s0:SampleOrder xmlns:s0="http://www.PAHospital.org/TransportationService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <PatientID>PatientID</PatientID>
        <RequestingUnit>RequestingUnit</RequestingUnit>
        <SampleID>SampleID</SampleID>
    </s0:SampleOrder>
            </bpel:literal>
        </bpel:from>
        <bpel:to variable="LabSampleTransportRequest" part="SampleTransportOrder"/>
    </bpel:copy>
</bpel:copy>
    <bpel:from part="LabOrder" variable="OrderLabTestRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID]]></bpel:query>
        </bpel:from>
        <bpel:to part="SampleTransportOrder" variable="LabSampleTransportRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[PatientID]]></bpel:query>
        </bpel:to>
    </bpel:copy>
</bpel:assign>
<bpel:invoke name="Invoke-OrderSampleTransport" partnerLink="TranspService"
operation="OrderSampleTransport" portType="ts:Transportation"
inputVariable="LabSampleTransportRequest"/>
<bpel:receive name="Receive-LabReport" partnerLink="LabService"
operation="SendLabReport" variable="LabReportResponse">
    <bpel:correlations>
        <bpel:correlation set="LabOrderID" initiate="no"/>
    </bpel:correlations>
</bpel:receive>
<bpel:assign validate="no" name="Assign-LabReport">

```



```

    <bpel:copy>
      <bpel:from>
        <bpel:literal>
          <s0:LabReport xmlns:s0="http://www.PAHospital.org/PatService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <LabOrderID>LabOrderID</LabOrderID>
            <PatientID>PatientID</PatientID>
            <SampleID>SampleID</SampleID>
            <LabValues>
              <LabParameter>LabParameter</LabParameter>
              <LabValue>0.0</LabValue>
            </LabValues>
          </s0:LabReport>
        </bpel:literal>
      </bpel:from>
      <bpel:to part="LabReport" variable="StoreLabReportRequest"/>
    </bpel:copy>
    <bpel:copy>
      <bpel:from part="LabReport" variable="LabReportResponse"/>
      <bpel:to part="LabReport" variable="StoreLabReportRequest">
        </bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:invoke name="Invoke-StoreReport" partnerLink="PatService"
operation="StoreLabReport" portType="pat:ElectronicPatientRecord"
inputVariable="StoreLabReportRequest">
      <bpel:sources>
        <bpel:source linkName="LaboratoryReport"/>
      </bpel:sources>
    </bpel:invoke>
  </bpel:sequence>
  <bpel:sequence name="ReviewReports">
    <bpel:assign validate="no" name="Assign-PhysicianRequest">
      <bpel:copy>
        <bpel:from>
          <bpel:literal>
            <tns:ExaminationReports xmlns:tns="http://www.PAHospital.org/
PhysicianService/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
              <LabReport>
                <LabOrderID>LabOrderID</LabOrderID>
                <PatientID>PatientID</PatientID>
                <SampleID>SampleID</SampleID>
                <LabValues>
                  <LabParameter>LabParameter</LabParameter>
                  <LabValue>0.0</LabValue>
                </LabValues>
              </LabReport>
              <RadiologyReport>
                <PatientID>PatientID</PatientID>
                <RadiologyOrderID>RadiologyOrderID</RadiologyOrderID>
                <DateOfExamination>2001-01-01</DateOfExamination>
                <ReportText>ReportText</ReportText>
              </RadiologyReport>
            </tns:ExaminationReports>
          </bpel:literal>
        </bpel:from>
        <bpel:to variable="PhysicianServiceRequest" part="ExaminationReports"/>
      </bpel:copy>
    <bpel:targets>
      <bpel:target linkName="RadiologyReport"/>
      <bpel:target linkName="LaboratoryReport"/>
    </bpel:targets>
  </bpel:copy>

```

```

        <bpel:from part="LabReport" variable="StoreLabReportRequest"/>
        <bpel:to part="ExaminationReports" variable="PhysicianServiceRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[LabReport]]></bpel:query>
        </bpel:to>
    </bpel:copy>
    <bpel:copy>
        <bpel:from part="RadiologyReport" variable="StoreRadiologyReportRequest"/>
        <bpel:to part="ExaminationReports" variable="PhysicianServiceRequest">
<bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![
CDATA[RadiologyReport]]></bpel:query>
        </bpel:to>
    </bpel:copy>
</bpel:assign>
    <bpel:invoke name="Invoke-DischargeLetter" partnerLink="PhysicianService"
operation="RequestDischargeLetter" portType="phys:PhysicianService"
inputVariable="PhysicianServiceRequest" outputVariable="PhysicianServiceResponse"/>
    <bpel:assign validate="no" name="Assign-DischargeLetter">
        <bpel:copy>
            <bpel:from>
                <bpel:literal>
                    <s0:DischargeLetter xmlns:s0="http://www.PAHospital.org/WardService/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                        <body>body</body>
                    </s0:DischargeLetter>
                </bpel:literal>
            </bpel:from>
            <bpel:to variable="WardServiceResponse" part="DischargeLetter"/>
        </bpel:copy>
        <bpel:copy>
            <bpel:from part="DischargeLetter" variable="PhysicianServiceResponse"/>
            <bpel:to part="DischargeLetter" variable="WardServiceResponse"/>
        </bpel:copy>
    </bpel:assign>
</bpel:sequence>
</bpel:flow>
<bpel:reply name="Reply-DischargeLetter" partnerLink="WardService"
operation="OrderPatientTreatment" variable="WardServiceResponse"/>
    </bpel:sequence>
</bpel:process>

```