

Project IST 026850 SUPER

Semantics Utilized for Process management within and between Enterprises

Deliverable 6.2

Process Execution Engine First Prototype

Leading Partner: ONTO

Contributing Partner: USTUTT, CEFRIEL, NUIG, Hanival

Security Classification: Public (PU)

September, 2007

Version 2.0

Project	SUPER	SUPER-Project-No	026850
	Process Execution Engine First Prototype		
Document	Deliverable 6.2	Date	01.10.2007

Project Details

IST Project Number	026850
Acronym	SUPER
Project Title	Semantics Utilised for Process management within and between EnteRprises
Project URL	http://www.ip-super.org
EU Project Officer	Werner Janusch

Authors (Partner)	Mihail Konstantinov (ONTO), Tammo van Lessen (USTUTT), Federico Michele Facca (CEFRIEL), Walid Gaaloul (NUIG), Gabriela Vulcu (NUIG) Bernhard Schreder (Hanival)		
Deliverable Owner (Partner)	Mihail Konstantinov (ONTO)	E-mail	mihail.konstantinov@ontotext.com
		Phone	+359 2 976-8310

Project	SUPER	SUPER-Project-No	026850
	Process Execution Engine First Prototype		
Document	Deliverable 6.2	Date	01.10.2007

Versioning and Contribution History

Version	Description	Comments
0.1	Initial draft	
0.2	Barry review issues fixed	
0.3	M18 Input integrated	
1.0	Final version	

Project	SUPER	SUPER-Project-No	026850
	Process Execution Engine First Prototype		
Document	Deliverable 6.2	Date	01.10.2007

Table of Contents

Project Details	II
Versioning and Contribution History	III
1 Executive Summary.....	1
1.1 Changes since M12i version.....	1
2 Availability and Contacts	2
2.1 Download and Compilation	2
3 Alignment with SUPER.....	4
3.1 Architecture	4
3.2 Methodology.....	7
3.3 Modelling Stack	8
3.3.1 The Semantic BPEL ontology (sBPEL)	8
3.3.2 The Events Ontology (EVO).....	9
4 Requirements	10
4.1 Functional requirements	10
4.2 Technical requirements summary	12
5 Purpose and Functionality	13
5.1 The Current Version	13
5.1.1 Pluggable ExtensionActivity implementations.....	13
5.1.2 Preserving Links to sBPEL.....	13
5.1.3 BPEL4SWS Implementation	13
5.1.4 Event Notification.....	13
5.1.5 Semantic Activities.....	14
5.1.6 Semantic assignment/mediation.....	15
6 Test Scenarios.....	18
6.1 Scope of the test scenarios.....	18
6.2 Description of the scenarios.....	19
6.3 Generating test cases from the scenarios	21
6.4 Mapping the Scenarios to the Use Cases.....	21
7 Licensing	22
7.1 License Agreement for SUPER-related Engine Extensions	22
7.2 Apache ODE License Agreement	22
7.3 Licensing of Third Party Libraries.....	22
8 Installation and Usage	24
9 Future Plans	25

Project	SUPER	SUPER-Project-No	026850
	Process Execution Engine First Prototype		
Document	Deliverable 6.2	Date	01.10.2007

References 26

Appendix A Licensing Schemas 27

A.1 Apache Software License (ASL) Version 2.0 27

A.2 ANTLR License 30

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

1 Executive Summary

One of the main objectives of this deliverable is to describe how process models are instantiated and executed by SUPER Execution Engine. We present information on how the user can build and use the engine to execute BPEL4SWS process model, as well as the functionality it provides and example test scenarios.

First, we give instructions on how the engine can be retrieved from SUPER source repository and how it can be built. Then we continue with the requirements the engine must satisfy. They are divided in two parts – functional and technical. The former are defined on conceptual level and are imposed by the role of the engine in the entire SUPER process and its relationships to other components. The latter are purely technology induced and are described as relations to other platforms, libraries, interfaces and tools. After presenting the requirements, we introduce the functionality provided by the engine and map it to the requirements in the previous section. To guide the user on how to utilize the provided functionality, we offer several test scenarios. Then we describe the use cases these test scenarios support. Finally the licensing scheme chosen is discussed, a short section on configuration of the engine is presented as well as an outlook of future development.

1.1 Changes since M12i version

Due to ongoing research since the submission of D6.2 M12i, changes on external software components the SBPELEE depends on and as well as due to reviewer comments, the following changes have been introduced:

- Section 1.1, a new build process using Rake is introduced. These replaces Maven tool for building
- Section 4, new functionality is implemented – viz. semantic invocations and semantic assignment / mediations. These are described accordingly.
- Section 5, test scenarios and use cases were defined and added as a new section.
- Some minor changes are made, by moving and updating text as features were implemented.

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

2 Availability and Contacts

Currently the Semantic BPEL Execution Engine (SBPELEE) engine is not available as compiled distribution because of its heavy dependency on Semantic Service Bus (SSB) [SUPER-D76]. Once SSB component is complete and SBPELEE is integrated and tested, an external distribution will be available. Currently the engine could be built from source code and deployed to a web container such as Apache Tomcat [APACHE-TOM]. The source code of the SBPELEE engine can be obtained from a Subversion server run by one of the project partner (LFUI).

Table 1 reports the basic information to retrieve the current version of the source code of the SBPELEE engine.

Table 1. Summary information for the availability of the SBPELEE engine.

Version:	0.1
Download:	Not currently available.
Accompanying architecture and design document:	Deliverable 6.1. Deliverable Description/Name.
Source control:	https://cvs.deri.at/super/WP6/trunk/sbpelee/ (For an account on the svn server contact Graham Hench graham.hench (at) deri.org (LFUI))
Contact person:	Tammo van Lessen < tammo.van.lessen (at) iaas.uni-stuttgart.de > (USTUTT)

2.1 Download and Compilation

Current version of SBPELEE derives from release first official incubator release of ODE, i.e. version 1.0. Requirements for building it and retrieving relative libraries are JDK 5 and Ruby 1.8.6. Once the current version has been retrieved of the source code from the SVN repository, execute the following commands in the folder containing the local repository folder to retrieve libraries from remote repositories and obtain a compiled and working version of the engine (the whole process is quite slow the first time):

```
rake -f sbpelee.rake package TEST=off
```

To prepare the source code to be used within the eclipse environment and link correctly the downloaded libraries:

```
rake -f sbpelee.rake eclipse
```

Rake will create the .project and .classpath files. Libraries pointer generated by Maven use an Eclipse or environment variable called "M2_REPO": you need to add classpath variable with this name to

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

Eclipse, pointing to your local maven repository (The default locations is \$home/.m2/repository, where home is your user home folder) so that Eclipse can find correctly the needed libraries.

Note: One needs to have ruby, rake and rubygems installed and setup as per BUILDING file in SBPELEE root directory. Additionally a gem called buildr (version described again in BUILDING) need to be installed, which is made by:

```
gem install buildr -v$VERSION
```

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

3 Alignment with SUPER

3.1 Architecture

The need for an alignment of this deliverable with the SUPER architecture is two-fold: both the Integration and Interoperability Specification in deliverable D7.1 [SUPER-D71] and the architecture itself, as detailed in deliverable D7.2 [SUPER-D72], depend on and influence the development of the SBPELEE. First, the alignment with the sections in D7.1, which contain user scenarios with direct relevance to the development of the SBPELEE, is surveyed below, followed by a section on the technical alignment with the overall architecture, as specified by D7.2.

The user scenarios presented in D7.1 are generic, derived from the BPM lifecycle and not tied to a specific technical implementation, setting them apart from the section on the Behavioural View in deliverable 6.1. Still, they provide a direct link to this deliverable by setting forth some specific requirements for the SBPELEE. From the phases corresponding to the SUPER BPM lifecycle, Configuration, Execution and Monitoring interact with the execution engine, as noted in the corresponding user scenarios. In the following paragraphs the use cases which are relevant for the SBPELEE are described, as they determine the needed functionality as defined by the architecture.

- The Semantic Business Process Configuration phase includes SBP deployment as one of its central use cases, conducted by an IT expert. The expert first has to perform necessary process refinements and has to create mediators and attach them to the process, should such mediation be required after the process composition. The main requirements from this phase are the deployment of the process model itself, and the possibility of exposing the deployed process as a web service itself, both of which is currently possible with this version of the SBPELEE.
- In the Semantic Business Process Execution phase the SBPELEE plays a core role, as it is responsible for the main use cases identified for this phase. Of these, SBP Instance Execution is of course the main functionality supported by the SBPELEE, including both Instance Identification (i.e. identifying to which running instance a message should be sent to) and Control Flow Navigation (i.e. navigating through the SBP control flow). These functionalities are currently supported by the prototype. Additionally, the SWS Invocation is involving the SBPELEE for the case of semantic web services implemented as SBPs.
- The Semantic Business Process Monitoring phase consists of four major use cases, all of which depend on functionality provided by the execution engine. Configuration allows the business analyst to enable or disable the monitoring of running process instances, or setting the detail level of the monitoring. The Management allows for basic control over running instances, such as stopping, aborting or resuming a process. Furthermore, two kinds of monitoring functionalities should be supported by the engine: Passive Monitoring, which displays currently relevant information to a business analyst, depending on his or her

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

requirements, and Active Monitoring, which allows for the querying of specific information by the analyst.

Further details on the SBPELEE's alignment with the SUPER methodology and the lifecycle phases are provided in Section 3.2.

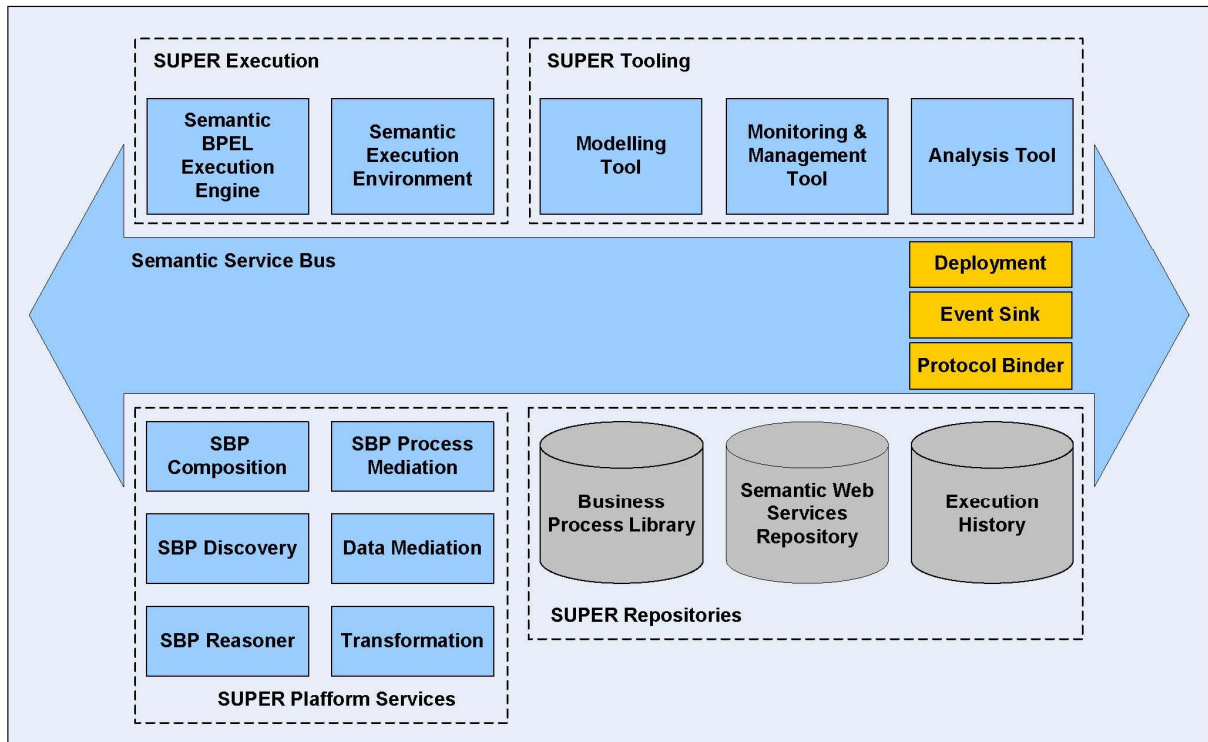


Figure 1 SUPER architecture

In the following, we examine the technical requirements resulting from the alignment of the SBPELEE with the overall SUPER architecture, and the positioning of the execution engine inside the architecture. Figure 1 shows the SUPER architecture. The SBPEL Execution Engine is located in the SUPER Execution environment dedicated for the discovery, interoperability and execution of SWS. It interacts with other components via the Semantic Service Bus (SSB) which is the central component of the SUPER architecture providing components communication infrastructure. Basically SBPELEE is related to the following components of the SUPER architecture:

- *Semantic Service Bus*: SBPELEE implements an *SSB Invocation and Management Framework* to send and receive (semantic) web service calls on the SSB. The messages that are exchanged between the SSB and the process engine are protocol independent (normalized). Thus the process engine doesn't have to cope with protocol specifics when invoking (semantic) web services or receiving web service calls. In addition, SBPELEE provides a set of functionalities for:

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

- Deployment: SBPELEE uses BPEL4SWS file and SA-WSDL description describing the service defined by the process model and WSDL files which contain service descriptions to import and deploy the process model.
 - Execution: SBPELEE provides an internal endpoint where it can consume incoming messages. These messages are passed to the corresponding process instance and are processed there according to its process description. If the message does not belong to a particular process instance, a new one will be created. In addition, the process instance might invoke other services provided as internal endpoints.
 - Monitoring/Event notification: SBPELEE publishes events related to the status changes of process instances, activities or variables during process instance execution in a sink within the SSB.
- *Lifting/lowering service*: the engine needs a lifting and lowering service that is able to perform such a transformation. The lifting and lowering services are identified in the SA-WSDL.
 - *Semantic Execution Environment*: SBPELEE needs to delegate native support for the invocation of semantic web services to the SEE via the SEE API.
 - *Execution History*: During process execution, SBPELEE writes the audit log into the Execution History. Technically, SBPELEE publishes events of the EVO ontology to a topic which is provided by the SSB. The Execution History subscribes to this topic, and writes the events to a persistent store.
 - *SBP Monitoring Tool*: The SBP Monitoring Tool can, same as the Execution History, subscribe to events of the EVO, which the process engine publishes. Alternatively, it can use the management framework interface of the SBPELEE to retrieve information on the running process instances.
 - *SBP Reasoner*: The engine may use the SBP when performing data handling activities, (i.e. to perform mediation during copying data between variables) and evaluation of transition conditions when process data is ontologically annotated.

We describe SBPELEE interactions with the other SUPER components during execution through the following scenario. This execution scenario is initiated by the service requestor which can be either an arbitrary user or BPEL4SWS instance running on SBPELEE. Service requestor does not have direct access to the BPEL4SWS since it is interfaced and described as SWS, therefore execution of SWS which interfaces business process entails BPEL4SWS instantiation and execution on SBPELEE. Bidirectional XML to WSML Transformation Service is utilized for executing mappings between WSML and XML during the interaction between SEE and SBPELEE. SBPELEE distinguishes between invocations of SWS and vanilla web services. For vanilla web service invocation SBPELEE uses SSB Protocol Binder while for the communication with SEE it sends WSML messages serialized in XML to the SEE. Internal and external web service calls are carried out via SSB Protocol Binder where resolution of the concrete endpoint and service invocation takes place. Ongoing state of the

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

BP4SWS execution by SBPELEE is stored into Instance Data repository. Whilst all EVO events are transparently stored into Execution History which subscribes to all events which are passed via SSB.

3.2 Methodology

In the SUPER methodology, the notion of most importance to the SBPELEE is the separation of process descriptions into the merely analytical versus the executable. This idea stems from the requirements of the two types of users of SUPER – namely business analysts and technical staff. While the former only need to define important aspects of a business process and its properties without specifying strict workflow, the produced business process inevitably must end as a formal specification of ordered decidable actions fulfilling a business goal. This formal specification is machine-interpretable and it is the model from which SBPELEE creates instances which are executed with specific input artefacts and satisfy concrete business needs. There are several types of artefacts defined by their lifespan and possibilities for reusability. These belonging to the type of most general and reusable artefacts are the general workflow, process, business process, organizational etc. ontologies. Those with most importance to SBPELEE are described in section 3.3 of this document. The second type is comprised of artefacts reusable in given organization or business unit. These are various semantic web services, domain ontologies and reusable processes or process fragments. They are created during design phase of Semantic Business Process (SBP) Lifecycle or are already available and stored in SUPER repositories. The third type consists of instances of process mining related ontologies and they are sensible only in the context of business process instance execution. These are the input parameters for the business process and various instances of log data (e.g. instances of PMO, EVO and CONTO ontologies). All of them have their role in different phases of a business process lifecycle.

The SBP Lifecycle outlines how business processes are created, managed and executed both within a single organization and across several interconnected organizations. It uses state-of-the-art methodologies from business process management domain and from semantic world domain as well as from analysis of various use-cases – general and from telecommunication domain. The main phases through which a business process in SUPER lives are the design, execution and analysis phase. In the design phase a new semantic process is created form scratch, imported from existing (non-semantic) process or composed of already created semantic business processes or fragments. The last step of design phase is the creation of deployable business process which is further deployed and executed by SBPELEE. While the processes based on any of business process modelling ontologies (BPMO, sBPEL, sBPMN, sEPC) may not have defined workflow and parameters, the deployable process artefact must have all of latest strictly defined to allow instantiation and execution in SBPELEE. To achieve this, the semantic business process is serialized in BP4SWS and deployed to the execution engine. All artefacts required for instantiation and execution of the process are stored in repositories where the SBPELEE access them. The process then goes to the second phase – instantiation and execution. This phase is completely covered by SBPELEE. During instantiation input parameters are provided as ontology instances in corresponding repositories.

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

SBPELEE have these repositories specified during its configuration via tools using configuration API of the engine. Another things specified during configuration are SWS repositories and execution history repositories. During its execution a process instance may be monitored – either manually or by automated process utilizing rules. This will allow process to be stopped if it has erroneous behaviour or by other unforeseen events. Every important step of a process execution is logged using instances of PMO, EVO and CONTO ontologies in a history repository for processing in the third phase of SBP lifecycle – the analysis. SBPELEE is responsible for creation of these instances which will be further processed by Semantic Business Process Modelling Environment.

3.3 Modelling Stack

As showed in Figure 2 the SUPER Modelling Stack is structured in five layers, which span from the business domain analysis to the concrete implementation. In particular SBPELEE can be positioned in the third layer from up. The main ontologies involved are sBPEL (as input in its XML serialization representation) to provide the description of the process to be executed and EVO (as output) to provide an ontology of the events occurred during the execution of a semantic business process.

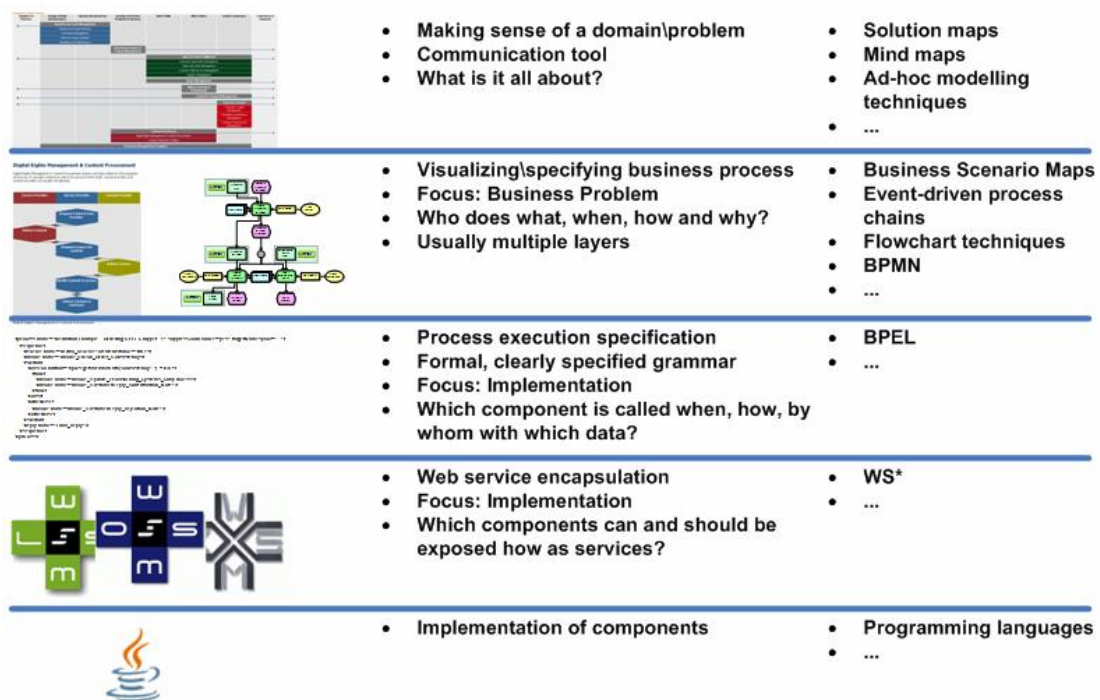


Figure 2: SUPER Modelling Stack

3.3.1 The Semantic BPEL ontology (sBPEL)

This is the ontology version of BPEL [BPEL] with additional constructs. Semantic BPEL is ontology of executable business processes, structured around the control concepts of the BPEL language. In common with BPEL, sBPEL presents a business process as a composition of a number of partners, which can be met at execution by services. Whereas BPEL describes each partner via its ability to meet a set of operations syntactically described in a WSDL port type, however, sBPEL describes them

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

as being able to meet a set of semantically-defined goals. Furthermore, sBPEL will describe communication between partners in terms of semantic mediation, rather than syntactic manipulation. Finally, in contrast with BPEL, future versions of sBPEL will allow the first-class reuse of process fragments, rather than requiring only the use of complete processes abstracted across the service boundary, as in BPEL.

SBPELEE relies on the serialized form of the sBPEL ontology (BPEL4SWS) as executable process description: i.e. SBPELEE will deploy and execute processes described according to sBPEL.

3.3.2 The Events Ontology (EVO)

This ontology represents events taking place during the execution of semantic business processes. It will be used mainly for monitoring and management purposes where Events are the key concept for supporting the analysis processes over the actual executions. The concepts of EVO are mainly based upon events identified in MXML (e.g. *Start Task*, *Abort Task*, etc.). Events refer to actual Processes and/or Activities definitions in order to indicate *what* was started, aborted or finished. This provides a generic and extensible framework where events refer to external processes or activities definitions, supporting the use of generic monitoring and mining techniques over specific processes. Obviously, additional domain specific analysis method or techniques can be defined if necessary. Furthermore, by means of this generality the very same methods can be applied to analyze the enactment of Processes at different levels of abstraction. For instance, business practitioners are given the means for focusing on the enactment of business processes whereas IT personnel can focus on the technical aspects of the service delivery.

SBPELEE will adopt EVO to provide an ontological representation of the different events occurring during the execution of a process so as to provide data to support process monitoring and process analysis.

For more details on the ontologies involved in the Semantic Process Execution we refer the interested reader to [BPOF].

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

4 Requirements

In order to clarify the scope of SBPELEE, we analyze the requirements imposed on the engine. We look at functional capabilities specific to the engine in Section 4.1. Since we are extending Apache ODE [APACHE-ODE] we have several dependencies on libraries, APIs, toolkits and frameworks. These are presented in section 4.2.

4.1 Functional requirements

In this section we describe functionality that pertains only to SPBELEE. These are the features added to Apache ODE and are the main target of the deliverable.

- Deployment of a Process Model

Once the process model is created in Semantic Business Process Modelling Environment (SBPME) it should be deployed to SBPELEE via the SBPELEE Deployment API. The deployment process translates SBP model into engine internal representation, which is stored for further processing. The process model is also exposed as a set of Web services and/or semantic Web services.

- Repository integration

During the lifecycle of a process several artefacts are used and produced. For example, the process model refers to the SUPER ontologies and domain ontologies for common conceptualization. Inevitably SWS are used in the process specification, too. Finally, ontology instances are used in the roles of input to SBP instance and audit events history. Because of these storage requirements an effective and efficient integration with various repositories is a must. This integration is enabled by the SSB.

- Instantiation of a process model

In order for a process to be executed, it must be created from a process model and supplied with its initial state parameters. These parameters are provided as references to artefacts in repositories, as well as in terms of process user input. The need may arise to create new artefacts just for this instance of a process model. This requires access to ontologies which are also stored in repositories and storing these new artefacts.

- Execution of a process instance

After instantiation a process depends on several functions to fulfil its goal.

- (Semantic) web service invocation

A process performs its actions by invoking web services – semantic or vanilla web services. Since the mechanism for execution of the two is different, SBPELEE should be able to make a distinction. Non-semantic web services may be executed by a module in the ODE engine itself, but in a way such that this module could be replaced

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

with another one with greater complexity and functionality. Since there are no requirements in the scope of SUPER for extending or substituting this module, the focus in this deliverable has been on extending the engine to enable invocation of semantic Web services. Semantic web services are invoked by delegating this task to the Semantic Execution Environment (SEE). The interaction with the SEE are made via the Semantic Service Bus.

- o Receiving a message

Receiving a message is the means by which other processes can interact with running process instance. The engine should be able to deliver the messages to the correct instance in the case that a SBP model has many instances. Here the concept of Conversation as introduced in [SUPER-D1.3] must be supported. Since messages are received via SSB, integration requirements specified in “web service invocation” section hold true here.

- o Semantic assignment/Data mediation

A process instance may interact with other process instances. Assignment of variables of incompatible types may be needed. For this reason, SBPELEE should be able to make use of mediation functionality. This functionality is provided as an output from SUPER WP 4 and is used via the SSB. Here again as in SWS invocation and message receiving a clear interface must be defined.

- o Storing and retrieving audit data

During the execution of a process the process engine produces events that are stored in terms of audit data. These data are comprised of instances of the audit ontologies: event ontology (EVO), component ontology (CONTO) and process monitoring and mining ontology (PMO). EVO formalizes different types of events. CONTO is used for identification of different software components. PMO is ontologization of MXML – XML-based format for storing process event logs used by ProM [PROM] for process mining. Log entities are handled in two ways. First, they are stored in the execution history repository where from they are accessed for audit purposes and by other infrastructure components. One of the main usages of audit data is for process mining, analysis and optimization of process execution. The tools performing all these tasks are implemented in other work packages and SBPELEE only provides API for accessing audit data in the case of passive monitoring only.

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

- Control and monitoring

An API is needed for basic control over SBPELEE and the artefacts it hosts. The basic functionality it should provide is:

- Suspending and resuming a process
- Stopping a process
- Manual monitoring of a running process

Additionally, a running process may be monitored for compliance to some rules, e.g. some legal regulations in the domain.

4.2 Technical requirements summary

Nature: Middleware

Interfaces (API, Web Services): a Java API.

Platform: JDK 1.5

Supported standards: BPEL, BPEL4SWS, WSMO, SOAP, WSDL

Required Libraries (SUPER, WSMO-related): WSMO4J

Required Libraries (others): see Test Scenarios

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

5 Purpose and Functionality

5.1 The Current Version

The Semantic Business Process Execution Language Execution Engine (SBPELEE) is in charge of the execution of BPEL4SWS [SUPER-D1.3] process models [LNDKKCL07]. These process models are produced prior to the execution phase of the SUPER Methodology [SUPER-D22] and are the executable representation of sBPEL process models. BPEL4SWS is an extension of WS-BPEL 2.0 [WS-BPEL20]. Since BPEL4SWS uses the extension mechanisms of BPEL 2.0, a BPEL4SWS process model still satisfies the BPEL 2.0 schema definition.

The implementation of the SBPELEE is based on the code base of Apache ODE [APACHE-ODE], a BPEL engine developed at Apache Software Foundation which aims at BPEL 2.0 compatibility.

Since Apache Ode does not support BPEL 2.0's extensibility techniques yet, the development of the SBPELEE has also required modifications in Ode's core modules [LNDKKCL07].

The major capabilities of the current version that have been enabled by the work on this deliverable are described in the following sections.

5.1.1 Pluggable ExtensionActivity implementations

The Apache Ode engine has been extended by a plug-in mechanism that enables the development and usage of <extensionActivity> elements without modifying the engine's core for each newly created extension activity. These changes have been embedded in the Ode core and provide the basis for a well designed and separated implementation of the BPEL4SWS extensions.

5.1.2 Preserving Links to sBPEL

A BPEL4SWS definition has links to the sBPEL process model it has been derived from. For analysis purposes it is necessary to preserve these links during the execution and add them to the related events. The current version of the SBPELEE already contains this "link passing" functionality.

5.1.3 BPEL4SWS Implementation

By using the aforementioned extension mechanism, the current version provides a proof of concept implementation for the BPEL4SWS semantic invoke activity [SUPER-D13, Section 4]. Since the Semantic Service Bus (the upcoming deliverable D7.6) is not yet implemented, it is not yet possible to delegate the invocation of a semantic web service to the SEE in the way described in the SUPER Architecture [SUPER-D72]. However, there is a proof-of-concept implementation that can delegate service invocation to either WSMX and IRS-III (SEE implementations).

5.1.4 Event Notification

During the execution of a BPEL4SWS process instance, the SBPELEE generates events that are of interest for the Monitoring Tool or for the Tools used in the SUPER Analysis phase. These Events are represented as instances of concepts defined by the Event Ontology (EVO). Events related to activity executions contain also pointers to the related sBPEL process model and events related to data

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

manipulation activities contain references to their related typing concepts. Current implementation of the event notification mechanism dispatches also ontological instances describing the process and activity instances and their related input/output data, according to UPO ontology. The current version of the SBPELEE provides an event notification mechanism that serializes execution events to EVO instances and publishes them to a JMS topic. Since in future WS-Notification will be used as a pub/sub framework a WS-Notification Topic will be utilized in later versions of the engine. Other SUPER components, such as the Monitoring Tool and the Execution History, can subscribe to this topic and can consume these events.

The generation of EVO instances and their serialization is based on a newly created API that allows one to programmatically create EVO instances. The API is included in the SBPELEE source code and can be reused by other components that need to create and serialized process events in the SUPER architecture (e.g. the Semantic Service Bus or the Semantic Execution Engine).

The EVO API is used internally by the `AMQBpelEventListener` class to generate and dispatch events. This class is a customized ODE BPEL listener that transforms internal SBPELEE events to EVO instances. Due to limitations of the ODE `BpelEventListener` interface that is implemented by the `AMQBpelEventListener` class, the component is not yet fully configurable. This issue would be resolved when the `BpelEventListener` interface is updated by the ODE developers according to the suggestions provided by the SBPELEE team.

Currently the event notification component has been tested for Start and Correct Completion of Processes and Activities; monitoring of other events such as manually triggered ones have not been tested yet. They will be tested when then Monitoring facility is ready for final integration (see D6.8). An early integration test has been performed for the showcases presented at the SUPER Review. The basic functionality of the event dispatching mechanism can be tested running locally the SBPELEE JUnit tests and an instance of a JMS message broker (e.g. ACTIVE MQ).

5.1.5 Semantic Activities

Semantic activities are the semantic equivalents of WS-BPEL interaction activities. These activities are represented in BPEL process definition by `<receive>`, `<reply>`, `<invoke>` and `<pick>` elements. According to [SUPER-D1.3] these have corresponding elements in BPEL4SWS. Currently only synchronous invocation of semantic web services is implemented. This functionality is implemented as pluggable `extentionActivity` as per Section 5.1.1. Since these extensions are defined in WS-BPEL specification, SBPELEE is backward compatible with standard BPEL and vanilla web service invocations. The latter is realized by the standard functionality inherited from Apache ODE, while the former is delegated to SEE via SSB.

Invoking semantic web services requires passing and receiving semantic descriptions. In our case, at the lowest level these are WSMO instances. However, for compatibility with WS-BPEL, all variables store their values as XML elements, too. To handle that, semantic invocation makes use of SA-WSDL mappings and lifting and lowering services to be provided by SSB. Before invocation all values are

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

lifted to concept instances, encapsulated in ontology and then sent to SSB/SEE together with a web service or goal definition (as per WSMO). The result from the invocation is an ontology containing WSMO instances, which then are lowered and set as XML variable values. All interactions take place via SUPER APIs as defined in [SUPER-D73].

5.1.6 Semantic assignment/mediation

During process execution, process instances may interact with other process instances whose data models are different. Thus the assignment of variables of incompatible types might happen. In order to overcome this obstacle of communication between incompatible entities in a flexible manner, SBPELEE needs mediation functionality. We introduce the use of mediators by extending the <assign> activity since BPEL2.0 allows for such extension (described in [SUPER-D1.3]).

In SUPER the mediation task is provided by the Data Mediation component (as described in [SUPER-D41]). Interactions with Data Mediation component are made via SSB (according to the architecture described in [SUPER-D72]). Since the SSB is not yet implemented it is not yet possible to delegate the mediation task to this component. However we provide a mock-up local implementation until then.

We provide the implementation code for the semantic assignment/mediation functionality in the SBPELEE. It remains to have a functional code when the SSB will be implemented and the mediation framework prototype will be available.

The Apache ODE initial base code did not provide any implementation for the Extension Assign Operation. However it allows for the extension of this operation with an element that must belong to another namespace than the WS-BPEL namespace. In accordance with [SUPER-D1.3] we provide extension for BPEL data handling, by introducing the <s:mediate> element for the Semantic Assign Operation as described below (see: Listing 1):

```
<assign validate="yes|no"? standard-attributes>
  Standard-elements
  (
    <copy keepSrcElementName="yes|no"?
      ignoreMissingFromData="yes|no"?>
      from-spec
      to-spec
    </copy>
    |
    <extensionAssignOperation>
      <s:mediate name = "NCName" mediatorURI = "anyURI"
        inputVariable = "NCName" outputVariable = "NCName">
    </extensionAssignOperation>
  )+
</assign>
```

Listing 1: Extension element for semantic assignment

The <s:mediate> element must belong to another namespace than the WS-BPEL namespace and it has the following attributes:

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

name: the name of the specific element
mediatorURI: the URI of the mediator that mediates between the two variables
inputVariable: the input variable is mediated and the result will be assigned to the output variable
outputVariable: the output variable

All of these attributes are required, as specified in [SUPER-D1.3].

In the implementation we extended/changed the existing code in order to support the new added element (i.e. <s:mediate>) and to provide its described functionality (i.e. the mediation between the input and output variables given the mediator's URI). During its execution a process goes through three phases: **deployment**, **compilation** and **runtime**. The implementation for the semantic assignment, basically, influences the implementation of these three phases:

The implementation provides support for the deployment of a BPEL process containing the <s:mediate> element: the BPEL object model (BOM) representation of the new element. We added a new class (i.e. ExtensionAssignOperationNestedElement) to represent the nested element of the extension assign operation in the BOM. We also added code to the classes AssignActivity and ExtensionAssignOperation in order to provide compatibility for the ExtensionAssignOperationNestedElement.

We added new internal class to hold the compiled version (i.e. an 'OClass') of the extension assign operation. For this purpose we implemented a new inner class in the OAssign class (i.e. OAssign.ExtensionAssignOperation). We modified BPELCompiler, AssignGenerator and AssignGeneratorMessages classes also to provide compilation code for the ExtensionAssignOperation. Until now the <assign> activity was considered to have only a set of <copy> elements. Now, since we added the <extensionAssignOperation> with the mediation functionality, we needed to reconsider the <assign> activity. <assign> activity consists now of a list of operations (i.e. either copy operations as before, but also semantic assignment operations) which is supported by all the above mentioned classes.

The implementation for the runtime part resides in the Assign class that have been changed to provide the semantic assignment functionality. The semantic assignment basically consists of three main operations: lifting, mediation and lowering which will be described next.

Lifting is the operation that transforms the XML variables into their ontological representation according to a schema mapping (i.e. the "liftingSchemaMapping" attribute used in SA-WSDL to annotate WSDL elements). We use lifting to prepare the variables for the mediation process that deals with semantic entities.

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

After this operation we do the mediation for the two variables, having their ontologies (source ontology and target ontology) and the instance for the input variable. The source/target ontologies are the ontologies where the semantic representations of the input/output variables belong to respectively. The Data mediation SUPER component will be invoked giving the mediator's URI, the source/target ontology and the instance which represent the lifted input variable.

The result of the mediation is then lowered and the XML representation is assigned to the output variable. As well as the lifting transformation, the lowering transformation needs a schema mapping (i.e. from semantic to XML representation). The lowering schema mapping is provided as a SA_WSDL annotation of the output variable type.

As specified earlier in this section we provide implementation code from SBPELEE side for the semantic assignment functionality. We do not have a functional code yet, since there is no implementation of the SSB, Data Mediation Component and lifting/lowering transformations. In this purpose we propose the lifting/lowering interface that needs to be implemented in the future (see: Listing 2.):

```
public interface LiftingLoweringTransformation {
    Ontology lift(Element source, URI schemaMapping);
    Element lower(Ontology instances, URI schemaMapping);
}
```

Listing 2. The proposed lifting/lowering interface

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

6 Test Scenarios

For this first version of the SBPELEE prototype, which has been described in the remainder of this deliverable, we have devised a set of test scenarios. These scenarios will be used to set up a comprehensive (unit) test framework, applicable to specifically test the new functionalities introduced by the components designed on top of the existing Apache ODE process engine.

This section was at first planned to include comprehensive scenarios based on the use cases, but for the M18 version of the prototypes for both the use cases and the SBPELEE this was not deemed appropriate. Some components which will support the whole SUPER methodology, and thus the business process lifecycle phases will only be created after the M18 milestone (examples include the translation components between BMPO and sBPEL, and between sBPEL and BPEL4SWS, respectively, as well as integrated versions of the SSB and SEE).

Rather these tests should concentrate on those parts and developed components which can be evaluated currently. This includes testing the parsing functionality of BPEL4SWS files, checking the logging for compliance with the EVO ontology and other validation checks. For these purposes we have collected a number of (manually created) artefacts for testing purposes, i.e. processes in BPEL4SWS and XML fragments for process variable contents. For example, as no automated translation has been available yet, we have manually translated available sBPEL processes, which were created for the Use Cases, to BPEL4SWS.

This section of the deliverable is thus structured as follows: The following section explains the scope and structure of the scenario descriptions and the creation of the artefacts used for the tests. The actual test scenarios for the M18 prototype of the SBPELEE are described next, which is followed by an overview how specific test cases are created.

6.1 Scope of the test scenarios

This first set of test scenarios is going to specifically address the current prototype of the SBPELEE, with the objective of evaluating the additional components which were added to the base Apache ODE functionalities. Among the components, for which test scenarios have been devised, are the BPEL4SWS parser, the logging functionality, the semantic assignment and mediation, as well as the interaction activities.

The scenario descriptions below define the three different, essential steps usually found in all tests:

- Preconditions which have to be met before the test can commence are established;
- The item to be exercised under the test must be declared;
- The post-conditions which need to be verified are described;

As these test scenarios are general in nature, in that each should be decomposed in a set of test cases, they are not going to explicate every testable postcondition of applying the particular test cases,

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

but will rather make general statements about the kind of verification that has to take place after exercising the test item.

Some of the following test scenarios make use of predefined BPEL4SWS processes, which specifically address some of the new features of the current SBPELEE prototype. To create valid BPEL4SWS processes, a BPEL4SWS XML Schema was created to validate the process documents against. The schema imports the official WS-BPEL 2.0 schemas from OASIS¹, and builds upon these to integrate the new language elements. A BPEL4SWS process model already satisfies the WS-BPEL 2.0 schema – due to the use of the BPEL 2.0 extension mechanisms for BPEL4SWS – we can therefore directly constrain created test processes to use the targeted language extensions. Other scenarios, such as the logging mechanism tests, don't directly depend on specific processes, For these existing processes have been reused (but these test scenarios in some cases need other artefacts to check an output's validity, such as event instances from the EVO Ontology, or the content of process variables).

6.2 Description of the scenarios

Test Scenario 1)

Test Name: **BPEL4SWS Parser and Deployment tests**

Test Description: This test scenario should encompass a set of tests for the parsing capabilities of the SBPELEE. Specifically every BPEL4SWS language element introduced until M18 should be parseable and result in the instantiation of corresponding in-memory representations of the business process' elements (as part of the BPEL Object Model).

Preconditions: A process is deployed to the SBPELEE,

Items under test: The SBPELEE parser

Postconditions: The parser should be able to read the supplied BPEL4SWS processes and parse them to a corresponding BPEL4SWS in-memory representation.

Artefacts used for testing: A number of different BPEL4SWS processes have been supplied for use with the test cases. Each of these artefacts features some of the currently defined BPEL4SWS language elements. The artefacts will also include "negative" examples, which should result in correct error messages to be generated during the parsing.

Test Scenario 2)

Test Name: **Event notification capability tests**

Test Description: As described in Section 3.1.4 of this deliverable, the logging for the SBPELEE produces events in compliance with the EVO Ontology, which has been developed as part of the ontology stack in WP1 [BPOF]. For use with other components, such as the analysis tools, these events will have to be stored in a dedicated Execution History repository, which is developed in WP6.

¹ Schemas for abstract and executable WS-BPEL are available at <http://docs.oasis-open.org/wsbpel/2.0/OS/process/>

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

Currently the logging generates messages sent to a JMS topic, therefore the test cases for this scenario need a JMS message broker instance. The message content (the events) can then be tested for compliance with the EVO Ontology. Though the logging component features a dedicated EVO Ontology API, the tests need to check if the event logging is conformant with the current version of the EVO Ontology, and will thus have to be adapted for future versions of the ontology stack.

Preconditions: SBPELEE, deployed processes, JMS message broker instance

Items under test: The event notification component

Postconditions: The events produced by the notification component include all correct instances and attributes in the current version of the EVO Ontology, and comply with the occurrence of particular events in the BPEL4SWS process instances.

Artefacts used for testing: BPEL4SWS processes which would generate the events to be tested (i.e. start and completion of processes and activities)

Test Scenario 3)

Test Name: **Semantic Assign and Mediation Tests**

Test Description: As described in Section 3.1.6, the Data Mediation, realized through an extension of the assign activity, depends on other components which are not yet available. Since the outcome of the actual mediation process will be tested in the respective component, test cases for the semantic assignment and mediation should rather evaluate whether the combination of the <copy> and <extensionAssignOperation> activities preserve and/or assign the expected values to the involved context variables.

Preconditions: SBPELEE, Mock-up implementation of an SSB, Lifting/Lowering API and the actual Data Mediation.

Items under test: The extended assign activity

Postconditions: The context variables of the process need to be checked and compared to their expected content.

Artefacts used for testing: BPEL4SWS processes with at least one semantic assignment/mediation activity, XML fragments for the context variables to be examined.

Test Scenario 4)

Test Name: **Semantic Interaction Activities Tests**

Test Description: From the semantic activities added to BPEL4SWS through the extension mechanism, currently synchronous invocation has been implemented. Test cases should evaluate the correct invocation of the requested web service (by comparing the output to a predefined XML fragment). Since currently no integration with a SEE is possible, a dummy service component has to return the needed data, after lifting and lowering have been conducted (as stated in Section 3.1.5).

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

Preconditions: Lifting/Lowering API is in place, Component which returns ontological instances corresponding to the (semantic) web service to be invoked (to be exchanged with the SEE at a later stage)

Items under test: The semantic activities implementation

Postconditions: The invocation activity receives a reply from the invoked web service (lowered from WSML to XML elements), and the reply corresponds to the expected output of the web service

Artefacts used for testing: BPEL4SWS process with a semantic invoke activity, XML data corresponding to the WS input and output

6.3 Generating test cases from the scenarios

For each of the test scenarios described in the previous section, a set of corresponding (unit) test cases is going to be developed, in order to automate the testing of the targeted components and to ensure their future test compliance.

A major goal of setting up the test cases in this way is not only to check the current correctness of the developed components (by validating the output of the components against the supplied artefacts), but also to provide the means to test for regression bugs. Specifically in the early stages of the prototype development, where the design is still subject to a fair amount of change, as needed by requirements submitted by other work packages, changes and further development can introduce problems with already working components. Running the test case suite to safeguard against regression is thus proposed after a new feature has been added to the SBPELEE.

6.4 Mapping the Scenarios to the Use Cases

The Use Cases of course all depend on the availability of the SBPELEE for the development of their prototypes. For the M18 version of these prototypes, the SBPELEE couldn't yet be used without a use-case specific configuration process, as dependencies exist between the execution engine and the SEE with which it has to communicate, as well as the SSB to be used. Still, a set of test cases for the currently available processes of the Use Cases will be created (with a manual translation of processes to BPEL4SWS where these are not already available). These test cases will be evaluated against future version of the SBPELEE.

Additional test scenarios will be provided, once new features are added to the SBPELEE (see also the roadmap in Section 8 for a list of features which are going to be developed and their projected availability).

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

7 Licensing

7.1 License Agreement for SUPER-related Engine Extensions

The license scheme adopted for the SUPER extension to the starting BPEL engine (Apache ODE) is the Apache Software License (ASL) version 2.0. This license agreement was adopted according to the licensing schema investigation conducted in D6.1. ASL 2.0 is a very permissive licensing schema. It requires maintaining the copyright notice and disclaimer, but allows for the use and distribution as open or closed source software. This licensing mode allows downloading software for free, using and modifying the code, keeping modifications secret, and selling one's modifications. It does not require the resulting software to be open-source, as long as a notice informing about the origin of the software is provided. Here are the most important reasons for the choice of ASL 2.0:

1. It authorizes running the program for any desire use;
2. It authorizes copying and distributing the software without restriction;
3. It authorizes using and modifying the software for commercial purposes;
4. It authorizes improving and distributing copies of the derived works to the public;
5. It authorizes combining non-free and proprietary software in the same product and its further distribution;
6. It authorizes distributing the derived works under the most convenient licensing schema.;
7. It authorizes incorporating the derived works into proprietary commercial products;
8. It is free.

The integral text of the ASL 2.0 license is reported in Appendix A.1.

7.2 Apache ODE License Agreement

The SBPELEE engine is an extension of the Apache ODE BPEL engine, hence it includes a number of software libraries from the original Apache ODE project. All these libraries are based on the ASL 2.0 licensing scheme.

7.3 Licensing of Third Party Libraries

The Apache ODE engine comes with a set of libraries that are outside the ODE incubator project, others like the WSMO4J API have been added to support ontologies within SBPELEE. Most of them still rely on the ASL 2.0 licensing scheme. Others use a different licensing scheme. All the licensing schemes of the different included libraries allows anyway for redistribution of the libraries. It's not clear who retains the ownership of modification of the source code of the libraries, nevertheless this not a problem since the libraries are going to be used as is. Table 1 summarizes all the license of the third party libraries included in the project. For more details on the licensing schemes refer to Appendix A.

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

Table 1 Licensing Schema of the various third party libraries included in SBPELEE.

Library Name	Licensing schema
Activation	CDDL 1.0
ActiveMQ	ASL 2.0
Annogen	ASL 2.0
Apache Ant	ASL 2.0
Antlr	See Appendix A.2
Asm	BSD License
Axiom	ASL 2.0
Axis 2	ASL 2.0
Backport Util	CC Public Domain
Carol	LGPL 2.1
CGILIB	ASL 2.0
Commons-*	ASL 2.0
Derby	ASL 2.0
Dom4j	BSD License (modified)
Ehcache	ASL 1.1
Geronimo	ASL 2.0
Hibernate	LGPL 2.1
Howl-logger	BSD License
HSQLDB	BSD License
Jakarta-http-core	ASL 2.0
Jaxen	ASL 2.0
Jibx-run	BSD License
Jotm	BSD License
JUnit	CPL 1.0
Log4j	ASL 2.0
Mail	CDDL 1.0
MX4J	BSD License (modified)
Neethi	ASL 2.0
Quartz	ASL 2.0
Saxon	MPL 1.1
Spring	ASL 2.0
Stax	ASL 2.0
Woden	ASL 2.0
WSDL4J	ASL 2.0
WSMO4J	LGPL 2.1
WSTX	ASL 2.0
Xalan	ASL 2.0
XBean	ASL 2.0
Xerces	ASL 2.0
XML-apis	ASL 2.0
XmlSchema	ASL 2.0
XStream	BSD License

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

8 Installation and Usage

The engine's code base is not yet feature complete to be delivered and tested by end users. Due to dependencies to other components in the SUPER architecture that will be delivered later in the project additional design and implementation work will be done in the follow-up deliverable D6.6, which aims at complete BPEL4SWS implementation.

Due to the integration of several components with SBPELEE, as of now a configuration mechanism is provided for the prototype. This mechanism includes various configuration files and API for dynamic configuration. The basic properties of the engine that should be configured are:

- Engine intrinsic properties – memory usage, internal storage (for internal process model representation), various communication parameters.

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

9 Future Plans

The major driving forces for the future development of the SBPELEE are:

- Full implementation of the BPEL4SWS specification [SUPER-D1.3]
 - Implementation of the remaining BPEL4SWS extension activities.
 - Complete implementation of the semantically extended assign activity.
 - Support for SUPER related WSMO extensions.
- Full implementation of the EVO based event notification framework.
- Preparatory work for the integration with the Semantic Service Bus (upcoming deliverable D7.6)
- Integration with Dynamic Composition Reasoning Framework
- Preparatory work on control and monitoring APIs

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

References

[BPOF] Liliana Cabral, Stijn Heymans et al. Business Process Ontology Framework. SUPER Deliverable 1.1.

[SUPER-D1.3] Armin Haller et al. Process Ontology Language and Operational Semantics for Semantic Business Processes, SUPER Project Deliverable 1.3

[SUPER-D22] Leading Partner: MIP Semantic Business Process Lifecycle SUPER Project Deliverable D2.2

[SUPER-D41] Liliana Cabral, Emilia Cimpian, John Domingue, Christian Drumm, Adrian Mocan, Jussi Vanhatalo, Ingo Weber, Deliverable Description/Name, SUPER Project Deliverable 1.3

[SUPER-D71] Lead Partner: iSOCO. Integration and Interoperability Test Specification. SUPER Project Deliverable D7.1, September 2006.

[SUPER-D72] Lead Partner: NUIG. Semantic Web Services-based Business Process Architecture. SUPER Project Deliverable D7.2, September 2006.

[SUPER-D73] Luchesar Cekov (ONTO), Branimir Wetzstein (USTUTT) Semantic Web Services based Business Process API, SUPER Project Deliverable D7.3

[SUPER-D76] Branimir Wetzstein et al.: Semantic Service Bus. SUPER Deliverable 7.6 Draft. September 2007

[BPEL] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., et al. (2003). Business Process Execution Language for Web Services Version 1.1. Retrieved Nov 30, 2005, from <http://www.siebel.com/bpel>

[WS-BPEL20] Web Services Business Process Execution Language Version 2.0 Public Review Draft, 23th August, 2006 <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html>

[APACHE-ODE] Apache ODE engine website <http://incubator.apache.org/ode/>

[APACHE-TOM] Apache Tomcat website <http://tomcat.apache.org/>

[PROM] ProM process mining tool website <http://is.tm.tue.nl/~cgunther/dev/prom/>

[BSD] The BSD License. <http://www.opensource.org/licenses/bsd-license.php>

[CDDL] The Common Development and Distribution License (CDDL) Version 1.0. <http://www.opensource.org/licenses/cddl1.php>

[MPL] Mozilla Public License 1.1. <http://www.opensource.org/licenses/mozilla1.1.php>

[CPL] Common Public License Version 1.0. <http://www.opensource.org/licenses/cpl1.0.php>

[CC] Creative Commons Public Domain License. <http://creativecommons.org/licenses/publicdomain>

[LGPL] GNU Lesser General Public License 2.1. <http://www.opensource.org/licenses/lgpl-license.php>

[ASL1] Apache Software License 1.1. <http://www.opensource.org/licenses/apachepl.php>

[LNDKKCL07] An Execution Engine for Semantic Business Processes. In Proceedings of the 2nd International Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing (ICSoC207), Vienna, 2007.

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

Appendix A Licensing Schemas

A.1 Apache Software License (ASL) Version 2.0

The following text is reproduced from the Apache Software License, that can be found at <http://www.apache.org/licenses/LICENSE-2.0>

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions: (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and (b) You must cause any modified files to carry prominent notices stating that You changed the files; and (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License. You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

IP- Project / Programme	SUPER	Project - No	026850
	Process Execution Engine First Prototype	Work Package 6	
Document	Deliverable 6.2	Date	01.10.2007

A.2 ANTLR License

The following text is reproduced from the ANTR License Agreement, that is included within the ANTLR library and can be found at <http://www.antlr.org/license.html>.

We reserve no legal rights to the ANTLR--it is fully in the public domain. An individual or company may do whatever they wish with source code distributed with ANTLR or the code generated by ANTLR, including the incorporation of ANTLR, or its output, into commercial software.

We encourage users to develop software with ANTLR. However, we do ask that credit is given to us for developing ANTLR. By "credit", we mean that if you use ANTLR or incorporate any source code into one of your programs (commercial product, research project, or otherwise) that you acknowledge this fact somewhere in the documentation, research report, etc... If you like ANTLR and have developed a nice tool with the output, please mention that you developed it using ANTLR. In addition, we ask that the headers remain intact in our source code. As long as these guidelines are kept, we expect to continue enhancing this system and expect to make other tools available as they are completed.