Engineering Virtual Domain-Specific Service Platforms

Specific Targeted Research Project: FP7-ICT-2009-5 / 257483

Tool Suite for Virtual Service Platform Engineering (Interim)

Abstract

In INDENICA, the notion of Virtual Service Platforms (VSP) shall be leveraged to protect the investments into service-based applications against potential external negative influences and threats such as the heterogeneity of the involving service platforms, service discontinuation, service evolutions, etc. In D3.1, we already presented the view-based design time and runtime architecture for developing and tailoring VSPs described. In this report, we present the implementation of the tool suite for virtual service platform engineering based on the aforementioned view-based architecture.

Document ID:	INDENICA – D3.3.1
Deliverable Number:	D3.3.1
Work Package:	3
Type:	Deliverable
Dissemination Level:	PU
Status:	Final
Version:	1.0
Date:	2012-06-30
Contributing partners	SUH, UNIVIE, PDM, SAP, SIE, TUV, TEL

Project Start Date: October 1st 2010, Duration: 36 months

Version History

0.1	01. Mar 2012	Initial version
0.2	15. Mar 2012	Update Section 2 & 3
0.3	15. Apr 2012	Release the first version and update partners' feedbacks
0.4	08. May 2012	Revise and re-release the second version
0.5	12. June 2012	Update the guidance in Section 3
1.0	30. June 2012	Finalize the deliverable

Document Properties

The spell checking language for this document is set to UK English.

Table of Contents

1 Overview of the Tool Suite for Virtual Service Platform Engineering

The Tool Suite for Virtual Service Platform Engineering is based on the view-based design time and run-time architecture described in D3.1. The fundamental feature of the view-based architecture is to leverage the notation of architectural views to capture different aspects of a complex system and leverage the model-driven development paradigm to link those architectural views and generate code, configurations, runtime directives, and so on.

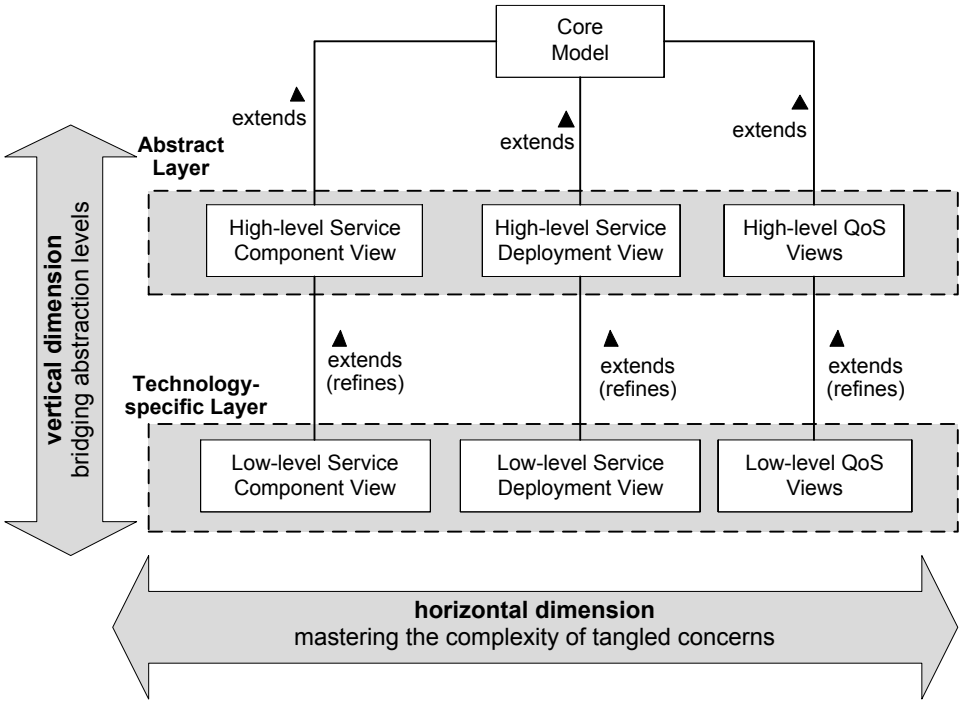


Figure 1 Overview of the view-based design time and runtime architecture

2 Tool Suite Implementation (Interim)

In this document, we report the interim release of the INDENICA Tool Suite for Virtual Service Platform Engineering (or Tool Suite for short, from now on).

2.1 Introduction

We realize the Tool Suite by facilitating the Eclipse Modelling Framework (EMF) and Eclipse MDD Framework. The most advantage of using Eclipse Modelling Framework is that we gain better integration and interoperability with existing development tools developed based on EMF Ecore, a MOF-compliant meta-model, and XMI, a standard for serializing models. The Tool Suite provides different editors for manipulating views, such as Service Component view, Deployment view, and so on. The template-based code generation rules are developed using the XPand language editor provided by Eclipse M2T project. Using these rules, we can automatically

generate virtual service platform configurations that can be deployed and executed in Apache Tuscany 1.6 for SCA.

The Tool Suite depends on following technologies and platforms:

Technology/Platform	Version	Website
Eclipse IDE	3.6+	http://eclipse.org
Eclipse Modelling Framework (EMF)	2.5+	http://eclipse.org/modeling/emf
Graphical Modeling Framework (GMF)	1.5+	http://www.eclipse.org/modeling/gmp
Eclipse Model to Text (M2T)	1.0+	http://www.eclipse.org/modeling/m2t
Xtext	2.0+	http://www.eclipse.org/Xtext

The Tool Suite has been developed and bundled in terms of Eclipse plug-ins. After installing the aforementioned required plug-ins, you can download the plug-ins bundle[1] and decompress it into the folder "dropins" of the Eclipse installation. Instead of searching and installing all aforementioned required plug-ins, which is tedious and error-prone, one can download a suitable *ready-to-use* bundle based on Eclipse Helios 3.6 to try out.

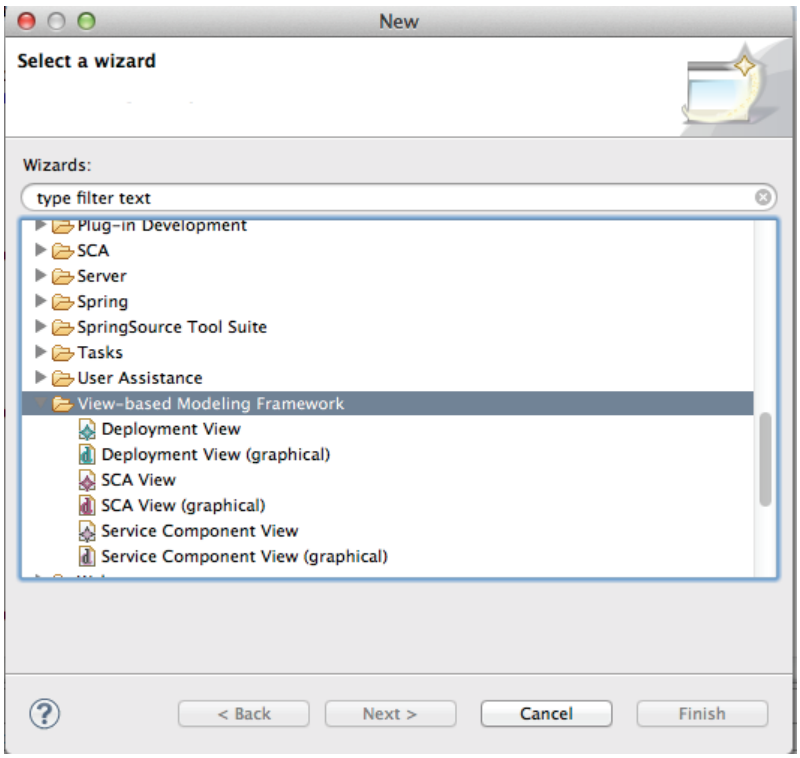
- Windows 32bit (https://swa.univie.ac.at/~huytran/vbmf/vbmf-w32.zip)
- Mac OS X 64bit (https://swa.univie.ac.at/~huytran/vbmf/vbmf-macox-64bit.dmg)

3 User's Guide

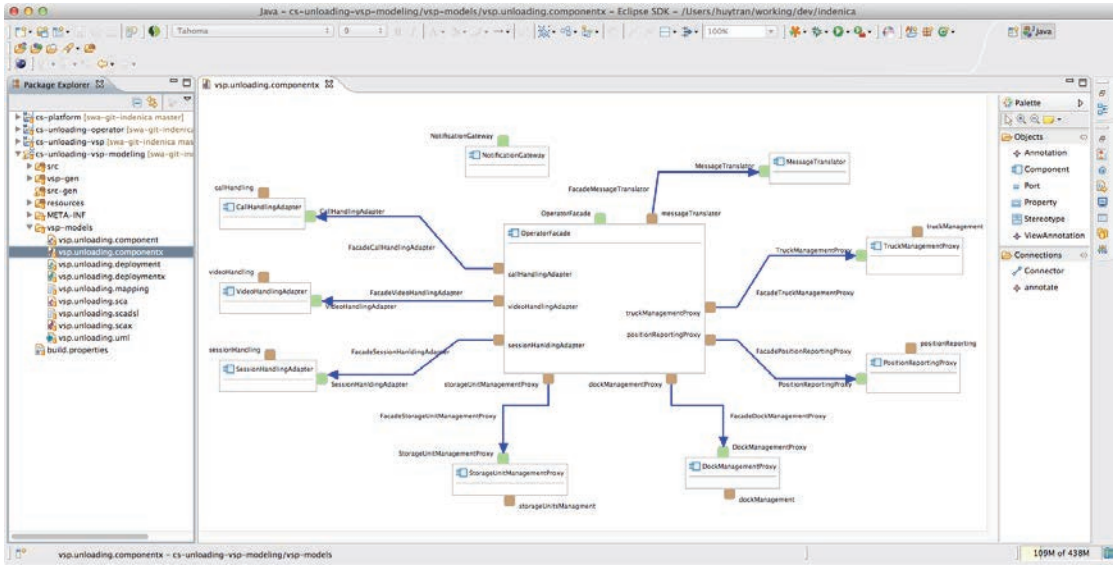
3.1 Create a high-level Service Component View

In Eclipse, go to "*File*", "*New*", "*Other*", or press Ctrl+N (Cmd + N in Mac OS X) to open the dialog. Go to the group "View-based Modeling Framework" to select a view model editor.

[1] https://swa.univie.ac.at/~huytran/vbmf/vbmf-plugins-only.zip



Add the components, ports, and connectors to describe the architecture of the Virtual Service Platform. One example shown in the following figure is derived from the „*Unloading Scenario*“ of the INDENICA Case Studies presented in D5.1.



3.2 Create or derive a low-level Service Component View (aka SCA View)

Similarly, the low-level Service Component View (SCA View) can also be created in the same manner. To save the effort of manually creating the SCA view, given the existing high-level counterpart created in the previous step, the Tool Suite supports model-to-model refinement. As the mechanism has not been totally integrated into Eclipse yet, a executable Java class is required to invoke the refinement.

Firstly, we need to define a mapping DSL that operationally specify how the high-level Service Component View will be refined to an SCA View. An example of the mapping DSL for the Unloading Scenario is provided below.

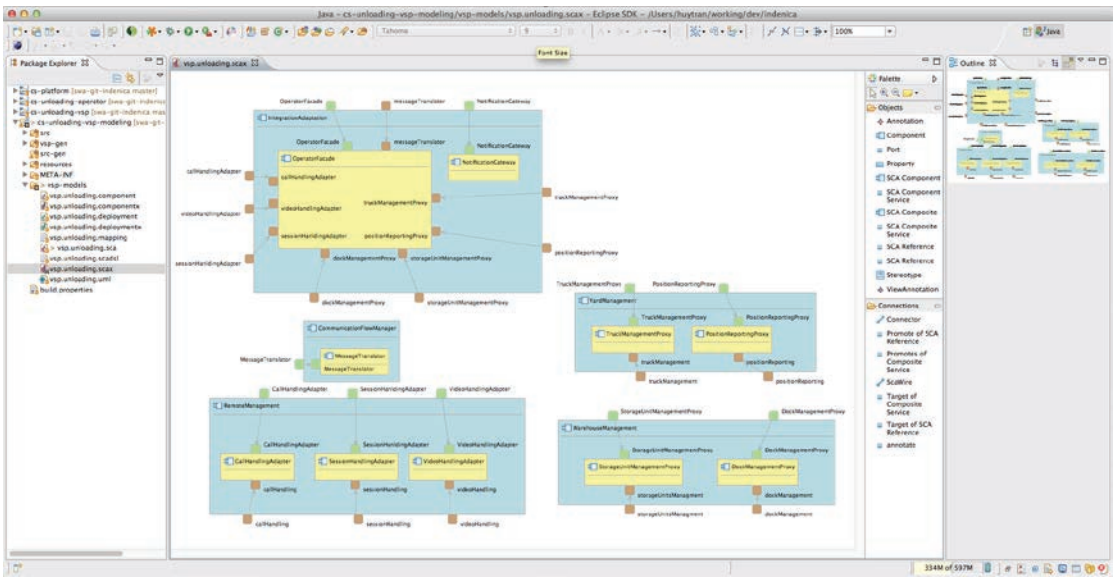
```
module vsp.unloading.mapping
Target View {
   name = vsp.unloading.sca
   nsURI = "http://indenica.eu/vsp/unloading/sca"

   Group G1 {
      source:
         vsp.unloading.component.OperatorFacade,
         vsp.unloading.component.NotificationGateway
      target:
         composite name = IntegrationAdaptation
         nsURI = "http://indenica.eu/vsp/integrationadaptation"
   }
   Group G2 {
      source:
         vsp.unloading.component.MessageTranslator
      target:
         composite name = CommunicationFlowManager
         nsURI = "http://indenica.eu/vsp/communicationflowmanager"
   }
   Group G3 {
      source:
         vsp.unloading.component.CallHandlingAdapter,
         vsp.unloading.component.SessionHandlingAdapter,
         vsp.unloading.component.VideoHandlingAdapter
      target:
         composite name = RemoteManagement
         nsURI = "http://indenica.eu/vsp/remotemanagement"
   }
   Group G4 {
      source:
         vsp.unloading.component.TruckManagementProxy,
         vsp.unloading.component.PositionReportingProxy
      target:
         composite name = YardManagement
         nsURI = "http://indenica.eu/vsp/yardmanagement"
   }
   Group G5 {
      source:
         vsp.unloading.component.StorageUnitManagementProxy,
         vsp.unloading.component.DockManagementProxy
      target:
         composite name = WarehouseManagement
         nsURI = "http://indenica.eu/vsp/warehousemanagement"
   }
}
```

For the time being, we provide suitable Java APIs for producing an SCA view given an input high-level Service Component view and a mapping DSL. In the next version of the Tool Suite, we will integrate more easy-to-use UI functions (e.g., right-click or context menu) in Eclipse that simply invoke these APIs.

```
   ScaView sv = Mapping.map(The_input_Service_Component_View, The_Mapping_DSL);
   Helper.serialize(sv, path_to_where_the_SCA_view_will_be_stored);
```

The resulting SCA View is somewhat following (after re-arranging the elements):



3.3 Verification of view models

To ensure the integrity and consistency of view models, the Tool Suite enables the verification of view models using an OCL-based model checking mechanism. This is implemented by using the Check language. For instance, we can write some simple OCL constraints for the Service Component view:

```
import core;
import component;
import sca;

context core::NamedElement ERROR
   "Name of " + this + " must not be empty" :
   name != null || name.length > 0;

context component::ComponentView ERROR
   "Port " + this + " is not a valid element of the Component View":
   element.typeSelect(component::Port).isEmpty
   ;

context component::Connector ERROR
      "Connector " + this + " must link a required port to a provided":
   source.kind == component::PortKind::REQUIRED
      && target.kind == component::PortKind::PROVIDED
   ;

context component::Property ERROR
   "Property " + this + " must have both name and value" :
   name != null && name.length > 0
   && value != null && value.length > 0;

context component::ComponentView ERROR
   "Ports are not valid elements of the Component View " + this :
   element.typeSelect(component::Port).isEmpty
   ;
```

Similarly, we provide relevant Java APIs for verification purposes. The following Java class illustrates the usage of the aforementioned APIs.

```
   static final Logger  log   = LoggerFactory.getLogger(Verification.class);

   public static void main(String[] args)
   {
      Issues issues = ComponentValidator.validateComponentView(CHECK_FILE,
COMPONENT FILE);
      if (issues != null && issues.hasErrors())
```

```
        for (MWEDiagnostic m : issues.getErrors())
        {
           log.error("\t" + m.getMessage());
        }

     issues = ComponentValidator.validateScaView(CHECK_FILE, SCA_FILE);
     if (issues != null && issues.hasErrors())
        for (MWEDiagnostic m : issues.getErrors())
        {
           log.error("\t" + m.getMessage());
        }
  }
```

3.4 Generate SCA Configuration

After achieving the SCA view (either as a refinement of the high-level Service Component View or a fresh one built from scratch), we need to define necessary SCA-specific annotations using the SCA DSL. To do so, we just create a new file with the extension „*.scadsl*“. The SCA DSL editor will be triggered and will support us to formulate the DSL. An example of the SCA DSL is shown in the following listing. The main responsibility of the SCA DSL is to specify the implementations of SCA components, the bindings of every SCA ports, and the representative interfaces of the bindings.

```
module vsp.unloading.scadsl
{
implementations {
      Java class = eu.indenica.vsp.impl.OperatorFacadeImpl implements
vsp.unloading.sca.IntegrationAdaptation.OperatorFacade
      Java class = eu.indenica.vsp.impl.NotificationGatewayImpl implements
vsp.unloading.sca.IntegrationAdaptation.NotificationGateway
      Java class = eu.indenica.vsp.impl.MessageTranslatorImpl implements
vsp.unloading.sca.CommunicationFlowManager.MessageTranslator
      Java class = eu.indenica.vsp.impl.CallHandlingAdapterImpl implements
vsp.unloading.sca.RemoteManagement.CallHandlingAdapter
      Java class = eu.indenica.vsp.impl.SessionHandlingAdapterImpl implements
vsp.unloading.sca.RemoteManagement.SessionHandlingAdapter
      Java class = eu.indenica.vsp.impl.VideoHandlingAdapterImpl implements
vsp.unloading.sca.RemoteManagement.VideoHandlingAdapter
      Java class = eu.indenica.vsp.impl.TruckManagementProxyImpl implements
vsp.unloading.sca.YardManagement.TruckManagementProxy
      Java class = eu.indenica.vsp.impl.PositionReportingProxyImpl implements
vsp.unloading.sca.YardManagement.PositionReportingProxy
      Java class = eu.indenica.vsp.impl.StorageUnitsManagementProxyImpl implements
vsp.unloading.sca.WarehouseManagement.StorageUnitManagementProxy
      Java class = eu.indenica.vsp.impl.DockManagementProxyImpl implements
vsp.unloading.sca.WarehouseManagement.DockManagementProxy
   }
   bindings {
      Web Service {
         uri = "http://localhost:4444/OperatorFacade"
      } binds {
         vsp.unloading.sca.IntegrationAdaptation.OperatorFacade
      }
      Web Service {
         uri = "http://localhost:5555/MessageTranslator"
      } binds {

   vsp.unloading.sca.CommunicationFlowManager.MessageTranslator.MessageTranslator,
         vsp.unloading.sca.IntegrationAdaptation.OperatorFacade.messageTranslator
      }
      Web Service {
         uri = "http://localhost:6666/CallHandlingAdapter"
      } binds {
         vsp.unloading.sca.RemoteManagement.CallHandlingAdapter.CallHandlingAdapter,
         vsp.unloading.sca.IntegrationAdaptation.OperatorFacade.callHandlingAdapter
      }
   Web Service {
         uri = "http://localhost:6666/SessionHanldingAdapter"
      } binds {
```

```
   vsp.unloading.sca.RemoteManagement.SessionHandlingAdapter.SessionHanldingAdapter,
         vsp.unloading.sca.IntegrationAdaptation.OperatorFacade.sessionHanldingAdapter
      }
      Web Service {
         uri = "http://localhost:6666/VideoHandlingAdapter"
      } binds {
         vsp.unloading.sca.RemoteManagement.VideoHandlingAdapter.VideoHandlingAdapter,
         vsp.unloading.sca.IntegrationAdaptation.OperatorFacade.videoHandlingAdapter
      }
      Web Service {
         uri = "http://localhost:7777/TruckManagementProxy"
      } binds {
         vsp.unloading.sca.YardManagement.TruckManagementProxy.TruckManagementProxy,
         vsp.unloading.sca.IntegrationAdaptation.OperatorFacade.truckManagementProxy
      }
      Web Service {
         uri = "http://localhost:7777/PositionReportingProxy"
      } binds {

   vsp.unloading.sca.YardManagement.PositionReportingProxy.PositionReportingProxy,
         vsp.unloading.sca.IntegrationAdaptation.OperatorFacade.positionReportingProxy
      }
      Web Service {
         uri = "http://localhost:8888/DockManagementProxy"
      } binds {

   vsp.unloading.sca.WarehouseManagement.DockManagementProxy.DockManagementProxy,
         vsp.unloading.sca.IntegrationAdaptation.OperatorFacade.dockManagementProxy
      }
      Web Service {
         uri = "http://localhost:8888/StorageUnitManagementProxy"
      } binds {

   vsp.unloading.sca.WarehouseManagement.StorageUnitManagementProxy.StorageUnitManagem
entProxy,

   vsp.unloading.sca.IntegrationAdaptation.OperatorFacade.storageUnitManagementProxy
      }
      JMS
      {
         uri = "jms:eu.indenica.vsp.NotificationGateway"
         initialContextFactory =
"org.apache.activemq.jndi.ActiveMQInitialContextFactory"
         jndiURL = "tcp://localhost:61616"
      } binds {

   vsp.unloading.sca.IntegrationAdaptation.NotificationGateway.NotificationGateway
      }
      Web Service {
         uri = "http://localhost:9999/CallHandling"
      } binds {
         vsp.unloading.sca.RemoteManagement.CallHandlingAdapter.callHandling
      }
      Web Service {
         uri = "http://localhost:9999/SessionHanlding"
      } binds {
         vsp.unloading.sca.RemoteManagement.SessionHandlingAdapter.sessionHandling
      }
      Web Service {
         uri = "http://localhost:9999/VideoHandling"
      } binds {
         vsp.unloading.sca.RemoteManagement.VideoHandlingAdapter.videoHandling
      }
      Web Service {
         uri = "http://localhost:9999/TruckManagement"
      } binds {
         vsp.unloading.sca.YardManagement.TruckManagementProxy.truckManagement
      }
      Web Service {
         uri = "http://localhost:9999/PositionReporting"
      } binds {
         vsp.unloading.sca.YardManagement.PositionReportingProxy.positionReporting
      }
      Web Service {
         uri = "http://localhost:9999/DockManagement"
```

```
      } binds {
         vsp.unloading.sca.WarehouseManagement.DockManagementProxy.dockManagement
      }
      Web Service {
         uri = "http://localhost:9999/StorageUnitManagement"
      } binds {

   vsp.unloading.sca.WarehouseManagement.StorageUnitManagementProxy.storageUnitsManagm
ent
      }
   }
   interfaces {
      Java interface = eu.indenica.vsp.operatorfacade.OperatorFacade describes {
         vsp.unloading.sca.IntegrationAdaptation.OperatorFacade
      }
      Java interface = eu.indenica.vsp.communicationflowmanager.MessageTranslator
describes {

   vsp.unloading.sca.CommunicationFlowManager.MessageTranslator.MessageTranslator
      }
      Java interface = eu.indenica.vsp.adapter.callhandlingadapter.CallHandlingAdapter
describes {
         vsp.unloading.sca.RemoteManagement.CallHandlingAdapter.CallHandlingAdapter
      }
      Java interface =
eu.indenica.vsp.adapter.sessionhandlingadapter.SessionHandlingAdapter describes {

   vsp.unloading.sca.RemoteManagement.SessionHandlingAdapter.SessionHanldingAdapter
      }
      Java interface =
eu.indenica.vsp.adapter.videohandlingadapter.VideoHandlingAdapter describes {
         vsp.unloading.sca.RemoteManagement.VideoHandlingAdapter.VideoHandlingAdapter
      }
      Java interface = eu.indenica.vsp.proxy.dockmanagementproxy.DockManagementProxy
describes {
         vsp.unloading.sca.WarehouseManagement.DockManagementProxy.DockManagementProxy
      }
      Java interface =
eu.indenica.vsp.proxy.storageunitsmanagementproxy.StorageUnitsManagementProxy
describes {

   vsp.unloading.sca.WarehouseManagement.StorageUnitManagementProxy.StorageUnitManagem
entProxy
      }
      Java interface = eu.indenica.vsp.proxy.truckmanagementproxy.TruckManagementProxy
describes {
         vsp.unloading.sca.YardManagement.TruckManagementProxy.TruckManagementProxy
      }
Java interface = eu.indenica.vsp.proxy.positionreportingproxy.PositionReportingProxy
describes {

   vsp.unloading.sca.YardManagement.PositionReportingProxy.PositionReportingProxy
      }
   }
}
```

3.5 Running SCA Configurations and Code with Apache Tuscany 1.6

Having the SCA View and SCA DSL, the Tool Suite can generate SCA configurations and launchers for deploying and executing the SCA runtime via the following Java APIs.

```
ScaGenerator.generateTuscany1Composite(SCA_VIEW, SCA_DSL, UML_CLASS_MODEL,
TARGET FOLDER);
ScaGenerator.generateTuscany1Launcher(SCA_VIEW, SCA_DSL, UML_CLASS_MODEL,
TARGET_FOLDER);
```

The generated configurations and code can be used to launch the Virtual Service Platform in Apache Tuscany 1.6

```
public class DistributedVirtualServicePlatform
{
```

```
   private final static org.slf4j.Logger log   =
org.slf4j.LoggerFactory.getLogger(DistributedVirtualServicePlatform.class);

   public static void main(String[] args) throws Exception
   {
      java.util.List<SCANode> nodes = new
DistributedVirtualServicePlatform().launch();
      log.info("The Virtual Service Platform has been started! Press ENTER to
exit...");
      System.in.read();
      log.info("Stopping the Virtual Service Platform...");
      for (SCANode node : nodes)
         if (node != null)
            node.stop();
      System.exit(0);
   }

   public java.util.List<SCANode> launch()
   {
      final SCANodeFactory nodeFactory = SCANodeFactory.newInstance();
      java.util.List<SCANode> nodes = new java.util.ArrayList<SCANode>();

      SCAContribution integrationAdaptation = new
SCAContribution("IntegrationAdaptation", "bin/node1");
      SCANode node1 = nodeFactory.createSCANode("IntegrationAdaptation.composite",
integrationAdaptation);
      node1.start();
      nodes.add(node1);

      SCAContribution communicationFlowManager = new
SCAContribution("CommunicationFlowManager", "bin/node2");
      SCANode node2 = nodeFactory.createSCANode("CommunicationFlowManager.composite",
communicationFlowManager);
      node2.start();
      nodes.add(node2);

      SCAContribution remoteManagement = new SCAContribution("RemoteManagement",
"bin/node3");
      SCANode node3 = nodeFactory.createSCANode("RemoteManagement.composite",
remoteManagement);
      node3.start();
      nodes.add(node3);

      SCAContribution yardManagement = new SCAContribution("YardManagement",
"bin/node4");
      SCANode node4 = nodeFactory.createSCANode("YardManagement.composite",
yardManagement);
      node4.start();
      nodes.add(node4);

      SCAContribution warehouseManagement = new SCAContribution("WarehouseManagement",
"bin/node5");
      SCANode node5 = nodeFactory.createSCANode("WarehouseManagement.composite",
warehouseManagement);
      node5.start();
      nodes.add(node5);

      return nodes;
   }
}
```

4 Developer's Guide

The whole source code of the Tool Suite for Virtual Service Platform Engineering is provided in the following INDENICA subversion repository (authentication needed):

https://repository.sse.uni-hildesheim.de/svn/Indenica/workpackages/wp3/implementation

Further details and guidance for the Tool Suite for Virtual Service Platform Engineering can be found (along with a demonstration movie) at:

https://swa.univie.ac.at/View-based_Modeling_Framework

5 Conclusions and future work

So far we describe the interim version of the INDENICA Tool Suite for Virtual Service Platform Engineering which will be finalized at M36. During this time period, we shall enhance tool support for the Tool Suite, enhance the integration with the architectural design decisions support developed in the scope of WP1 and the runtime supporting infrastructure developed in WP4.

Table of Figures