Hasso Plattner Institute
for Software Systems Engineering
Final year bachelor project
WS 2005/2006

Semantic SOA – Realization of the
Adaptive Services Grid

Open-Source Workflow Engine Integration into ASG Platform

February 28, 2006

Bastian Steinert, Jan Möller, Philipp Sommer,
Sebastian Steinhauer, Stefan Hüttenrauch,
Tobias Queck, Torsten Hahmann

Tables of Contents

List of Figures

List of Listings

List of Tables

1 INTRODUCTION

As part of efforts to develop an "Adaptive Services Grid" including automatic service discovery, selection, composition, and enactment we want to prove that alternative products can be used instead of existing ASG modules. The most important task is to show how composed services can be enacted. For that purpose the focus lies on workflow engines.

In ASG the upcoming standard for describing web service choreographies "Business Process Execution Language for Web Services" (WS-BPEL) is used. One goal of our bachelor project is to show that another workflow engine can be integrated into ASG. This can be reached either by integrating a WS-BPEL compliant workflow engine or by using a workflow engine with a different process definition language that BPEL can effortless be mapped to and provide a mapping component. In brief we want to design and implement an enactment component for the ASG platform using an existing workflow engine which executes given processes.

The market of workflow engines has developed rapidly during recent years. More and more implementations became available and especially the number of open-source solutions increased. The scope of these open-source implementations is often very limited and thus very special process definition languages are used.

The following document describes the evaluation process for choosing an open source workflow engine suiting ASG needs. Chapter 2 describes shortly the enactment subsystem interface. In chapter 3 we define a set of criteria to assess workflow engines in ASG context. Chapter 4 contains a list of currently available open-source workflow engines. In the following chapter 5 the criteria from chapter 3 are applied to the list of engines. In chapter 6 two example processes are defined and tested with the most promising candidates. The last two chapters describe the integration of the best fitting workflow engine – Fivesight PXE – in detail.

2 Interfaces for a Workflow Engine Integration

In accordance with the ASG reference architecture [1] the enactment subsystem provides functionality of enacting composed services. The primary realization approach is the use of a workflow engine. Two types of interfaces have to be considered: the interfaces provided and the interfaces used by the enactment subsystem.

The interface provided by the enactment subsystem is kept simple. Following listing shows the required methods in Java code.

```
public interface ServiceEnactment {

      public Response enactService(ComposedService composedService,
            Request request) throws ServiceEnactmentException;

      public void registerEventHandler(EnactmentEventHandler
            eventHandler) throws ServiceEnactmentException;

      public void unregisterEventHandler(EnactmentEventHandler
            eventHandler) throws ServiceEnactmentException;
}
```

Listing 1: Java interface of the Enactment Component

Main task of the enactment subsystem is to invoke Atomic Services of an underlying Services Grid Infrastructure. Invocation of an Atomic Service corresponds to a standardized web service call by using WS-Addressing. WS-Addressing information base on the concept of logical endpoint references. Hence, a *ComposedService* includes an endpoint reference for each Atomic Service that should be invoked during execution.

Furthermore the enactment subsystem needs to handle failures that occur during enactment. If the execution of an Atomic Service fails, the component tries to renegotiate a new service implementation to fulfil the process plan by calling the method *renegotiate()* of the negotiation manager. If renegotiation is not successful an specified exception has to be thrown to notify the upper architecture layer signalling problems while executing the services.

3 Evaluation Criteria

To decide which open-source workflow engine can be used in the ASG platform as an alternative to OfficeObjects WorkFlow, it is necessary to define a set of evaluation criteria according to the project requirements (see section requirements of Deliverable D4.III-1 [2]). Criteria can be divided into mandatory and optional ones. Mandatory criteria must be satisfied by all workflow engines that will be further considered. Optional criteria are additionally weighted by priorities. The following table lists all defined criteria to determine the engine that fits our needs best.

ID	Priority	Criteria	Rationale
CR1	Mandatory	Easy mapping from BPEL to internal process definition language of workflow engine	Enacting of BPEL process definitions
CR2	Mandatory	Workflow pattern support (D4.III-1 Requirement 2 [2]) ● all basic control patterns ● all cancellation patterns	
CR3	Mandatory	Web service invocation using WS-Addressing	Need to invoke Atomic Services in ASG platform
CR4	Mandatory	Open source licence (LGPL, CPL, ASL, BSD, ...) allowing use without need to publish own source code	Financial aspects; No publication of own code
CR5	Mandatory	Maturity (state, last version)	Stable workflow engine for productive use needed
CR6	Mandatory	Expected further development (work continuation)	Confirm to new standards in future (WS-BPEL 2.0)
CR7	Optional - 1	Direct support of at least a basic subset of BPEL 1.1 ● all basic control patterns ● all cancellation patterns	No need to transform BPEL definitions into process definition language of the workflow engine
CR8	*Ease of integration into ASG platform*		
CR8-1	Optional - 1	Functionality provided by available interfaces ● Process deployment ● Process instantiation ● Query process state ● Service invocation	Better interfaces allow implementation of a lightweight adapter
CR8-2	Optional - 1	Simple access to interfaces ● Use of standard mechanisms (technical realization) ● Well-documented	
CR8-3	Optional - 2	Expected runtime environment ● Component framework ● Additional libraries ● Persistence/DBMS	Less third-party dependencies
CR8-4	Optional - 3	Extensibility ● Programming language ● Code documentation ● Code structure	Potential changes to engine code might be required for integration

ID	Priority	Criteria	Rationale
CR9	Optional - 2	Interface for activity termination and Atomic Service cancellation (D4.III-1 Requirement 9 [2])	If SLA are not met, it should be possible to terminate process execution and execution of Atomic Services to save resources
CR10	Optional - 1	Notification for creation, starting, termination and completion of activity instances (D4.III-1 Requirement 11 [2])	Enactment service is required to notify listeners (monitoring, SLA manager) about process execution and service execution states
CR11	Optional - 3	Advanced workflow pattern support (D4.III-1 Requirement 3 [2]) ● most advanced branching and synchronization patterns ● structural pattern: implicit termination	Support for more sophisticated patterns that can be defined with BPEL might be requested during later phases of ASG
CR12	Optional - 3	Availability of additional tools ● Monitoring ● State ● Performance ● Visualization ● Debugging ● Process Designer	Easier process design, debugging and monitoring during integration helpful
CR13	Optional - 4	Ease of installation	
CR14	Optional - 4	Usability	

Table 1: Evaluation Criteria

4 Overview of Open-Source Workflow Engines

Name last version date	License	BPEL-Support	Description
ActiveBPEL 2.0 2005-09-08	GPL	Yes	The ActiveBPEL engine is a commercial-grade open source implementation of the BPEL4WS 1.1 specification and is fully compliant with that specification. [3]
Apache Agila	ASL	?	Apache Agila is a new donation to the Apache Software Foundation consisting of a lightweight BPM engine and auxiliary services. [4]
AntFlow 1.0 RC1 2004-10-29	ASL	No	AntFlow is a tool for the automation and scheduling of data system tasks, including those with complex dependencies and workflow logic. Antflow represents a new approach to simplify system automation that leverages pipelines of hot folders chained together to perform a given task. Using XML, Antflow associates an automated task, such as data transfer, compression, or encryption, with a directory on the local system. Whenever a file is copied or written into the hot folder, the associated task is executed and the file is moved to the next hot folder in the pipeline for further processing. [5]
bexee BPEL Execution Engine 0.1 2004-12-15	LGPL	Yes (partial imple-menta-tion ?)	bexee stands for BPEL Execution Engine and is an open source implementation of the BPEL standard. The bexee project has been initiated in the scope of a diploma project at the Berne University of Applied Sciences, School of Engineering and Information Technology by the students P. Fornasier and P. Kowalski under supervision of the project coach Dr. Eric Dubuis. [6]
BigBross Bossa Workflow-System 0.8.0 2004-03-18	GPL or com-mercial	No, Petri-Nets	Bossa is a workflow engine written in Java. The engine is very fast and lightweight, uses a very expressive Petri net notation to define workflows, does not require a RDBMS and is very simple to use and to integrate with Java applications. [7]
con:cern 2.0.1 2005-06-29	LGPL	No	con:cern is a workflow engine based on an extended case handling approach. A process is described as a set of activities with pre- and postconditions. An activity is executed when its preconditions are met. It manipulates the process item, thereby creating postconditions. The process flow is determined at run-time. [8]

Name last version date	License	BPEL-Support	Description
Enhydra Shark 1.1.2 2005-07-07	LGPL	No, XPDL	Shark is an extendable workflow engine framework including a standard implementation completely based on WfMC specifications using XPDL (without any proprietary extensions) as its native workflow process definition format and the WfMC "ToolAgents" API for server side execution of system activities. [9]
FiveSight PXE 1.0 RC1 2005-06-27	CPL	1.1 and 2.0	FiveSight PXE (short for Process eXecution Engine) is a runtime component for executing processes defined by the BPEL4WS 1.1 specification and forthcoming WS-BPEL 2.0 OASIS standard. [10]
Freefluo Workflow Enactor 0.9 2005-06-05	LGPL	No, WSFL and XScufl	Freefluo is a workflow orchestration tool for web services. It can handle WSDL based web service invocation. It supports two XML workflow languages, one based on IBM's WSFL and another named XScufl that is under development as part of the Taverna project. Freefluo is very flexible, at its core reusable orchestration framework that is not tied to any workflow language or execution architecture. Freefluo includes extension libraries that enable execution of workflows written in a subset of WSFL. There exists support for discovery via standard UDDI and recording of provenance. [11]
Jboss jBPM 3.0.1 2005-06-24	LGPL	No, jPDL	JBoss jBPM is a powerful workflow and BPM engine that enables the creation of business processes that coordinate between people, applications, and services. With its modular architecture, JBoss jBPM combines easy development of workflow applications with a flexible and scalable process engine. [12]
JFlower 0.5.1 2004-04-21	GPL	No	JFlower is workflow engine written in Java, extendible with Java plugins. The server parses XML documents in order to execute jobs and check conditions. The session data are stored in a database, so the server is fully scalable. [13]
JFolder 1.0 RC1 2004-09-13	LGPL	No	JFolder (formerly PowerFolder) is a workflow server and development studio. It can be configured to work on J2EE application servers and a variety of persistence stores (databases). It contains features critical to many applications - including web pages, workflow, security, persistence, email, file management, and data access. [14]

Name last version date	License	BPEL-Support	Description
MidOffice BPEL Editor 1.0 RC1 2004-10-29	GPL	Yes	The MidOffice BPEL Editor (MOBE) is an open-source platform for process orchestration which executes, monitors, adjusts, and terminates pre-defined processes. The platform is implemented using J2EE technologies and globally accepted standards like BPEL, XML and SOAP. [15]
ObjectWeb Bonita 1.6 2005-05-17	LGPL	No	Bonita is a flexible cooperative workflow system, compliant to WfMC specifications, based on the workflow model proposed by the ECOO Team, which incorporates the anticipation of activities as a more flexible mechanism of workflow execution. [16]
Open for Business (OfBiz) Workflow Engine 3.0 2004-03-12	MIT licence	No, XPDL	The Open For Business Project is an open source enterprise automation software project. By open source enterprise automation Open Source ERP, Open Source CRM, Open Source E-Business/E-Commerce, Open Source SCM, Open Source MRP, Open Source CMMS/EAM, and so on is meant. The Open for Business Workflow Engine is based on the WfMC and OMG specifications. [17]
Open Symphony OSWorkflow 2.7.0 2004-05-06	modi-fied ASL	No	Extremely flexible "low level" workflow system without graphical tools for developing workflows. Workflow descriptors are recommended to be "hand-written". Integration with existing code and databases is left to the application developer. [18]
Open Workflow Engine (OpenWFE) 1.5.5c 2005-08-25	BSD	No	OpenWFE is an open source Java workflow engine. It is a complete Business Process Management suite, with 4 components: an engine, a worklist, a webclient, and a reactor (host for automatic agents). A python access library is available: your python application/client can interact with an OpenWFE REST worklist. [19]
Plaengine (Engine)		No	Plaengine is a workflow management system for integrated process planning and enactment. Its unique feature is the automatic composition of processes. This allows individual process definitions for each business case. [20]
PL/FLOW 2.0b2 2005-06-29	LGPL	No, XPDL	PL/FLOW is a workflow engine written in Oracle PL/SQL, implementing interfaces 1 and 2 (Process Definition and the client API) as specified by the Workflow Management Coalition. [21]

Name last version date	License	BPEL-Support	Description
Syrup 0.3.2 2005-08-23	Artistic License	No	Syrup can be used to describe the tasks, procedural steps, required input and output information, and tools needed for each step in a business process. Syrup provides five basic concepts: Tasks, Links, Workflows, Workers, and the Workspace. Syrup can overcome the von Neumann bottleneck that stops traditional software systems from scaling. Syrup doesn't follow the more complex standards such as Wf-XML, BPML, and XPDL. [22]
Twister 0.3 2005-03-25	LGPL	Yes	Twister is a whole open source workflow (or business process management) solution, written in Java, using the WS-BPEL standard. It is web services oriented but also supports other ways of interaction. Twister joins the Apache Software Foundation as an incubated project. [23]
WfMOpen 1.3.2 2005-09-02	GPL or com-mercial	No, XPDL	WfMOpen is a J2EE based implementation of a workflow facility (workflow engine) as proposed by the Workflow Management Coalition (WfMC) and the Object Management Group (OMG). [24]
XFlow 1.2 2004-01-30	ASL	No	XFlow is a pure J2EE platform for building, executing, and managing business processes and workflows. It is a basis for building collaborative applications as well as integrating processes across an enterprise. XFlow has a small footprint but is extremely powerful. It is designed to be easy to use from the development, deployment, and management standpoints. [25]
XFlow2 2005-06-11	ASL	No	Xflow2 has the same simple workflow definition language than Xflow and was developed to improve the old implementation. [26]
YAWL 5.0.1 2005-07-15	LGPL	No, YAWL	YAWL is a workflow/business processing system. It supports a concise and powerful workflow language and handles complex data, transformations, and web service integration. Uses Java, Schema, XPath/Query, SOAP, and WSDL. [27]
Zebra Workflow 1.0 RC1 2004-12-13	ASL	No	Zebra is a workflow engine originally developed to fill in the gaps in some commercial and open source workflow engines. Its key features are the possibility to model all the workflows described in workflow patterns, a GUI designer, and Hibernate persistence layer. [28]

Table 2: Long List of Workflow Engines

5 Detailed Evaluation of Selected Workflow Engines

This chapter selects candidates from the long list of open-source workflow engines that can be integrated into the ASG platform. The criteria as defined in chapter 3 are the basis of selection procedure.

First only the mandatory criteria are regarded. Later we look closer at the expected costs of integration in order to narrow the selection of potential candidates. Goal of this chapter is a set of workflow engines that can be integrated in a limited period of time.

5.1 Potential Candidates for Integration based on Mandatory Criteria

For the selection of workflow engines that fit the needs for the ASG project the mandatory criteria defined in chapter 3 are used. The most restricting criteria are CR1 (process definition language), CR4 (license) and CR5 (maturity). Only few candidates satisfy all of those three requirements. Other aspects like ease of integration or stability must be examined to distinguish between them.

Potential candidates	Process Definition Language
Enhydra Shark	XPDL
FiveSight PXE	BPEL
FreeFluo Workflow Enactor (part of Taverna)	WSFL, XScufl
JBoss jBPM	jPDL
Open for Business (OfBiz) Workflow Engine	XPDL
Twister	BPEL
YAWL	YAWL

Table 3: Short List of Workflow Engines

Candidates excluded because of their GPL license (CR4) are ActiveBPEL, BigBross Bossa Workflow System, Jflower, MidOffice BPEL Editor, and WfMOpen. Those cannot be integrated into ASG without publication of ASG platform source code under GPL as well. PL/FLOW is distributed under LGPL but for execution it requires an Oracle database that requires a commercial license.

Other candidates (AntFlow, con:cern, JFolder, ObjectWeb Bonita, Syrup, XFlow, Xflow2, and Zebra Workflow) are excluded because of proprietary process definition languages without sufficient documentation to allow a complete mapping from BPEL to the native language format (CR1)).

Open Workflow Engine (OpenWFE) and Open Symphony OSWorkflow are excluded because they do not seem to support web service invocation, especially not conforming to WS-Addressing (CR3). Documentation of the workflow engines left imply that they support web service invocation. Nevertheless evaluation of web service invocation functionality must be analysed in more detail.

A subset of workflow engines do not show a current state that makes sense to integrate them into ASG. In order to base the enactment component upon a specific workflow engine its interfaces must be mature enough to ensure that no extensive alterations occur in the future

(CR5). Candidates lacking sufficient maturity (in addition to some of the already excluded engines) are Apache Agila, bexee BPEL Execution Engine, and Plaengine.

The criteria CR2 and CR3 can be examined by using simple business scenarios. Scenarios used for this purpose are introduced in the following chapter.

5.2 Further Restriction based on Expected Integration Costs

Taking into account the mandatory criteria CR1 requiring an easy mapping from BPEL to the process definition language used by the workflow engine, it is necessary to look for such mapping from BPEL to XPDL respectively WSFL, jPDL, YAWL.

BPEL is a block-structured programming language supporting graph structures (defined by a <flow> element containing <links>). Therefore a workflow engine with another process definition language must support both block and graph structures in some way. That includes the possibility to find a mapping from block structures to graph-based concepts in the workflow engines' process definition language and vice versa.

What concepts are used by the languages XPDL, WSFL, jPDL and YAWL? All these languages mainly use graph concepts. Therefore the block structures available in BPEL must be mapped to graph structures. Such a mapping is possible, however, it is not easy. Furthermore a mapping from the graph structures defined in BPEL to appropriate graph structures in the other process definition languages must be defined. In literature only a detailed theoretical approach for a mapping between BPEL a XPDL has been found [29]. However no concrete mapping between all individual language constructs is proposed. A full mapping that can be implemented would require additional efforts out of scope of a workflow engine integration. For the other process definition languages (WSFL, jPDL, YAWL) even theoretical approaches for a concrete mapping are rare. Costs for defining an adequate mapping would far exceed integration efforts.

We can conclude that an easy mapping to one of the above mentioned languages is not possible within reasonable limits. Therefore only two workflow engines (FiveSight PXE, Twister) directly supporting BPEL are left for detailed analysis.

Potential candidates	**Process Definition Language**
FiveSight PXE	BPEL
Twister	BPEL

Table 4: Workflow engines regarded for integration

6 Test of Prototypic ASG Scenarios

This chapter uses optional criteria defined in chapter 3 for detailed evaluation of Twister and FiveSight PXE. Furthermore the open mandatory criteria CR2 and CR3 are verified.

First two scenarios derived from the attraction booking scenario [30] are defined as BPEL process definitions with according WSDL-definitions. Those two scenarios are enacted with the workflow engine integration candidates Twister and FiveSight PXE. Such prototypic implementation is necessary to determine the actual effort required to integrate one of the workflow engines with ASG as specified in CR8. The scenarios are defined in a way to cover all basic control patterns as well as implicit termination. This complies with CR2 respectively CR7 from the criteria defined above.

Web service invocations are tested as well. This is done directly without using ASG C5 coordination layer. Support for WS-Addressing as stated in CR3 required for compliance with ASG coordination layer must be evaluated additionally.

6.1 Definition of ASG-Scenarios in BPEL

To evaluate current BPEL engine functionalities two scenarios are defined. The appropriate processes are defined in a graph-based structure, where a <flow> element contains all activities and their execution order is limited by <links>. Block structured BPEL is not tested at all as ASG process definitions are completely graph-based. BPMN is used for graphical representation of the scenarios [31]. Both scenarios have been fully implemented in BPEL4WS 1.1 with according WSDL-files for the service description [32]. The BPEL4WS executable process definitions can be found in the appendix.

A set of example ASG services developed for the Attraction Booking Scenario are used for web service invocation. Those are prototypes themselves but they perfectly serve test purposes. The first scenario basically tests web service invocation functionalities. A set of example ASG services developed for the Attraction Booking Scenario is used for web service invocation. Scenario one simply consists of a sequence of a receive activity followed by three web service calls and concluded by a reply activity. Additional variable manipulation is necessary to allow a consistent data flow.

The second scenario tests all basic workflow patterns – including the patterns parallel split, synchronisation, exclusive choice and simple merge in addition to the sequence pattern that has already been tested in scenario one [33]. All those patterns are represented as a single flow containing a set of activities with synchronization dependencies. Similar block structures (like <sequence>) are not used at all.

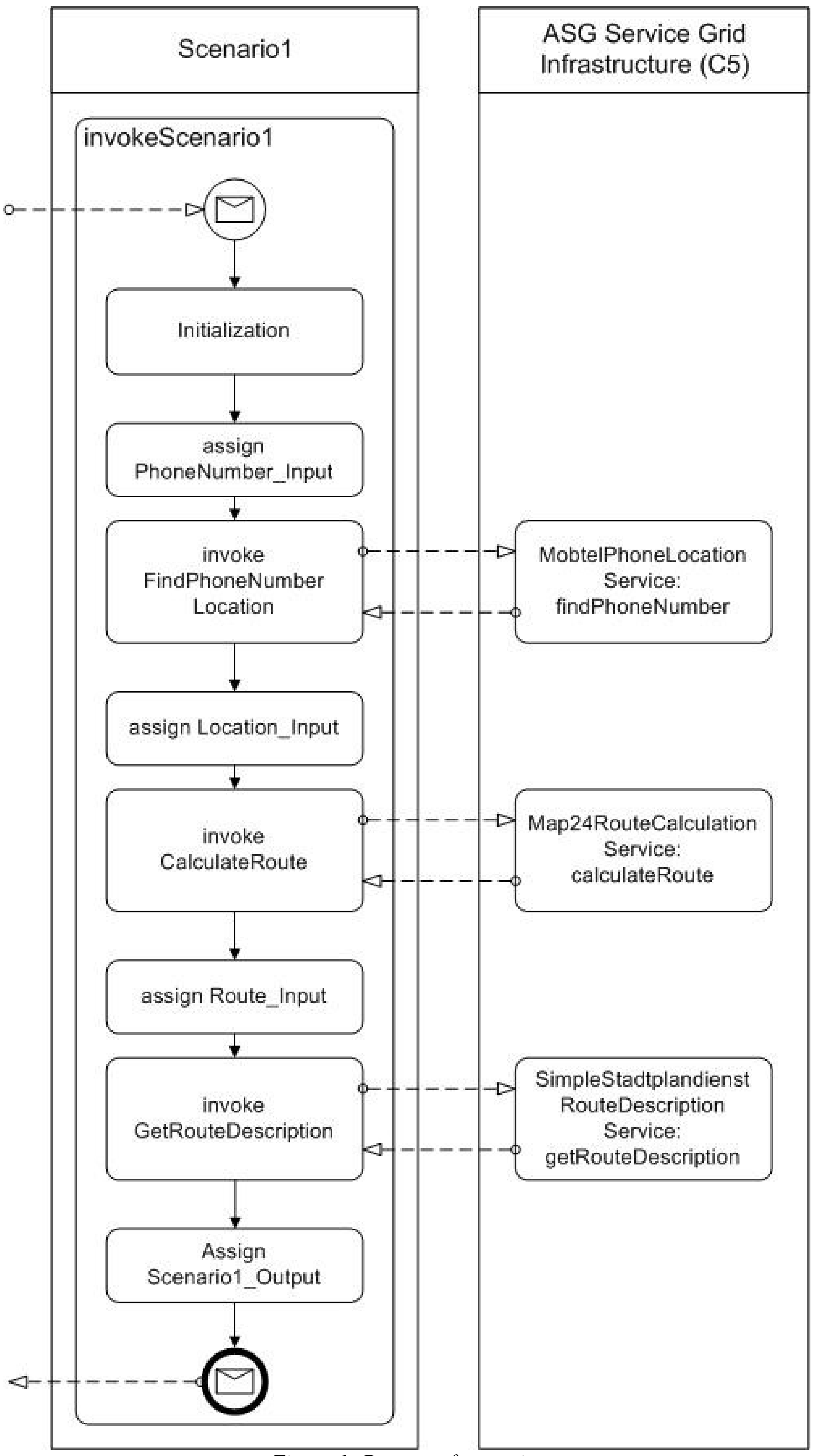
Scenario1
ASG Service Grid Infrastructure (C5)
invokeScenario1
Initialization
assign PhoneNumber_Input
invoke FindPhoneNumber Location
MobtelPhoneLocation Service: findPhoneNumber
assign Location_Input
invoke CalculateRoute
Map24RouteCalculation Service: calculateRoute
assign Route_Input
invoke GetRouteDescription
SimpleStadtplandienst RouteDescription Service: getRouteDescription
Assign Scenario1_Output

Figure 1: Process of scenario one

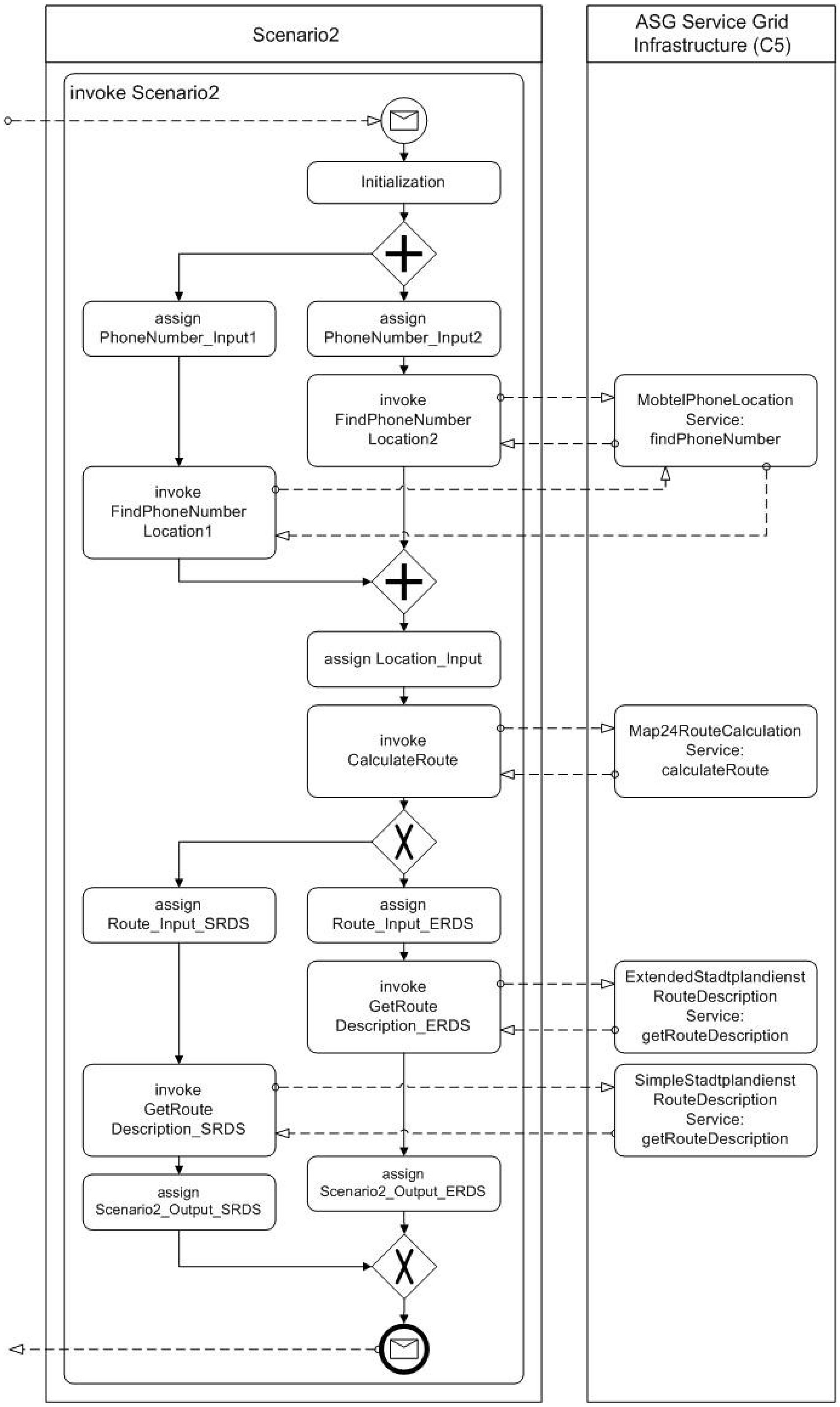
Scenario2
ASG Service Grid Infrastructure (C5)
invoke Scenario2
Initialization
assign PhoneNumber_Input1
assign PhoneNumber_Input2
invoke FindPhoneNumber Location2
MobtelPhoneLocation Service: findPhoneNumber
invoke FindPhoneNumber Location1
assign Location_Input
invoke CalculateRoute
Map24RouteCalculation Service: calculateRoute
assign Route_Input_SRDS
assign Route_Input_ERDS
invoke GetRoute Description_ERDS
ExtendedStadtplandienst RouteDescription Service: getRouteDescription
invoke GetRoute Description_SRDS
SimpleStadtplandienst RouteDescription Service: getRouteDescription
assign Scenario2_Output_ERDS
assign Scenario2_Output_SRDS

Figure 3: Process of scenario two

6.2 Scenario Realization for Twister

To run a process inside Twister one has to deploy an web service description in WSDL describing the service first. The process definition itself can be deployed afterwards. Starting with the easier scenario – scenario one, we tried to deploy its process definition. However, deployment failed. For easier debugging we defined an even more simple process definition consisting of only one web service invocation. Its deployment failed as well. Reason for both failures is the use of <links>. The roadmap of Twister development shows that the basic potential parallel execution semantic of <flow> is already implemented. That does not apply to the link semantic: causal dependencies between activities inside a flow cannot be expressed in Twister so far. Hence deployment of process definitions using link semantic fails. An equivalent process definition describing the same process with a simple sequence block shows the desired result. However, ASG service composition is based upon a graph that is mapped to a BPEL flow. Therefore evaluation of Twister has been discontinued.

6.3 Scenario Realization for FiveSight PXE

Most workflow engines need engine-specific deployment descriptors in addition to the process definition in BPEL and the web service description in WSDL. In the case of PXE a file called pxe-system.xml containing PXE-specific deployment information (see example in appendix) must be added. It is bundled together with the BPEL process definition and the WSDL-file describing the web service provided by the process definition to a .sar package. In BPEL process definitions it is only possible to declare partnerLinks for web services and to specify a portType that is used. The deployment descriptor has to select a concrete binding for partnerLink. The binding contains information about message format (e.g. “SOAP”) and transport protocol (e.g. “HTTP”) [34]. Additionally the deployment descriptor references a concrete URL where the service is located. PXE's descriptor uses a WSDL-file for those deployment information. Deployment consists of several steps: first the BPEL process must be compiled, afterwards packaged with its WSDL-file and deployment descriptor, and then deployed into PXE. PXE ships with a set of command-line tools executing those steps individually. For integration PXE provides Java APIs that can be used for the same purpose.

Both scenarios as specified in the appendix have been successfully deployed and executed with PXE. For web service invocation WS-Addressing has not been used so far. Currently there is no WS-Addressing support in PXE. For use in ASG context implementation of WS-Addressing support is required. Criteria CR2 and CR7 are fulfilled except non-deterministic behaviour while calling one web service twice in parallel execution paths using PXE in version 1.0RC1. Sometimes this parallel execution passed, in other situations it failed. This strange behaviour needs detailed analysis.

7 Integration Concepts for PXE Derived from ASG Requirements

Integration should closely follow the criteria defined in CR8. The scenarios in the previous chapter show that PXE has the potential to fulfil ASG requirements. For an actual integration the interfaces defined in chapter 2 need to be implemented using functionalities of PXE. This chapter draws an outline how integration can be achieved. Each refinement of CR8 is regarded individually. First available interfaces are analysed in respect to CR8-1, CR8-2, CR9 and CR10. The expected runtime environment (CR8-3) must be considered as well. At last extensibility (CR8-4) of PXE is assessed where necessary. This especially applies to the extension in respect to WS-Addressing support.

7.1 PXE Interfaces for ASG C4 Integration

PXE's flexible architecture allows different runtime environments to be used. For each of them an implementation of the Binding API (BAPI) must be provided. PXE ships with an implementation of the BAPI for execution in JBoss application server and for stand-alone execution. If those two versions do not suit application needs, one has to implement the BAPI oneself. From the application developer's perspective the BAPI can be seen as “enabling technology” for all of PXE's APIs. PXE's implementation relies on JMX for process definition deployment and event notification during process enactment and on Java Transaction API (JTA) for transactional process execution. Therefore the BAPI itself must provide a MBeanServer and a TransactionManager for both functionalities. JMX and JTA are documented standards fulfilling CR8-2 adequately [35], [36].

An own implementation of the BAPI is clearly not in scope of our project, therefore a decision has to be made between a JBoss environment and a stand-alone server. Both have a common interface for deploying processes, controlling process execution (e.g. initiation, activity/process termination), and monitoring processes using events. Hence CR9 is completely covered. The interface *enactService()* as specified in chapter 2 can rely on PXE's Deployment and Management API (specifically *com.fs.pxe.swfk.deployment* and *com.fs.pxe.swfk.mngmt*) for deployment and service instance management in combination with service enactment. Service enactment is currently allowed in two different ways: sending SOAP messages over HTTP or sending plain XML messages over a special PXEConnection. Notifications as specified by *register/unregisterEventHandler* can be based upon PXE's BPEL API (*com.fs.pxe.bpel.evt* and *com.fs.pxe.bpel.jmx*). This will sufficiently fulfil CR10. Except “service invocation” all interface requirements from CR8-1 are thus covered. PXE adaptations to allow ASG-conform service invocations are discussed in the following chapter.

7.2 Web Service Invocation using WS-Addressing in PXE

For the usage of WS-Addressing, endpoint references must be declared in either the BPEL process definition or in the workflow engines' process deployment descriptor. None of both ways is currently implemented in PXE. However, PXE's documentation describes how endpoint references can be specified in deployment descriptors. Implementing a WS-Addressing extension for PXE can utilize those property definitions (CR8-4). This option restricts each partner link to use exactly one static endpoint reference. A more flexible option is to assign endpoint references in BPEL using the <assign> activity. Basically an endpoint reference defined as plain XML is copied to a specific partner link. This has the disadvantage of extending BPEL process definitions with information that can potentially only be interpreted by our adapted version of PXE workflow engine. It is not clear whether this syntax conforms to the intended semantics of the BPEL specification. Both ways would provide an interface for service invocation as required by criteria CR8-1.

8 Implementation

This chapter sketches the integration of PXE with the ASG enactment component. Throughout this chapter we will discuss the implementation of an adapter for utilizing PXE in ASG context as well as concepts for supporting the WS-Addressing standard.

8.1 PXE Interface

The integration of PXE with the enactment component requires an interface enabling process deployment, undeployment, execution, instance control, and monitoring. PXE provides a low level, JMX based API that can fulfil nearly all of these requirements (see also CR8-1 and CR8-2).

Considering our overall project goals, we concentrated on the mandatory tasks deployment, undeployment, and execution. Therefore we designed a more abstract API shown in the class diagram in Figure 4.

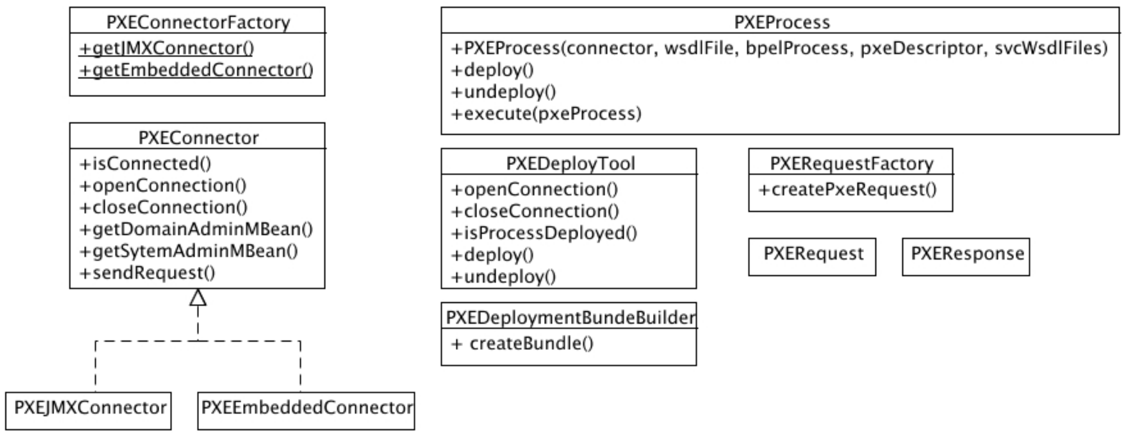

Figure 4: Designed PXE API

PXEProcess forms the main class of our API. The necessary input to construct a new instance encompasses: a *PXEConnector*, a process interface defined in WSDL, the actual process in BPEL4WS notation, a PXE specific deployment descriptor, as well as WSDL interfaces for all web services that will be invoked by the process.

The *PXEConnector* interface encapsulates two possible PXE execution environments. One possible runtime environment is an application server with JMX/MBean support. Alternatively it can run embedded in a JAVA application. This second possibility was especially developed in order to enhance PXE with WS-Addressing support. To create a particular *PXEConnector* of your choice you must use the *PXEConnectorFactory* class.

The process and interface definitions have to be provided as *java.io.InputStream*.

Advanced features like process controlling and monitoring are out of scope of our project.

8.2 WS-Addressing Extension of PXE

In 7.2 we described two approaches of extending PXE with WS-Addressing support. One approach uses BPEL <assign> activities to dynamically assign an endpoint reference to a partner link (detailed information see [32] chapter 7.4). Unfortunately PXE does not support such assignments. A second approach is to provide necessary WS-Addressing information within PXE's deployment descriptor. We already discussed relevant restrictions in 7.2.

Both extensions are possible solutions to realize invocation of stateful web services. However, we decided to implement the second approach because it is anticipated to be less time- and resource consuming.

8.3 ASG-conform Enactment Component

The enactment component is responsible for enacting a service composition [1]. Core of this component should be a BPEL workflow engine. As described in chapters 5 and 6 PXE is the most suitable candidate for our needs.

```
public interface ServiceEnactment {

      public Response enactService(ComposedService composedService,
            Request request) throws ServiceEnactmentException;

      ...
}
```

Listing 2: Enactment interface excerpt

The interface method *enactService(...)* (see Listing 2) triggers the execution of a composed service. A *ComposedService* represents a process definition in BPEL notation extended with endpoint references which contain amongst other parts the interface definitions of the web services to invoke. To deploy and execute such a composed service with the designed interface to PXE (see 8.1) following artifacts must be prepared:

(A1) process interface defined in WSDL

(A2) business process in BPEL4WS notation

(A3) PXE specific deployment descriptor

(A4) WSDL interfaces for all web services that will be invoked

As above-mentioned the *ComposedService* already contains A2 and A4. WSDL interfaces (A4) are included within <partnerLinks> of the process description.

The execution of a deployed PXE process is triggered with SOAP messages which requires a web service definition for the process itself (A1). The more abstract port type of this interface is already part of the composed service. A particular binding (transport and message protocol) and port (location) might be specific for workflow engines. Therefore these elements are generated and added by the enactment component. The main purpose of the PXE deployment descriptor (A3) is the mapping of the abstract port types to certain endpoint ports and bindings. The descriptor can be dynamically created based on the process definition (A2) and the selected services (A4).

After extracting and generating the artifacts A1 – A4, they are assembled to a bundle (SAR-archive) that can be deployed to PXE. The SOAP request message that will trigger PXE to execute the deployed process is build from the *Request* (see Listing 2), the second input parameter of the *enactService(...)* method. It already contains several XML fragments according the ASG users' input parameters.

On process termination PXE will response with a SOAP message containing all output parameters. The Enactment component can build an appropriate Response object (see Listing 2) from it .

8.4 Limitations / Restrictions

The current implementation is restricted to the functionality described above. Handling of errors that occur during process execution, e.g. through service unavailability, is yet not implemented. This lack of error handling implies that intercepting concepts like renegotiation and replanning are not yet feasible.

9 Conclusion

Throughout the evaluation process we narrowed down the potential workflow engines for ASG integration to a single candidate. PXE is the most promising workflow engine in regard to the evaluation criteria defined in chapter 3. All necessary functionalities except WS-Addressing support are currently provided.

The integration of PXE was done in a limited period of time including the extension of WS-Addressing support. However, it provides to the ASG prototype the functionality of process enactment.

Future work mainly spans renegotiation and replanning functionality in case of process failures. Error handling and monitoring features are needed to complement the current enactment component.

References

[1] Laures, G., Jank K., D6.V-1: Reference Architecture: Requirements, Current Efforts and Design; The ASG Project, Europe 2005.

[2] Momotko, M., D4.III-1: Report on the requirements analysis and evaluation of process enactment, monitoring and visualisation; The ASG Project, Europe 2004.

[3] ActiveBPEL, http://www.activebpel.org/ , September 2005.

[4] Apache Agila, http://incubator.apache.org/projects/agila/index.html , September 2005.

[5] AntFlow, http://antflow.onionnetworks.com/ , September 2005.

[6] bexee BPEL Execution Engine, http://bexee.sourceforge.net/ , September 2005.

[7] BigBross Bossa WorkflowSystem, http://www.bigbross.com/bossa/ , September 2005.

[8] con:cern, http://con-cern.org/ , September 2005.

[9] Enhydra Shark, http://shark.objectweb.org/ , September 2005.

[10] FiveSight PXE, http://pxe.fivesight.com/wiki/display/PXE/Home , September 2005.

[11] Freefluo Workflow Enactor, http://freefluo.sourceforge.net/ , September 2005.

[12] JBoss jBPM, http://www.jboss.com/products/jbpm , September 2005.

[13] JFlower, http://sourceforge.net/projects/jflower/ , September 2005.

[14] JFolder, http://www.powerfolder.org/ , September 2005.

[15] MidOffice BPEL Editor, http://mobe.objectweb.org/ , September 2005.

[16] ObjectWeb Bonita, http://bonita.objectweb.org/ , September 2005.

[17] Open For Business - Workflow Engine, http://www.ofbiz.org/docs/workflow.html , September 2005.

[18] OpenSymphony OSWorkflow, http://www.opensymphony.com/osworkflow/ , September 2005.

[19] Open Workflow Engine (OpenWFE), http://web.openwfe.org/display/openwfe , September 2005.

[20] Plaengine, http://plaengine.com , September 2005.

[21] PL/FLOW, http://plflow.sourceforge.net , September 2005.

[22] Syrup, http://syrup.sourceforge.net/ , September 2005.

[23] Twister, http://www.smartcomps.org/twister/ , September 2005.

[24] WfMOpen, http://wfmopen.sourceforge.net/ , September 2005.

[25] Xflow, http://xflow.sourceforge.net/ , September 2005.

[26] Xflow2, http://www.kgionline.com/xflow2/ , September 2005.

[27] YAWL, http://www.yawl.fit.qut.edu.au/ , September 2005.

[28] Zebra Workflow, http://zebra.berlios.de/ , September 2005.

[29] Shapiro, R., A Technical Comparision of XPDL, BPML and BPEL4WS; Cape Visions, 2002.

[30] ASG Project: AttractionBookingService, https://asg-platform.org/cgi-bin/twiki/viewauth/Internal/AttractionBookingServiceSpecification , September 2005.

[31] White, S., IBM, Business Process Modeling Notation (BPMN), Version 1.0; BPMI, 2004.

[32] Andrews, T. et al., Business Process Execution Language for Web Services (BPEL4WS) 1.1; BEA, IBM, Microsoft, SAP, Siebel, 2003.

[33] van der Aalst, W. et al., Workflow Patterns; www.workflowpatterns.com, 2002.

[34] Christensen, E. et al., Web Services Description Language (WSDL) 1.1; W3C (Ariba, IBM, Microsoft), 2001.

[35] Sun Microsystems, Inc., Java Management Extensions Instrumentation and Agent Specification, version 1.2; Sun Microsystems, Inc., 2002.

[36] Cheung, S., Matena, V., Java Transaction API (JTA), version 1.0.1; Sun Microsystems, Inc., 1999.

[37] Green, R., Available BPEL runtime environments, evaluation criteria and evaluation results; RepoMMan Project, September 2005.

APPENDIX

SCENARIO 1 – BPEL PROCESS DEFINITION

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="Scenario1"
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        targetNamespace="http://asg-platform.org/bachelor/scenario1/process"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:asg-pls="http://www.asg-platform.org/scenarios/attractionbooking/services/MobtelPhoneLocationService"
        xmlns:asg-rcs="http://www.asg-platform.org/scenarios/attractionbooking/services/Map24RouteCalculationService"
        xmlns:asg-rds="http://www.asg-platform.org/scenarios/attractionbooking/services/
                        SimpleStadtplandienstRouteDescriptionService"
        xmlns:s1="http://asg-platform.org/bachelor/scenario1/wsdl"
        xmlns:asg-cts="http://asg-platform.org/scenarios/attractionbooking/contacts"
        xmlns:asg-rts="http://asg-platform.org/scenarios/attractionbooking/routes"
        suppressJoinFailure="yes">

        <partnerLinks>
            <partnerLink name="Scenario1PL" myRole="Scenario1Provider" partnerLinkType="s1:Scenario1PLT"/>
            <partnerLink name="PhoneLocationPL" partnerRole="PhoneLocationProvider"
                partnerLinkType="s1:PhoneLocationPLT"/>
            <partnerLink name="RouteCalculationPL" partnerRole="RouteProvider"
                partnerLinkType="s1:CalculateRoutePLT"/>
            <partnerLink name="RouteDecriptionPL" partnerRole="RouteDescriptionProvider"
                partnerLinkType="s1:RouteDecriptionPLT"/>
        </partnerLinks>

        <variables>
            <variable name="Scenario1_Input" messageType="s1:Scenario1_Request"/>
            <variable name="PhoneNumber_Input" messageType="asg-pls:PhoneLocationEndpoint_findPhoneLocation"/>
            <variable name="Location_Output"
                messageType="asg-pls:PhoneLocationEndpoint_findPhoneLocationResponse"/>
            <variable name="Location_Input" messageType="asg-rcs:RouteCalculationEndpoint_calculateRoute"/>
            <variable name="Route_Output" messageType="asg-rcs:RouteCalculationEndpoint_calculateRouteResponse"/>
            <variable name="Route_Input" messageType="asg-rds:RouteDescriptionEndpoint_getRouteDescription"/>
            <variable name="RouteDescription_Output"
                messageType="asg-rds:RouteDescriptionEndpoint_getRouteDescriptionResponse"/>
            <variable name="Scenario1_Output" messageType="s1:Scenario1_Response"/>
        </variables>

        <flow name="Scenario1Flow">
            <links>
                <link name="ReceiveScenario1_Input-To-Initialization"/>
                <link name="Initialization-To-AssignPhoneNumber_Input"/>
                <link name="AssignPhoneNumber_Input-To-InvokeFindPhoneLocation"/>
                <link name="InvokeFindPhoneLocation-To-AssignLocation_Input"/>
                <link name="AssignLocation_Input-To-InvokeCalculateRoute"/>
                <link name="InvokeCalculateRoute-To-AssignRoute_Input"/>
                <link name="AssignRoute_Input-To-InvokeGetRouteDescription"/>
                <link name="InvokeGetRouteDescription-To-AssignScenario1_Output"/>
                <link name="AssignScenario1_Output-ReplyScenario1_Output"/>
            </links>

            <receive createInstance="yes" name="startScenario1" variable="Scenario1_Input"
                operation="invokeScenario1" partnerLink="Scenario1PL" portType="s1:Scenario1PT">
                <source linkName="ReceiveScenario1_Input-To-Initialization"/>
            </receive>
```

```
<assign name="Initialization">
    <target linkName="ReceiveScenario1_Input-To-Initialization"/>
    <source linkName="Initialization-To-AssignPhoneNumber_Input"/>
    <copy>
        <from>
            <asg-pls:findPhoneLocation>
                <asg-pls:phoneNumber>
                     <asg-cts:countryCode/>
                     <asg-cts:areaCode/>
                     <asg-cts:number/>
                </asg-pls:phoneNumber>
            </asg-pls:findPhoneLocation>
         </from>
        <to variable="PhoneNumber_Input" part="parameters"/>
    </copy>
    <copy>
        <from>
            <asg-rcs:calculateRoute>
                <asg-rcs:location/>
                <asg-rcs:location/>
            </asg-rcs:calculateRoute>
        </from>
        <to variable="Location_Input" part="parameters"/>
    </copy>
    <copy>
        <from>
            <asg-rds:getRouteDescription>
                <asg-rds:route/>
            </asg-rds:getRouteDescription>
        </from>
        <to variable="Route_Input" part="parameters"/>
    </copy>
    <copy>
        <from>
            <s1:routeDescription>
                <asg-rts:map/>
                <asg-rts:description/>
            </s1:routeDescription>
        </from>
        <to variable="Scenario1_Output" part="parameters"/>
    </copy>
</assign>

<assign name="AssignPhoneNumber_Input">
    <target linkName="Initialization-To-AssignPhoneNumber_Input"/>
    <source linkName="AssignPhoneNumber_Input-To-InvokeFindPhoneLocation"/>
    <copy>
        <from variable="Scenario1_Input" part="parameters" query="//s1:scenario1Input/s1:phoneNumber"/>
        <to variable="PhoneNumber_Input" part="parameters"
            query="//asg-pls:findPhoneLocation/asg-pls:phoneNumber"/>
    </copy>
</assign>

<invoke name="invokeFindPhoneNumberLocation" operation="findPhoneLocation"
    inputVariable="PhoneNumber_Input" outputVariable="Location_Output"
    partnerLink="PhoneLocationPL" portType="asg-pls:PhoneLocationEndpoint">
    <target linkName="AssignPhoneNumber_Input-To-InvokeFindPhoneLocation"/>
    <source linkName="InvokeFindPhoneLocation-To-AssignLocation_Input"/>
</invoke>
```

```
        <assign name="AssignLocation_Input">
            <target linkName="InvokeFindPhoneLocation-To-AssignLocation_Input"/>
            <source linkName="AssignLocation_Input-To-InvokeCalculateRoute"/>
            <copy>
                <from variable="Location_Output" part="result" query="//asg-pls:findPhoneLocationResponse/location"/>
                <to variable="Location_Input" part="parameters"
                    query="//asg-rcs:calculateRoute/asg-rcs:location[position()=1]"/>
            </copy>
            <copy>
                <from variable="Scenario1_Input" part="parameters" query="//s1:scenario1Input/s1:location"/>
                <to variable="Location_Input" part="parameters"
                    query="//asg-rcs:calculateRoute/asg-rcs:location[position()=2]"/>
            </copy>
        </assign>

        <invoke name="invokeCalculateRoute" operation="calculateRoute"
            inputVariable="Location_Input" outputVariable="Route_Output"
            partnerLink="RouteCalculationPL" portType="asg-rcs:RouteCalculationEndpoint">
            <target linkName="AssignLocation_Input-To-InvokeCalculateRoute"/>
            <source linkName="InvokeCalculateRoute-To-AssignRoute_Input"/>
        </invoke>

        <assign name="AssignRoute_Input">
            <target linkName="InvokeCalculateRoute-To-AssignRoute_Input"/>
            <source linkName="AssignRoute_Input-To-InvokeGetRouteDescription"/>
            <copy>
                <from variable="Route_Output" part="result" query="//asg-rcs:calculateRouteResponse/route"/>
                <to variable="Route_Input" part="parameters" query="//asg-rds:getRouteDescription/asg-rds:route"/>
            </copy>
        </assign>

        <invoke name="invokeGetRouteDescription" operation="getRouteDescription"
            inputVariable="Route_Input" outputVariable="RouteDescription_Output"
            partnerLink="RouteDecriptionPL" portType="asg-rds:RouteDescriptionEndpoint">
            <target linkName="AssignRoute_Input-To-InvokeGetRouteDescription"/>
            <source linkName="InvokeGetRouteDescription-To-AssignScenario1_Output"/>
        </invoke>

        <assign name="AssignScenario1_Output">
            <target linkName="InvokeGetRouteDescription-To-AssignScenario1_Output"/>
            <source linkName="AssignScenario1_Output-ReplyScenario1_Output"/>
            <copy>
                <from variable="RouteDescription_Output" part="result" query="//asg-
                    rds:getRouteDescriptionResponse/routeDescription"/>
                <to variable="Scenario1_Output" part="parameters" query="//s1:routeDescription"/>
            </copy>
        </assign>

        <reply name="ReplyScenario1_Output" variable="Scenario1_Output" operation="invokeScenario1"
            partnerLink="Scenario1PL" portType="s1:Scenario1PT">
            <target linkName="AssignScenario1_Output-ReplyScenario1_Output"/>
        </reply>

    </flow>

</process>
```

Scenario 1 – WSDL

```
<definitions name="Scenario1.wsdl"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        targetNamespace="http://asg-platform.org/bachelor/scenario1/wsdl"
        xmlns:tns="http://asg-platform.org/bachelor/scenario1/wsdl"
        xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:asg-pls="http://www.asg-platform.org/scenarios/attractionbooking/services/MobtelPhoneLocationService"
        xmlns:asg-rcs="http://www.asg-platform.org/scenarios/attractionbooking/services/Map24RouteCalculationService"
        xmlns:asg-rds="http://www.asg-
platform.org/scenarios/attractionbooking/services/SimpleStadtplandienstRouteDescriptionService">

        <import namespace="http://www.asg-
platform.org/scenarios/attractionbooking/services/MobtelPhoneLocationService" location="..."/>
        <import namespace="http://www.asg-
platform.org/scenarios/attractionbooking/services/Map24RouteCalculationService" location="..."/>
        <import namespace="http://www.asg-
platform.org/scenarios/attractionbooking/services/SimpleStadtplandienstRouteDescriptionService"
              location="..."/>

        <types>
              <xsd:schema targetNamespace="http://asg-platform.org/bachelor/scenario1/wsdl"
                    xmlns:asg-cts="http://asg-platform.org/scenarios/attractionbooking/contacts"
                    xmlns:asg-rts="http://asg-platform.org/scenarios/attractionbooking/routes">
                    <xsd:import namespace="http://asg-platform.org/scenarios/attractionbooking/routes" schemaLocation="..."/>
                    <xsd:import namespace="http://asg-platform.org/scenarios/attractionbooking/contacts"
                          schemaLocation="..."/>
                    <xsd:complexType name="Scenario1Input">
                          <xsd:sequence>
                                <xsd:element name="phoneNumber" type="asg-cts:PhoneNumber"/>
                                <xsd:element name="location" type="asg-rts:Location"/>
                          </xsd:sequence>
                    </xsd:complexType>
                    <xsd:element name="scenario1Input" type="tns:Scenario1Input"/>
                    <xsd:element name="routeDescription" type="asg-rts:RouteDescription"/>
              </xsd:schema>
        </types>

        <message name="Scenario1_Request">
              <part name="parameters" element="tns:scenario1Input"/>
        </message>
        <message name="Scenario1_Response">
              <part name="parameters" element="tns:routeDescription"/>
        </message>

        <portType name="Scenario1PT">
              <operation name="invokeScenario1">
                    <input message="tns:Scenario1_Request"/>
                    <output message="tns:Scenario1_Response"/>
              </operation>
        </portType>

        <binding name="Scenario1ServiceBinding" type="tns:Scenario1PT">
              <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
              <operation name="invokeScenario1">
                    <soap:operation soapAction=""/>
                    <input><soap:body use="literal"/></input>
                    <output>   <soap:body use="literal"/></output>
              </operation>
```

```
    </binding>
    <service name="Scenario1">
        <port binding="tns:Scenario1ServiceBinding" name="Scenario1Port">
            <soap:address location="..."/>
        </port>
    </service>

    <plnk:partnerLinkType name="Scenario1PLT">
        <plnk:role name="Scenario1Provider">
            <plnk:portType name="tns:Scenario1PT"/>
        </plnk:role>
    </plnk:partnerLinkType>

    <plnk:partnerLinkType name="PhoneLocationPLT">
        <plnk:role name="PhoneLocationProvider">
            <plnk:portType name="asg-pls:PhoneLocationEndpoint"/>
        </plnk:role>
    </plnk:partnerLinkType>

    <plnk:partnerLinkType name="CalculateRoutePLT">
        <plnk:role name="RouteProvider">
            <plnk:portType name="asg-rcs:RouteCalculationEndpoint"/>
        </plnk:role>
    </plnk:partnerLinkType>

    <plnk:partnerLinkType name="RouteDecriptionPLT">
        <plnk:role name="RouteDescriptionProvider">
            <plnk:portType name="asg-rds:RouteDescriptionEndpoint"/>
        </plnk:role>
    </plnk:partnerLinkType>

</definitions>
```

Scenario 2 – BPEL Process Defnition

```
<process name="Scenario2"
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        targetNamespace="http://asg-platform.org/bachelor/scenario2/process"
        xmlns:tns="http://asg-platform.org/bachelor/scenario2/process"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:asg-pls="http://www.asg-platform.org/scenarios/attractionbooking/services/MobtelPhoneLocationService"
        xmlns:asg-rcs="http://www.asg-platform.org/scenarios/attractionbooking/services/Map24RouteCalculationService"
        xmlns:asg-srds="http://www.asg-platform.org/scenarios/attractionbooking/services/
                        SimpleStadtplandienstRouteDescriptionService"
        xmlns:asg-erds="http://www.asg-platform.org/scenarios/attractionbooking/services/
                        ExtendedStadtplandienstRouteDescriptionService"
        xmlns:s2="http://asg-platform.org/bachelor/scenario2/wsdl"
        xmlns:asg-cts="http://asg-platform.org/scenarios/attractionbooking/contacts"
        xmlns:asg-rts="http://asg-platform.org/scenarios/attractionbooking/routes"
        suppressJoinFailure="no">

        <partnerLinks>
            <partnerLink name="Scenario2PL" myRole="Scenario2Provider" partnerLinkType="s2:Scenario2PLT"/>
            <partnerLink name="PhoneLocationPL" partnerRole="PhoneLocationProvider"
                partnerLinkType="s2:PhoneLocationPLT"/>
            <partnerLink name="RouteCalculationPL" partnerRole="RouteProvider"
                partnerLinkType="s2:CalculateRoutePLT"/>
            <partnerLink name="SimpleRouteDecriptionPL" partnerRole="SimpleRouteDescriptionProvider"
                partnerLinkType="s2:SimpleRouteDecriptionPLT"/>
            <partnerLink name="ExtendedRouteDecriptionPL" partnerRole="ExtendedRouteDescriptionProvider"
                partnerLinkType="s2:ExtendedRouteDecriptionPLT"/>
        </partnerLinks>

        <variables>
            <variable name="Scenario2_Input" messageType="s2:Scenario2_Request"/>
            <variable name="PhoneNumber_Input1" messageType="asg-pls:PhoneLocationEndpoint_findPhoneLocation"/>
            <variable name="PhoneNumber_Input2" messageType="asg-pls:PhoneLocationEndpoint_findPhoneLocation"/>
            <variable name="Location_Output1" messageType="asg-pls:
                PhoneLocationEndpoint_findPhoneLocationResponse"/>
            <variable name="Location_Output2" messageType="asg-pls:
                PhoneLocationEndpoint_findPhoneLocationResponse"/>
            <variable name="Location_Input" messageType="asg-rcs:RouteCalculationEndpoint_calculateRoute"/>
            <variable name="Route_Output" messageType="asg-rcs:RouteCalculationEndpoint_calculateRouteResponse"/>
            <variable name="Route_Input_SRDS" messageType="asg-srds:
                RouteDescriptionEndpoint_getRouteDescription"/>
            <variable name="RouteDescription_Output_SRDS"
                messageType="asg-srds:RouteDescriptionEndpoint_getRouteDescriptionResponse"/>
            <variable name="Route_Input_ERDS" messageType="asg-erds:
                RouteDescriptionEndpoint_getRouteDescription"/>
            <variable name="RouteDescription_Output_ERDS"
                messageType="asg-erds:RouteDescriptionEndpoint_getRouteDescriptionResponse"/>
            <variable name="Scenario2_Output" messageType="s2:Scenario2_Response"/>
        </variables>

        <flow name="Scenario2Flow">
            <links>
                <link name="ReceiveScenario2_Input-To-Initialization"/>
                <link name="Initialization-To-AssignPhoneNumber_Input1"/>
                <link name="Initialization-To-AssignPhoneNumber_Input2"/>
                <link name="AssignPhoneNumber_Input1-To-InvokeFindPhoneLocation1"/>
                <link name="AssignPhoneNumber_Input2-To-InvokeFindPhoneLocation2"/>
                <link name="InvokeFindPhoneLocation1-To-AssignLocation_Input"/>
                <link name="InvokeFindPhoneLocation2-To-AssignLocation_Input"/>
```

```
    <link name="AssignLocation_Input-To-InvokeCalculateRoute"/>
    <link name="InvokeCalculateRoute-To-AssignRoute_Input_SRDS"/>
    <link name="InvokeCalculateRoute-To-AssignRoute_Input_ERDS"/>
    <link name="AssignRoute_Input_SRDS-To-InvokeGetRouteDescription_SRDS"/>
    <link name="AssignRoute_Input_ERDS-To-InvokeGetRouteDescription_ERDS"/>
    <link name="InvokeGetRouteDescription_SRDS-To-AssignScenario2_Output_SRDS"/>
    <link name="InvokeGetRouteDescription_ERDS-To-AssignScenario2_Output_ERDS"/>
    <link name="AssignScenario2_Output_SRDS-ReplyScenario2_Output"/>
    <link name="AssignScenario2_Output_ERDS-ReplyScenario2_Output"/>
</links>

<receive createInstance="yes" name="ReceiveScenario2_Input" variable="Scenario2_Input"
    operation="invokeScenario2" partnerLink="Scenario2PL" portType="s2:Scenario2PT">
    <source linkName="ReceiveScenario2_Input-To-Initialization"/>
</receive>

<assign name="Initialization">
    <target linkName="ReceiveScenario2_Input-To-Initialization"/>
    <source linkName="Initialization-To-AssignPhoneNumber_Input1"/>
    <source linkName="Initialization-To-AssignPhoneNumber_Input2"/>
    <copy>
        <from>
            <asg-pls:findPhoneLocation>
                <asg-pls:phoneNumber/>
            </asg-pls:findPhoneLocation>
        </from>
        <to variable="PhoneNumber_Input1" part="parameters"/>
    </copy>
    <copy>
        <from>
            <asg-pls:findPhoneLocation>
                <asg-pls:phoneNumber/>
            </asg-pls:findPhoneLocation>
        </from>
        <to variable="PhoneNumber_Input2" part="parameters"/>
    </copy>
    <copy>
        <from>
            <asg-rcs:calculateRoute>
                <asg-rcs:location/>
                <asg-rcs:location/>
             </asg-rcs:calculateRoute>
        </from>
        <to variable="Location_Input" part="parameters"/>
    </copy>
    <copy>
        <from>
            <asg-srds:getRouteDescription>
                <asg-srds:route/>
            </asg-srds:getRouteDescription>
        </from>
        <to variable="Route_Input_SRDS" part="parameters"/>
    </copy>
    <copy>
        <from>
            <asg-erds:getRouteDescription>
                <asg-erds:route/>
            </asg-erds:getRouteDescription>
        </from>
        <to variable="Route_Input_ERDS" part="parameters"/>
    </copy>
```

```
        <copy>
            <from>
                <s2:routeDescription>
                    <asg-rts:map/>
                    <asg-rts:description/>
                </s2:routeDescription>
             </from>
            <to variable="Scenario2_Output" part="parameters"/>
        </copy>
    </assign>

    <assign name="AssignPhoneNumber_Input1">
        <target linkName="Initialization-To-AssignPhoneNumber_Input1"/>
        <source linkName="AssignPhoneNumber_Input1-To-InvokeFindPhoneLocation1"/>
        <copy>
            <from variable="Scenario2_Input" part="parameters"
                query="//s2:scenario2Input/s2:phoneNumber[position()=1]"/>
            <to variable="PhoneNumber_Input1" part="parameters"
                query="//asg-pls:findPhoneLocation/asg-pls:phoneNumber"/>
        </copy>
    </assign>

    <invoke name="invokeFindPhoneNumberLocation1" operation="findPhoneLocation"
        inputVariable="PhoneNumber_Input1" outputVariable="Location_Output1"
        partnerLink="PhoneLocationPL" portType="asg-pls:PhoneLocationEndpoint">
        <target linkName="AssignPhoneNumber_Input1-To-InvokeFindPhoneLocation1"/>
        <source linkName="InvokeFindPhoneLocation1-To-AssignLocation_Input"/>
    </invoke

    <assign name="AssignPhoneNumber_Input2">
        <target linkName="Initialization-To-AssignPhoneNumber_Input2"/>
        <source linkName="AssignPhoneNumber_Input2-To-InvokeFindPhoneLocation2"/>
        <copy>
            <from variable="Scenario2_Input" part="parameters"
                query="//s2:scenario2Input/s2:phoneNumber[position()=2]"/>
            <to variable="PhoneNumber_Input2" part="parameters"
                query="//asg-pls:findPhoneLocation/asg-pls:phoneNumber"/>
        </copy>
    </assign>

    <invoke name="invokeFindPhoneNumberLocation2" operation="findPhoneLocation"
        inputVariable="PhoneNumber_Input2" outputVariable="Location_Output2"
        partnerLink="PhoneLocationPL" portType="asg-pls:PhoneLocationEndpoint">
        <target linkName="AssignPhoneNumber_Input2-To-InvokeFindPhoneLocation2"/>
        <source linkName="InvokeFindPhoneLocation2-To-AssignLocation_Input"/>
    </invoke>

    <assign name="AssignLocation_Input"
        joinCondition="bpws:getLinkStatus('InvokeFindPhoneLocation1-To-AssignLocation_Input')
        and bpws:getLinkStatus('InvokeFindPhoneLocation2-To-AssignLocation_Input')">
        <target linkName="InvokeFindPhoneLocation1-To-AssignLocation_Input"/>
        <target linkName="InvokeFindPhoneLocation2-To-AssignLocation_Input"/>
        <source linkName="AssignLocation_Input-To-InvokeCalculateRoute"/>
        <copy>
            <from variable="Location_Output1" part="result"
                query="//asg-pls:findPhoneLocationResponse/location"/>
            <to variable="Location_Input" part="parameters"
                query="//asg-rcs:calculateRoute/asg-rcs:location[position()=1]"/>
        </copy>
        <copy>
        <from variable="Location_Output2" part="result"
            query="//asg-pls:findPhoneLocationResponse/location"/>
        <to variable="Location_Input" part="parameters"
            query="//asg-rcs:calculateRoute/asg-rcs:location[position()=2]"/>
    </copy>
</assign>
<invoke name="invokeCalculateRoute" operation="calculateRoute"
    inputVariable="Location_Input" outputVariable="Route_Output"
    partnerLink="RouteCalculationPL" portType="asg-rcs:RouteCalculationEndpoint">
    <target linkName="AssignLocation_Input-To-InvokeCalculateRoute"/>
    <source linkName="InvokeCalculateRoute-To-AssignRoute_Input_SRDS" transitionCondition=
        "bpws:getVariableData('Scenario2_Input', 'parameters', '//s2:scenario2Input/s2:extendedDescription') =
        'false'"/>
    <source linkName="InvokeCalculateRoute-To-AssignRoute_Input_ERDS" transitionCondition=
        "bpws:getVariableData('Scenario2_Input', 'parameters', '//s2:scenario2Input/s2:extendedDescription') !=
        'false'"/>
</invoke>

<assign name="AssignRoute_Input_SRDS" suppressJoinFailure="yes">
    <target linkName="InvokeCalculateRoute-To-AssignRoute_Input_SRDS"/>
    <source linkName="AssignRoute_Input_SRDS-To-InvokeGetRouteDescription_SRDS"/>
    <copy>
        <from variable="Route_Output" part="result" query="//asg-rcs:calculateRouteResponse/route"/>
        <to variable="Route_Input_SRDS" part="parameters"
            query="//asg-srds:getRouteDescription/asg-srds:route"/>
    </copy>
</assign>

<invoke name="invokeGetRouteDescription_SRDS" operation="getRouteDescription"
    inputVariable="Route_Input_SRDS" outputVariable="RouteDescription_Output_SRDS"
    partnerLink="SimpleRouteDecriptionPL" portType="asg-srds:RouteDescriptionEndpoint"
    suppressJoinFailure="yes">
    <target linkName="AssignRoute_Input_SRDS-To-InvokeGetRouteDescription_SRDS"/>
    <source linkName="InvokeGetRouteDescription_SRDS-To-AssignScenario2_Output_SRDS"/>
</invoke>

<assign name="AssignScenario2_Output_SRDS" suppressJoinFailure="yes">
    <target linkName="InvokeGetRouteDescription_SRDS-To-AssignScenario2_Output_SRDS"/>
    <source linkName="AssignScenario2_Output_SRDS-ReplyScenario2_Output"/>
    <copy>
        <from variable="RouteDescription_Output_SRDS" part="result"
            query="//asg-srds:getRouteDescriptionResponse/routeDescription"/>
        <to variable="Scenario2_Output" part="parameters" query="//s2:routeDescription"/>
    </copy>
</assign>

<assign name="AssignRoute_Input_ERDS" suppressJoinFailure="yes">
    <target linkName="InvokeCalculateRoute-To-AssignRoute_Input_ERDS"/>
    <source linkName="AssignRoute_Input_ERDS-To-InvokeGetRouteDescription_ERDS"/>
    <copy>
        <from variable="Route_Output" part="result" query="//asg-rcs:calculateRouteResponse/route"/>
        <to variable="Route_Input_ERDS" part="parameters"
            query="//asg-erds:getRouteDescription/asg-erds:route"/>
    </copy>
</assign>

<invoke name="invokeGetRouteDescription_ERDS" operation="getRouteDescription"
    inputVariable="Route_Input_ERDS" outputVariable="RouteDescription_Output_ERDS"
    partnerLink="ExtendedRouteDecriptionPL" portType="asg-erds:RouteDescriptionEndpoint"
    suppressJoinFailure="yes">
    <target linkName="AssignRoute_Input_ERDS-To-InvokeGetRouteDescription_ERDS"/>
    <source linkName="InvokeGetRouteDescription_ERDS-To-AssignScenario2_Output_ERDS"/>
```

```
        </invoke>

        <assign name="AssignScenario2_Output_ERDS" suppressJoinFailure="yes">
            <target linkName="InvokeGetRouteDescription_ERDS-To-AssignScenario2_Output_ERDS"/>
            <source linkName="AssignScenario2_Output_ERDS-ReplyScenario2_Output"/>

            <copy>
                <from variable="RouteDescription_Output_ERDS" part="result"
                    query="//asg-erds:getRouteDescriptionResponse/routeDescription"/>
                <to variable="Scenario2_Output" part="parameters" query="//s2:routeDescription"/>
            </copy>
        </assign>

        <reply name="ReplyScenario2_Output" variable="Scenario2_Output" operation="invokeScenario2"
            partnerLink="Scenario2PL" portType="s2:Scenario2PT">
            <target linkName="AssignScenario2_Output_SRDS-ReplyScenario2_Output"/>
            <target linkName="AssignScenario2_Output_ERDS-ReplyScenario2_Output"/>
        </reply>

    </flow>

</process>
```

Scenario 2 – WSDL

```
<definitions name="Scenario2.wsdl"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        targetNamespace="http://asg-platform.org/bachelor/scenario2/wsdl"
        xmlns:tns="http://asg-platform.org/bachelor/scenario2/wsdl"
        xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:asg-pls="http://www.asg-platform.org/scenarios/attractionbooking/services/MobtelPhoneLocationService"
        xmlns:asg-rcs="http://www.asg-platform.org/scenarios/attractionbooking/services/Map24RouteCalculationService"
        xmlns:asg-srds="http://www.asg-platform.org/scenarios/attractionbooking/services/
            SimpleStadtplandienstRouteDescriptionService"
        xmlns:asg-erds="http://www.asg-platform.org/scenarios/attractionbooking/services/
            ExtendedStadtplandienstRouteDescriptionService">
        <import namespace="http://www.asg-platform.org/scenarios/attractionbooking/services/
            MobtelPhoneLocationService" location="..."/>
        <import namespace="http://www.asg-platform.org/scenarios/attractionbooking/services/
            Map24RouteCalculationService" location="..."/>
        <import namespace="http://www.asg-platform.org/scenarios/attractionbooking/services/
            SimpleStadtplandienstRouteDescriptionService"
            location="..."/>
        <import namespace="http://www.asg-platform.org/scenarios/attractionbooking/services/
            ExtendedStadtplandienstRouteDescriptionService" location="..."/>

        <types>
            <xsd:schema targetNamespace="http://asg-platform.org/bachelor/scenario2/wsdl"
                xmlns:asg-cts="http://asg-platform.org/scenarios/attractionbooking/contacts"
                xmlns:asg-rts="http://asg-platform.org/scenarios/attractionbooking/routes">
                <xsd:import namespace="http://asg-platform.org/scenarios/attractionbooking/routes" schemaLocation="..."/>
                <xsd:import namespace="http://asg-platform.org/scenarios/attractionbooking/contacts"
                    schemaLocation="..."/>
                <xsd:complexType name="Scenario2Input">
                    <xsd:sequence>
                      <xsd:element name="phoneNumber" type="asg-cts:PhoneNumber" minOccurs="2" maxOccurs="2" />
                      <xsd:element name="extendedDescription" type="xsd:boolean" />
                    </xsd:sequence>
                </xsd:complexType>
                <xsd:element name="scenario2Input" type="tns:Scenario2Input"/>
                <xsd:element name="routeDescription" type="asg-rts:RouteDescription"/>
            </xsd:schema>
        </types>

        <message name="Scenario2_Request">
            <part name="parameters" element="tns:scenario2Input"/>
        </message>
        <message name="Scenario2_Response">
            <part name="parameters" element="tns:routeDescription"/>
        </message>

        <portType name="Scenario2PT">
            <operation name="invokeScenario2">
                <input message="tns:Scenario2_Request"/>
                <output message="tns:Scenario2_Response"/>
            </operation>
        </portType>
```

```
    <binding name="Scenario2ServiceBinding" type="tns:Scenario2PT">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="invokeScenario2">
            <soap:operation soapAction=""/>
            <input><soap:body use="literal"/></input>
            <output><soap:body use="literal"/></output>
        </operation>
    </binding>

    <service name="Scenario2">
        <port binding="tns:Scenario2ServiceBinding" name="Scenario2Port">
            <soap:address location="..."/>
        </port>
    </service>

    <plnk:partnerLinkType name="Scenario2PLT">
        <plnk:role name="Scenario2Provider">
            <plnk:portType name="tns:Scenario2PT"/>
        </plnk:role>
    </plnk:partnerLinkType>

    <plnk:partnerLinkType name="PhoneLocationPLT">
        <plnk:role name="PhoneLocationProvider">
            <plnk:portType name="asg-pls:PhoneLocationEndpoint"/>
        </plnk:role>
    </plnk:partnerLinkType>

    <plnk:partnerLinkType name="CalculateRoutePLT">
        <plnk:role name="RouteProvider">
            <plnk:portType name="asg-rcs:RouteCalculationEndpoint"/>
        </plnk:role>
    </plnk:partnerLinkType>

    <!-- Simple RouteDescriptionService invoked -->
    <plnk:partnerLinkType name="SimpleRouteDecriptionPLT">
        <plnk:role name="SimpleRouteDescriptionProvider">
            <plnk:portType name="asg-srds:RouteDescriptionEndpoint"/>
        </plnk:role>
    </plnk:partnerLinkType>

    <plnk:partnerLinkType name="ExtendedRouteDecriptionPLT">
        <plnk:role name="ExtendedRouteDescriptionProvider">
            <plnk:portType name="asg-erds:RouteDescriptionEndpoint"/>
        </plnk:role>
    </plnk:partnerLinkType>

</definitions>
```

Example PXE Deployment Descriptor

```
<system-descriptor name="Scenario2"
  wsdlUri="file:Scenario2.wsdl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.fivesight.com/pxe/system-descriptor/ http://www.fivesight.com/pxe/system-descriptor/"
  xmlns="http://www.fivesight.com/pxe/system-descriptor/"
  xmlns:s2="http://asg-platform.org/bachelor/scenario2/wsdl"
  xmlns:asg-pls="http://www.asg-platform.org/scenarios/attractionbooking/services/MobtelPhoneLocationService"
  xmlns:asg-rcs="http://www.asg-platform.org/scenarios/attractionbooking/services/Map24RouteCalculationService"
  xmlns:asg-srds="http://www.asg-
platform.org/scenarios/attractionbooking/services/SimpleStadtplandienstRouteDescriptionService"
  xmlns:asg-erds="http://www.asg-
platform.org/scenarios/attractionbooking/services/ExtendedStadtplandienstRouteDescriptionService">

  <channels>
    <channel name="inboundChannel" />
    <channel name="outboundPhoneLocationChannel" />
    <channel name="outboundRouteCalculationChannel" />
    <channel name="outboundSimpleRouteDescriptionChannel" />
    <channel name="outboundExtendedRouteDescriptionChannel" />
  </channels>

  <services>

    <service name="Scenario2" provider="uri:protocoladapter.soap">
      <properties>
        <property name="serviceWsdlNamespace" value="http://asg-platform.org/bachelor/scenario2/wsdl"/>
        <property name="serviceWsdlName" value="Scenario2"/>
      </properties>
      <imports>
        <port name="Scenario2Port" type="s2:Scenario2PT" channel-ref="inboundChannel"/>
      </imports>
    </service>

    <service name="MobtelPhoneLocationService" provider="uri:protocoladapter.soap">
      <properties>
        <property name="concreteWsdlUrl" value="..."/>
        <property name="serviceWsdlNamespace"
         value="http://www.asg-platform.org/scenarios/attractionbooking/services/MobtelPhoneLocationService"/>
        <property name="serviceWsdlName" value="MobtelPhoneLocationService"/>
      </properties>
      <exports>
        <port name="PhoneLocationEndpointPort" type="asg-pls:PhoneLocationEndpoint" channel-
ref="outboundPhoneLocationChannel"/>
      </exports>
    </service>

    <service name="Map24RouteCalculationService" provider="uri:protocoladapter.soap">
      <properties>
        <property name="concreteWsdlUrl" value="..."/>
        <property name="serviceWsdlNamespace"
         value="http://www.asg-platform.org/scenarios/attractionbooking/services/Map24RouteCalculationService"/>
        <property name="serviceWsdlName" value="Map24RouteCalculationService"/>
      </properties>
      <exports>
        <port name="RouteCalculationEndpointPort" type="asg-rcs:RouteCalculationEndpoint" channel-
ref="outboundRouteCalculationChannel"/>
      </exports>
    </service>
    <service name="SimpleStadtplandienstRouteDescriptionService" provider="uri:protocoladapter.soap">
      <properties>
        <property name="concreteWsdlUrl" value="..."/>
        <property name="serviceWsdlNamespace"
         value="http://www.asg-platform.org/scenarios/attractionbooking/services/
               SimpleStadtplandienstRouteDescriptionService"/>
        <property name="serviceWsdlName" value="SimpleStadtplandienstRouteDescriptionService"/>
      </properties>
      <exports>
        <port name="RouteDescriptionEndpointPort" type="asg-srds:RouteDescriptionEndpoint"
         channel-ref="outboundSimpleRouteDescriptionChannel"/>
      </exports>
    </service>

    <service name="ExtendedStadtplandienstRouteDescriptionService" provider="uri:protocoladapter.soap">
      <properties>
        <property name="concreteWsdlUrl" value="..."/>
        <property name="serviceWsdlNamespace"
         value="http://www.asg-platform.org/scenarios/attractionbooking/services/
               ExtendedStadtplandienstRouteDescriptionService"/>
        <property name="serviceWsdlName" value="ExtendedStadtplandienstRouteDescriptionService"/>
      </properties>
      <exports>
        <port name="RouteDescriptionEndpointPort" type="asg-erds:RouteDescriptionEndpoint"
         channel-ref="outboundExtendedRouteDescriptionChannel"/>
      </exports>
    </service>

    <service name="BpelProcess" provider="uri:bpelProvider">
      <properties>
        <property name="compiledProcess" value="Scenario2.cbp"/>
      </properties>
      <imports>
         <port name="PhoneLocationPL.PhoneLocationProvider" type="asg-pls:PhoneLocationEndpoint"
              channel-ref="outboundPhoneLocationChannel"/>
         <port name="RouteCalculationPL.RouteProvider" type="asg-rcs:RouteCalculationEndpoint"
              channel-ref="outboundRouteCalculationChannel"/>
         <port name="SimpleRouteDecriptionPL.SimpleRouteDescriptionProvider" type="asg-srds:RouteDescriptionEndpoint"
              channel-ref="outboundSimpleRouteDescriptionChannel"/>
         <port name="ExtendedRouteDecriptionPL.ExtendedRouteDescriptionProvider
               type="asg-erds:RouteDescriptionEndpoint"
              channel-ref="outboundExtendedRouteDescriptionChannel"/>
      </imports>
      <exports>
         <port name="Scenario2PL.Scenario2Provider" type="s2:Scenario2PT" channel-ref="inboundChannel"/>
      </exports>
    </service>

  </services>

 </system-descriptor>
```