Engineering Virtual Domain-Specific Service Platforms

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

Report on Concepts for Tailoring and Extending Service Platforms

Abstract

This document gives an overview of business case for tailoring service platforms and outlines the current state and possible improvements for each of the industrial partners.

Document ID:	INDENICA – D5.2
Deliverable Number:	D5.2
Work Package:	5
Type:	Deliverable
Dissemination Level:	PU
Status:	final
Version:	1.0
Date:	2012-03-31
Author(s):	SIE, SAP, TEL

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

Version History

0.1	01. Feb 2012	Initial version
0.2	24. Apr 2012	TARC update
1.0	30. Sep 2012	Final version

Document Properties

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

Table of Contents

1 Introduction

This document gives an overview of business case for tailoring service platforms from each of the individual industrial partner's perspective. The main targets are the NetWeaver Cloud platform for SAP and various solution platforms including Silog. The document outlines the currently existing shortcomings of the platforms and shows some approaches for future improvements.

The document is structured as follows:

- The first section outlines business cases and scenarios for the industrial partners. We do not touch any technical details here, only the business perspective is illustrated.
- The second section contains an overview of the existing shortcomings, this time more from a technical perspective.
- The third section outlines possible improvement to overcome those shortcomings and illustrates areas of work which serve the industrial partners as starting points for future improvements.

2 Business Cases for Tailoring Service Platforms

This section outlines different business cases for tailoring service platforms.

2.1 SAP

SAP is active in different business areas. For this reason we'll divide examine potential business cases for the area of conventional on-premise computing and on-demand computing which is rapidly gaining momentum.

Variability is often present among different business processes of SAP's customers requiring adaptions of solutions SAP ships to these customers. With only a handful of customers the management of this variability might be done in an ad-hoc way, however, in case many customers need to served, dedicated variability management techniques are necessary. But at this point we must not forget that being able to serve different variants of a solution does not necessarily create a business value. Instead, variability management is only a technology which supports a certain business.

In the past SAP's technical platform(s) were designed to handle each and every situation in which the solutions that were built on top of these platforms were expected to work. This results in a very large, monolithic and complex platform covering all imaginable application areas. Such a platform is very hard to evolve; due to the large amount of side effects changes to the platform may have on variants that have been deployed to customers. In addition, due to the monolithic structure of the platform, components, which contain functionality a customer does not require, are still deployed. This results in a very large installation footprint with functionality that is never executed.

Today there's a clear customer demand for faster platform evolution. New platform features are expected to be introduced in shorter cycles. This is necessary to stay in a competitive position for both sides: for the platform provider who, of course, wants to prevent customers from a transition to competitors and also for the customer, who wants to adopt new functionality in order to improve solutions that were built on top of the platform.

It is obvious that supporting such a flexible and agile platform evolution model requires different approaches for developing and deploying the platform. The platform needs to be slim and lightweight, a one-size-fits-all approach will not hold. On the other hand, providing dedicated platforms for each single customer will not scale when the number of addressed customers reaches a number of thousands or even hundreds of thousands.

A possible solution to that is the provisioning of platforms not for single stakeholders but for different groups of stakeholders, which is a typical meet-in-the-middle approach. Members of such a stakeholder group share a large set of similar requirements and needs. However, the set of requirements does not need to be absolutely equivalent. The amount of "allowed" variation between those sets is

defined by the amount of effort that is needed to handle the differences. The lower the effort, the more diverse the stakeholder groups may become.

There is currently no proven approach about how to define such stakeholder groups best. In general there are two general ways of doing this; however, there might be others. This first approach of structuring these groups is along technical aspects. This means a group consists of stakeholders with similar functional requirements. A simple example would be the runtime support for processing BPM models, which would only be included when applications built on top of the platform, need to handle BPM models in some way. This approach is mainly for lowering costs on the provider side. A definition along technical features doesn't provide instant customer value; rather it helps to decrease costs for platform management and maintenance for the provider.

A second approach to structure stakeholder groups is along business aspects. For each group a set of functional and non-functional features may be defined. Membership to a certain group can be purchased at a certain price point, which is defined by the richness of the appropriate platform variant. A stripped-down version can be offered to entry customers while a premium version is offered to advanced customers at a higher price point. A flexible variation management system in the back can also enable the purchasing of extra features. An example would be a customer using the entry version of the platform, but with additional BPM support (to orient the example to the one given in the paragraph before). In addition non-functional aspects like for example the number of customers sharing one platform instance (which influences the performance) might be considered.

No matter how the stakeholder groups are structured, technically the required capabilities of the variation management system in the backend will stay the same.

The concept of providing platform variants on the customer side and an appropriate variation management system on the provider side can form the technical backbone for a whole new ecosystem of solution providers and consumers.

2.2 Siemens

Siemens is active in three different business types: Product Business, System Business and Solution Business.

In Product Business, Siemens creates products for a wide market that can be used by a variety of different customers. The development budget of a product is usually R&D funded; later on the sale of the product creates revenue. Examples for Siemens products are Simatic S7 Controllers for the industry automation domain or SiProtect protection devices for the power distribution domain.

In System Business, Siemens develops a single system on behalf of a single customer. The customer assigns the development and construction of a (usually large scale) system to Siemens and pays for the development and construction effort. In this kind of business, usually very little R&D budget is involved. Examples of systems build by Siemens are power plants, airports or warehouse management systems.

In Solution business, Siemens offers value adding services to its customers. In this kind of business, Siemens holds a set of resources necessary for the service, including

people, IT infrastructure, spare parts, etc. Customers order the service usually on a time-based contract. An Example for Solution business is Maintenance for a production plant.

System business is Siemens' main focus for Domain specific Service Platforms. Siemens wants to reduce the risk, development time and cost of these projects by increasing software reuse. In addition to that, also sales personal shall be focused to sell low-risk systems. This helps to avoid project crises which can consume the revenue up to ten successful projects in this kind of business.

3 Current state

3.1 SAP

SAP is currently developing a new Platform-as-a-Service (PaaS) product — NetWeaver Cloud (NWC) — that will allow customers to build custom applications in the cloud to complement and integrate with existing on-premise applications. NWC is intended as a quicker, more cost-effective alternative to custom-developing applications on NetWeaver on-premise, or other platforms. NWC will not only serve as a new basis for SAP itself, in addition SAP partners will also be able to build and sell applications on the platform as well.

NWC is meant to be a multi-programming paradigm environment. At the moment it supports Java as programming language for the backend as well as popular frameworks such as Spring and Ruby on Rails. There is also support for the In-memory database solution HANA, which can be integrated if necessary. It ships as a hosted-by-SAP solution.

Integration is a common concern when deploying SaaS applications (which NWC is clearly targeted at). With NWC, SAP aims to ease integration concerns not only with SAP applications but with non-SAP applications as well. NWC will use Web services standards for integration with non-SAP applications, such as REST. For the connection to SAP systems a dedicated connector is shipped with NWC.

SAP NetWeaver Cloud applications are based on the Java EE Web application model. It defines that the application frontend consists of Java Servlet, Java Server Pages and HTML components. It's possible to use the SAP UI development toolkit for HTML5 to create Web applications. SAP services enable a developer to easily add complex functionality to applications. Among others, NWC currently provides the following basic services through dedicated APIs:

- SAP Persistence Service
- SAP ID Service
- SAP Connectivity Service
- SAP Mail Service
- SAP Document Service

A usually sequence of steps with which it is possible to create a complete NWC application:

1. Implement the basic functional logic of the application, using Java EE technologies. It could be contained in the Servlet or JPA layer, for example.
2. Inside, embed the usage of services provided by the NWC platform.
3. Develop the user interface based on the Java EE user interface technologies, for example, JSP and HTML files.
4. Use the SAP UI development toolkit for HTML5 to implement a rich and attractive Web design.
5. Configure the deployment descriptors (web.xml, persistence.xml, etc.) and the manifest file.

6. Deploy the application on the local server, and test the expected behavior. Some components of NWC provide utilities for local testing.
7. Deploy your application to NWC.

When developing applications that run on the SAP NetWeaver Cloud platform, a developer can rely on certain Java EE standard APIs. These APIs are provided with the runtime service of the platform. They are based on standards and backward compatible as defined in the Java EE specifications. The following APIs are currently available:

- javax.activation
- javax.annotation
- javax.el
- javax.faces
- javax.mail
- javax.persistence
- javax.servlet
- javax.servlet.jsp
- javax.servlet.jsp.jstl
- javax.transaction

In addition to the standard APIs, NWC offers platform-specific services that define their own APIs that can be used from the platform SDK.

The NWC SDK contains a platform API folder for compiling Web applications. It contains the above content, that is, all standard and third-party API JARs (for legal reasons provided "as is", that is, they also have non-API content on which one should not rely on) and the platform APIs of the SAP NetWeaver Cloud services. Additionally, it's possible to add additional (pure Java) application programming frameworks or libraries and use them in applications. For example, one can include the Spring Framework in the application (in its application archive) and use it in the application. In such cases, the application should handle all dependencies to such additional frameworks or libraries and a developer should ensure that such additional frameworks or libraries are fully assembled inside the application itself. NWC also provides numerous other capabilities and APIs that might be accessible for applications. However, one should rely only on the APIs listed above.

As outlined above integration might be an important aspect of an NWC application. For this reason we give an example of how to use the NWC cloud connector to establishing connections to existing SAP backend systems.

There are three main stakeholders they will probably use the connectivity service:

- Application developers - develop the NWC application. They create a connectivity-enabled application by using NWC Connectivity API.
- Application operators - access the NWC account page and are responsible for productive deployment and operation of an application. They are also responsible for configuring the remote connections that an application might need.

- IT administrators - set up the connectivity to NWC in the customer's on-premise network.

On way to establish integration via the Cloud Connector is to use Connectivity Destinations: they are part of the NWC Connectivity Service and are used for the outbound communication of an NWC application to a remote system. They contain the connection details for the remote communication of an application. Connectivity destinations are represented by symbolic names that are used by on-demand applications to refer to remote connections. NWC Connectivity Service resolves the destination at runtime based on the symbolic name provided. The result is an object that contains customer-specific configuration details, such as the URL of the remote system or service, the authentication type, and the relative credentials. The currently supported destination type is *HttpDestination*. This destination type uses the HTTP protocol for data communication and is used for both Internet connections and on-premise connections. In addition one may define proxy types. There are currently two:

- Internet - The application can connect to an external REST or SOAP service on the Internet.
- OnPremise - The application can connect to an on-premise back-end system through SAP Cloud Connector.

The proxy type used for a destination must be specified by the destination property ProxyType. The property's default value (if not configured explicitly) is Internet.

The Cloud Connector also serves as the link between on-demand applications in NWC and existing on-premise systems. It combines an easy setup with a clear configuration of the systems that are exposed to NWC. In addition, it’s possible to control the resources available for the Java EE applications in those systems. Thus, one can benefit from existing assets without exposing the whole internal landscape. This is a relatively light-weight approach to integration.

3.2 Siemens

Currently, system projects heavily reuse Siemens products, like Simatic S7 controllers. Since these products are targeted to a wide market, they can be seen as technical platforms (for solving e.g. Safety or Real Time issues) that provide little or no domain specific logic. For domain logic, usually opportunistic reuse approaches are used. Also, the domain specific integration has to be done on a project base. On lower levels, usually proprietary integration approaches are used instead of the SOA paradigm.

The current approach has a number of issues:

- Maintenance of opportunistically reused components is more expensive and involves higher risk than the maintenance of strategically reused components.
- The approach leads to high integration efforts in the projects.
- Proprietary integration approaches usually come with proprietary monitoring techniques, leading to several different monitoring solutions in one plant instead of a unified monitoring solution.

Siemens tried to address those issues by providing domain specific platforms. These platforms included domain specific components for different domains. Often the business case for those platforms failed. Recurring reasons for these failures were too big and too complex platforms and platforms to rigid to tailor to the customers' needs. Also, the integration costs could not be lowered with this approach.

3.3 Telcordia

The main role of remote maintenance subsystem is to act as a communication platform for audio/video streaming, call initiation and management, and short message gateway. It also provides tools for near real-time monitoring and adaptation, i.e. it detects operational anomalies and act to them accordingly. In order to develop RMS platform, we will use and extend some of the functionalities of Mobicents platform.

3.3.1 Mobicents – a base platform for RMS

In order to develop RMS platform, we will use and extend some of the functionalities of Mobicents platform. Mobicents is a VoIP platform, which provides support for SIP (Session Initiation Protocol) [4] signalling and RTP/RTSP (Real-time Streaming Protocol) [5] for streaming. In addition, it brings several monitoring and adaptation components based on Java Management Extensions (JMX) technology and Simple Network Management Protocol (SNMP).

More specifically, Mobicents VoIP services are based on SIP servlet technology. SIP servlet is a server side interface for message exchange between VoIP-based applications. Main components of SIP server are:

- Proxy service - routing of call requests,
- Registrar service - mapping user names to network addresses,
- Presence service – accepting, storing and distributing SIP Presence information, eg. User status

The main role of SIP server is to handle SIP signalling exchange between VoIP clients, ie. to provide call controll capabilities for VoIP. However, it does not have any control over media streams transmitted using RTP protocol.

In order to process media streams, Mobicents platform provides a separate unit – Media Server. It is open source implementation of the VoIP network element responsible for media handling. Mobicents Media server provides support for both distributed and centralized services including circuit switch voice/video, announcements, tones, etc. It is capable of handling following Audio and Video codecs: G.711 u-law, G.711 A-law, GSM, Speex (narrow band), G.729, H.261. This server is especially useful to support legacy systems, ie. provide media transcoding for VoIP clients, which do not have compatible audio/video codecs. Additionally, Media Server can act as Conference Access Point. It means that all media streams are synchronized and optionally transcoded before they are send to VoIP clients.

The Mobicents Media server can operate in pair with various call controllers such as Mobicents Jain SLEE [6] application server or Mobicents SIP Servlet Container (JSR-309) [7].

3.3.2 RMS description

To build a RMS platform we will be using the Mobicents SIP Servlets Server - a certified implementation of the SIP Servlet v1.1 (JSR 289) specification [7] that can run on top of either the JBoss Application Server or the Tomcat Servlet Container. To process media streams, we will use the Mobicents Media Server implemented using JBoss Microcontainer kernel, which allows archiving maximum flexibility. It gives the ability to adopt media server for task specific and/or extend the functions of the media server by installing additional media processing components. This will provide necessary infrastructure for SIP-based voice/video communication services, eg. real-time streaming of multimedia content, point-to-point or group.

More specifically, the RMS platform will provide:

- Call session management - it will be based on a session Initiation Protocol (SIP) that is used to set up, modify, and terminate a session between two endpoints. Currently, Mobicents VoIP services provide the functionality of VoIP session establishment, presence or registration information. We will extend such call control mechanism by adding user specific information: localization, case, place in the corporate structure.
- Remote monitoring – Mobicents provides several mechanism for platform monitoring such as JMX or SNMP. It is possible to perform some simple monitoring tasks such as tracking a number of VoIP sessions. However, present Mobicents monitoring tools do not have knowledge of events occured on other platforms. Therefore, it is hard to predict future events. RMS platform will provide monitoring interface, built on top of Mobicents, which will deliver information about call control events, streamed media and platform parameters. Based on Mobicent functionality, call control can be monitored via:
    - Through the industry standard Simple Network Management Protocol - SNMP
    - Through Jopr/RHQ – Mobicents plugin which provides administration, monitoring, alerting, operational control and configuration in an enterprise setting with fine-grained security and an advanced extension model.
    - SIP servlets – SIP events can be monitored using Mobicents VoIP server components, eg. SIP Proxy, SIP registrar, etc.
    - JAIN SLEE – SIP events are monitored using SLEE SIP Resource Adaptor being part of SLEE Service Building Block. JSLEE is an event driven application server with protocol agnostic architecture, spanning any legacy or potential future protocols.

Parameters of Mobicents platform can be monitored and management via:

- SNMP [8]
- Java Management Extensions (JMX) – controlled resources are represented by objects called MBeans (Managed Bean).
- Mobicents provides system metrics (such as memory usage, number of VoIP session) which can be management via Jopr/RHQ plugins. Additionally, it is

possible to gather information of specific system metrics using either SNMP or JMX interface.

3.3.3 Disadvantages of existing solutions

The RMS platform is based on SIP protocol and is using Mobicent platform, hence some problems related to these technologies apply also to the RMS:

- SIP is a light weight application level protocol for fast call setup and termination [3], and, as such, it is a strong candidate for multimedia and voice communication over IP. However, the SIP protocol does not cover directly many communication services, so there is a need for an extensible service integration environment that allows applications to integrate SIP as part of the business transaction process [3]. At the moment SIP is under constant expansion to cover new services that are not currently at its scope, which results in frequent SIP standards and application revision and updates [WSIP]. As new versions of SIP user agent endpoints are created, interoperability among them becomes a serious issue. Furthermore, SIP is not XML and WEB-service based [2], hence integrating SIP with business transactions through XML is very difficult, and existing solutions have many limitations.
- As above-mentioned, the RMS platform is based on Mobicents - a high-performance core engine for Service Delivery Platforms (SDP) and IP Multimedia Subsystems (IMS). It contains several integration points with WEB, SOA, and CRM end-points, and it enables the composition of service-building blocks [1]. Although the above-mentioned features enable rapid implementation of new VoIP services, the platform itself cannot be easily used as a platform-building block from the product line engineering point of view. RMS API is not coherent with other platforms, so each usage of RMS in platform composition must be followed by an integration process. In addition, there are some technologies that carry common functionality for RMS and other platforms, but they cannot be easily combined to create one coherent system. An example of such technology is monitoring. Although there are several monitoring tools available on Mobicents, which allow monitoring the activities of various components, they cannot be directly used to monitor the performance of a platform composed of distinct systems. Moreover, RMS does not provide tools to interact and exchange information with monitoring components of other platforms.
- RMS is using monitoring tools provided by Mobicents platform, but it lacks an adaptation engine. In other words, the platform can monitor performance of its different components, but it has no tools to use the knowledge gained by monitoring to adapt its behaviour to minimize the impact of detected problems. In particular, it cannot react to events fired by other platforms, so adaptation of platform compositions is not feasible.

4 Concepts for Improvements

4.1 SAP

The possible transition for SAP from an on-premise business to an on-demand business can be illustrated in the following picture:

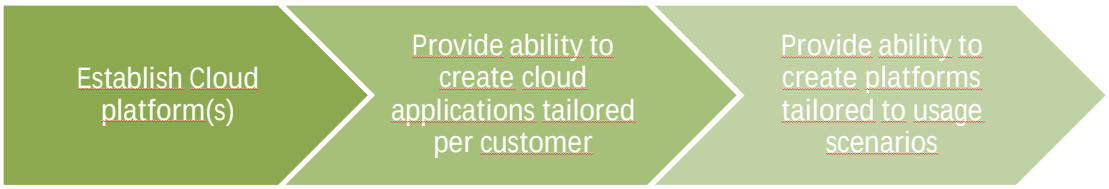

Figure 1 Transitioning from On-Premise to On-Demand

We can outline three phases:

- Phase 1: establishing (an) appropriate cloud platform(s)
- Phase 2: provide the ability to tailor cloud applications to single customers
- Phase 3: provide the ability to customize platforms for certain usage scenarios

Phase 1 establishes clearly the technological basis for the succeeding phases. This stage can describe where we are today. Appropriate cloud platforms are currently in the works or have already been released to customers. SAP NetWeaver Cloud is an example.

Phase 2 needs to incorporate technologies for lightweight customization of cloud applications. Customization means tailoring that is executable by business specialists as well as tailoring possibilities for developers. Variation management techniques in the solution space can be applied here.

Phase 3 is further in the future. We can also apply variation management solutions for the solution space, but this time on a different level. However, we only can approach 3 after phase 2 has been understood. It is clearly research work and INDENICA can actively contribute to phase 3.

4.2 Siemens

Siemens' Concept to overcome the shortcomings of the current approach is to create a product line of pre-integrated products and domain services. For modelling the domain variability, advanced variability modelling techniques are necessary to avoid systems to rigid for the customers. Especially, cardinality-based variability modelling, non-boolean variability and integration of domain specific languages for modelling subdomains (as described in [10]) will help to create integrated variability models for Siemens' Business Domains.

Also, the multi-product-line approach will help to create profitable platforms and increase the flexibility for system business. For each subdomain, a product line can

be created. This (component) product lines can be reused in a variety of system level product lines, therefore increasing the reuse while limiting the complexity. An example for these kinds of multi-product-line is shown in the figure below:

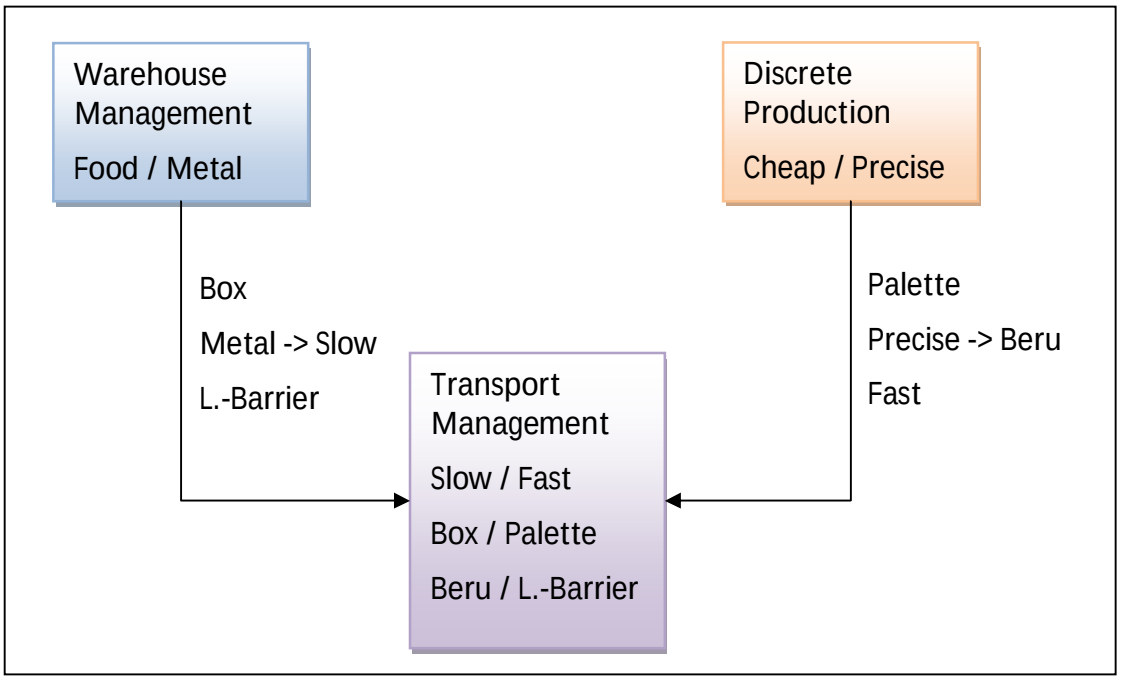

Figure 2 Multi-product-line

The figure shows two system product lines, one for warehouse management and one for discrete production. In addition, a subdomain product line for transport management is shown.

Warehouse Management provides a variability point whether the system is used in Food industry or Metal industry.

Discrete Manufacturing has a high precision option.

Transport Management can be used for slow and fast transports. The system can control Box conveyors and palette conveyors, and can find material position either by beru or light barrier sensors.

Multi-product-line techniques allow deducing variability configurations of subdomain platforms from system platforms' configurations. Therefore, the complexity of the configuration of the system platforms is not increased by using subdomain platforms, while the reuse rate of the subdomain platform is increased.

By combining advanced variability techniques with multiproduct line approaches, the flexibility needs of the system business can be met while high reuse makes the product lines profitable.

4.3 RMS and VSP integration

4.3.1 Overview

In order to ensure the possibility of monitoring and adapting the whole platform (composition of different platforms), we have developed monitoring and adaptation

engines in VSP that communicate with each platform separately. More specifically, each platform contains a monitoring module that collects data about the performance of its components. This information (filtered by the internal filter) is sent through the monitoring interface to the monitoring engine in VSP. The monitoring engine, based on reasoning logic, analyzes incoming data (from all platforms) and informs the adaptation engine about detected patterns of the platform's behaviour [9]. Based on the received information, the adaptation engine [9] chooses adaptation rules that maximize the performance of the whole system. A request-of-adaptation-rule execution is sent to the specific platform or set of platforms via the adaptation interface.

In order to follow this pattern, we will extend the RMS platform with a monitoring and an adaptation interface.

- The monitoring interface receives data from the RMS platform, converts them into standardized event model provided for all platforms (unification is necessary for efficient monitoring) [9], and it ensures their successful delivery to VSP.
- The adaptation interface can receive requests-of-adaptation-rule execution from the adaptation engine at VSP. Based on received request, the adaptation interface can trigger an adaptation procedure, which will, for example, change the specific parameters, runtime configuration of the specific components, or runtime configuration of the RMS platform.

Being equipped with the above mentioned interfaces (the monitoring and control interfaces, which are described in detail below) and using functionality provided by VSP, the RMS platform will be able to react on the events fired on other platforms in the whole system.

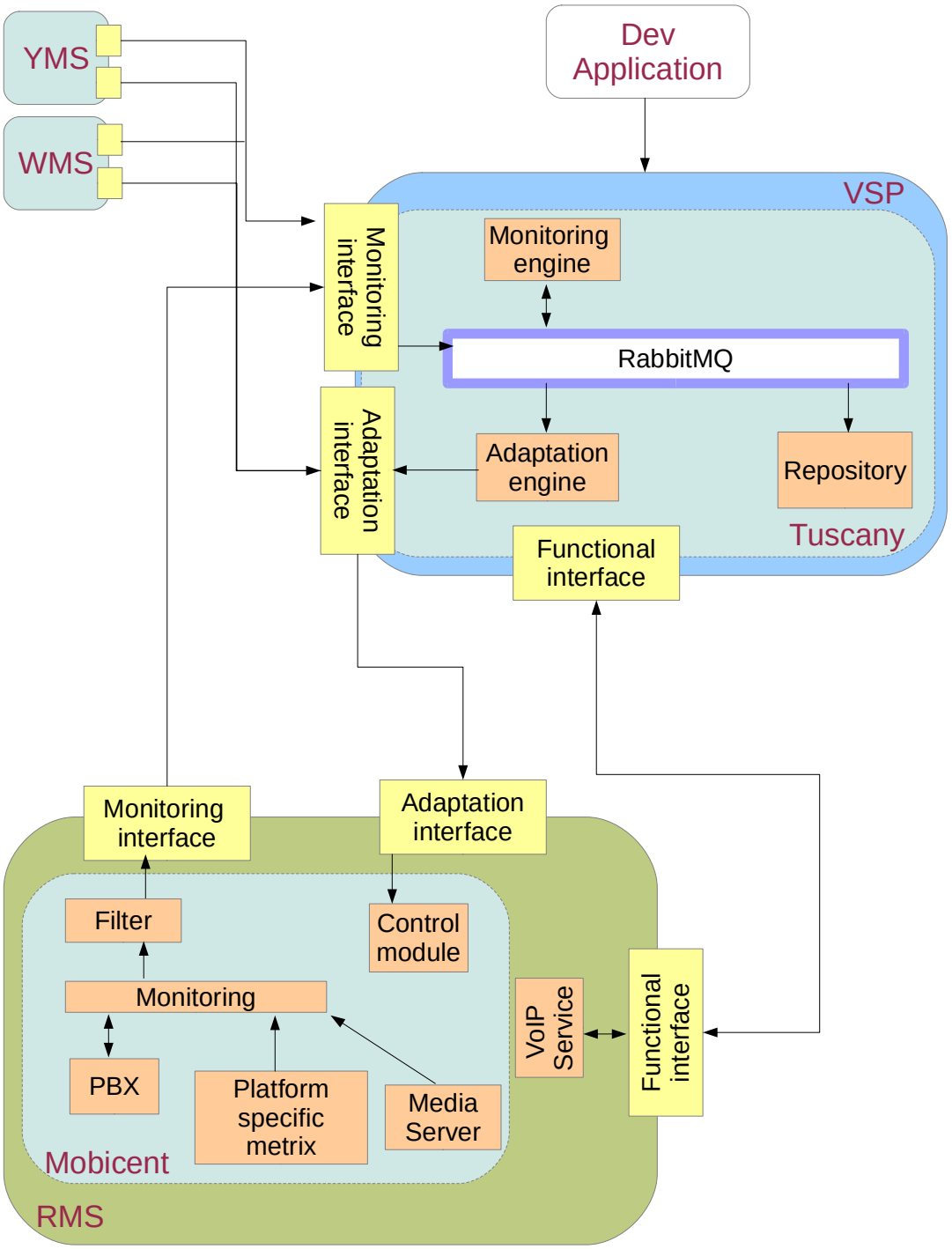

Figure 3 RMS and VSP integration

Integration of VSP and RMS base platform also require design a new functional interface. In the case of RMS it is a VoIP interface, which is responsible mainly for call control and session establishment. As a result, VSP platform will provide universal VoIP interface able to replace one base platform with another one, eg. Mobicents with Astersik. Moreover, RMS will be also signalling protocol agnostic, ie. it will be possible to replace, for example SIP with XMPP.

4.3.2 Extended monitoring interface

In the INDENICA project we are working on development and unification of Mobicents monitoring interfaces to provide single VSP interface. Such interface can be easily reusable by other base platforms connected to VSP. As was described above Mobicents provides several tools for system management and monitoring. In our work we would like to integrate JMX and SNMP interfaces. Therefore, it is necessary to define standard format of event generated by RMS platform. Data gathered in JMX has MBeans object format; whereas, SNMP data is stored date in MIB (Management Information Base).

The type of generated events on the Mobicents base platform should be set via Jopr/RHQ interface. Such operation can be done during the platform configuration. However, VSP should have the ability to select events in the runtime. For that purpose we are developing the event filter service as a part of base platform.

Monitoring interface at the base platform is also responsible for translation of Mobicents events to VSP event format. Translated events are sent and stored at the RabbitMQ messaging server.

4.3.3 Extended adaptation interface

Mobicents supports adaptation of a RMS platform by switching platform parameters and configuration during the runtime using SNMP/JMX interface. However, changes in the configuration made via SNMP/JMX are not persistent and are not visible after service restart. Therefore, configuration files should be treated as an initial set of parameters.

Figure 3 indicates control module which is responsible for reacting on information received from VSP adaptation interface. Next, control module adapts Mobicents parameters according to received events. This part of RMS platform will be integrated with JMX/SNMP management interfaces.

4.3.4 Extended functional VoIP interface

Besides monitoring and adaptation interfaces, RMS has also “Functional Interface”, which is responsible for establishing the VoIP request and call management. Such interface is based on web-service technology. Therefore, application developer can establish the VoIP connection without the knowledge of signalling protocols. Hence, it will be possible to change the base VoIP platform from Mobicents to other solution.

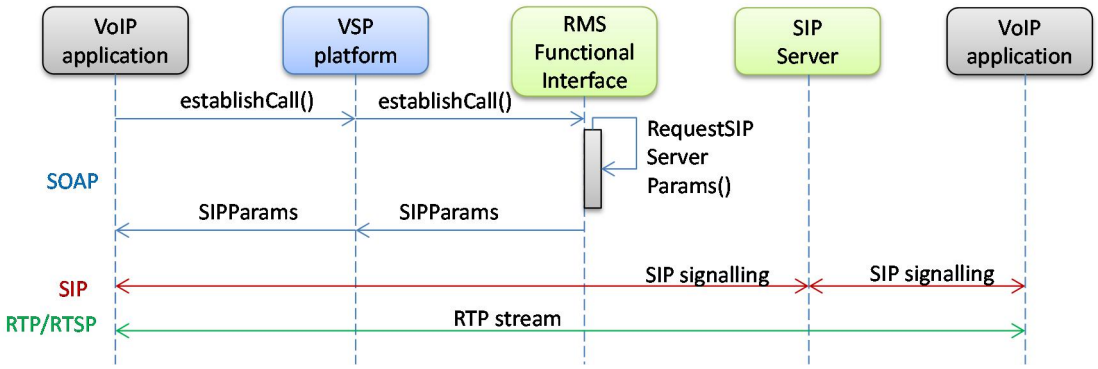

Figure 4 Example of VoIP connection on integrated RMS and VSP platforms

Figure 4 shows the message flow of the VoIP/functional interface on the integrated VSP platform. In this scenario, two VoIP clients tries to establish a call. At the beginning VoIP client use the web-service interface on the VSP platform and sends a request *establishCall()* using SOAP protocol. Next, application router on VSP platform forwards such request to RMS base platform. Functional interface process the incoming request (*RequestSIPServerParams()*) and send back (SOAP) necessary parameters needed for call establishment by VoIP client, eg. SIP server IP. In the next step VoIP application use appropriate signalling protocol (SIP/XMPP) to create a connection between VoIP clients. If it is a SIP protocol than all signalling data is passed through SIP server (SIP Proxy) on the Mobicents platform. Finally, VoIP clients send encoded media data using RTP/RTSP protocols.

5 Conclusion

This document gives an overview of business case for tailoring service platforms from each of the individual industrial partner's perspective. The main targets are the NetWeaver Cloud platform for SAP and various solution platforms including Silog. The document outlines the currently existing shortcomings of the platforms and shows some approaches for future improvements.

During the runtime of the INDENICA project the industrial partners will pursue opportunities to actively apply research results to improve the current technical issues in their existing platforms.

Table of Figures

References

[1]. Jboss, “Mobicents Communications Platform”, See http://www.mobicents.org.

[2]. Wu Chou, Li Li, and Feng Liu. “Web services for communication over IP.” *IEEE Communications Magazine,* 2008 136-143.

[3]. Feng Liu, Wu Chou, Li Li, and Jenny Li. 2004. “WSIP - Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP”. In *Proceedings of the IEEE International Conference on Web Services* (ICWS '04). IEEE Computer Society, Washington, DC, USA,

[4]. Session Initiation Protocol (SIP) specification, 2002, on-line at: http://www.ietf.org/rfc/rfc3261.txt?number=3261.

[5]. Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[6]. Sun Microsystems, "JSLEE Specification - JSR 240", See http://jcp.org/en/jsr/detail?id=240.

[7]. BEA, Sun Microsystems, "SIP Servlet Specification - JSR 289", 2008. See http://jcp.org/en/jsr/detail?id=289.

[8]. D. Harrington, R. Presuhn, B. Wijnen, “[RFC 3411], An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks”, December 2002

[9]. INDENICA Deliverable D4.2.1, „Tool Suite for Deployment, Monitoring & Controlling of Virtual Service Platforms (Interim)“, 2012

[10]. INDENICA Deliverable D2.1, „Open Variability Modelling Approach for Service Ecosystems”, 2011