Institute of Architecture of Application Systems
University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Master's Thesis No. 3347

Extending an Open Source Enterprise Service Bus for Dynamic Discovery and Selection of Cloud Data Hosting Solutions based on WS-Policy

Mansur Uralov

Course of Study: INFOTECH

Examiner: Prof. Dr. Frank Leymann
Supervisor: Steve Strauch
Commenced: May 31, 2012
Completed: November 30, 2012

CR-Classification: C.2.4, D.2.11, H.3.4

Abstract

As part of Cloud computing, the service model Platform-as-a-Service (PaaS) has emerged, where customers can develop and host internet-scale applications on Cloud infrastructure. The Enterprise Service Bus (ESB) is one possible building block of a PaaS offering, providing integration capabilities for Service-Oriented architectures. Dynamic service discovery and selection support for an ESB increases flexibility of the application composed of reusable services in the Cloud and gives providers the possibility react faster on changes in the market.

In this master's thesis we specify, design and implement Dynamic Discovery and Selection of Cloud Data Hosting Solutions for an open-source ESB. Provided dynamic service discovery and selection endpoint/service allows users of tenants to send requests with attached policies, while tenants register Cloud Data Hosting Solutions with the policies that describe their capabilities. To provide uniform policy language a new WS-Policy Assertion Language is created and specified that is used to express functional and non-functional properties of Cloud Data Hosting Solutions. By matching a policy in a request and policies of Cloud Data Hosting Solutions, a suitable Cloud data store service is discovered. Moreover, we ensure data isolation between tenants while providing dynamic service discovery and selection.

Contents

List of Figures

List of Tables

List of Listings

1. Introduction

1.1. Motivating Scenario

In this section we describe a scenario that will be used to validate the implementation of the master's thesis. In the scenario, the three requesters are demonstrated that belong to two companies (Company A and Company B). Requesters search for the Cloud data stores to migrate or build their database layer in the Cloud, while companies provide services for their requesters. In this work, companies are considered to be tenants and the requesters are user of the tenants.

The companies register their services to make them available to their users. As you see in the Figure 1.1, Company A registered the Cloud data store services Amazon RDS MySQL Engine [ARD], Amazon SimpleDB [ASD], and Xeround [Xer], while the Company B registered Amazon RDS MySQL Engine, Amazon RDS PostgreSQL Engine, Amazon DynamoDB [ADy], SQL Database [WAS], and Google Cloud SQL [Goo] Cloud data store services.

In this scenario (see Fig. 1.1), the users of the tenants specify their requirements in policy documents and attach them to the request messages that they send to the ESB. Based on the requirements specified in the policy documents attached to the message requests, a corresponding Cloud data service fulfilling the requirements is discovered.

In this work we describe concept and design as well as implement a prototype of the work based on concepts and design solutions to provide dynamic service selection and binding for Cloud Data Hosting Solutions [SKLU11]. According to our scenario, we need to develop dynamic service selection functionality in an ESB to increase flexibility of the application composed of reusable services in the Cloud and gives providers the possibility react faster on changes in the market. While providing dynamic service discovery and selection, data isolation between tenants should be maintained.

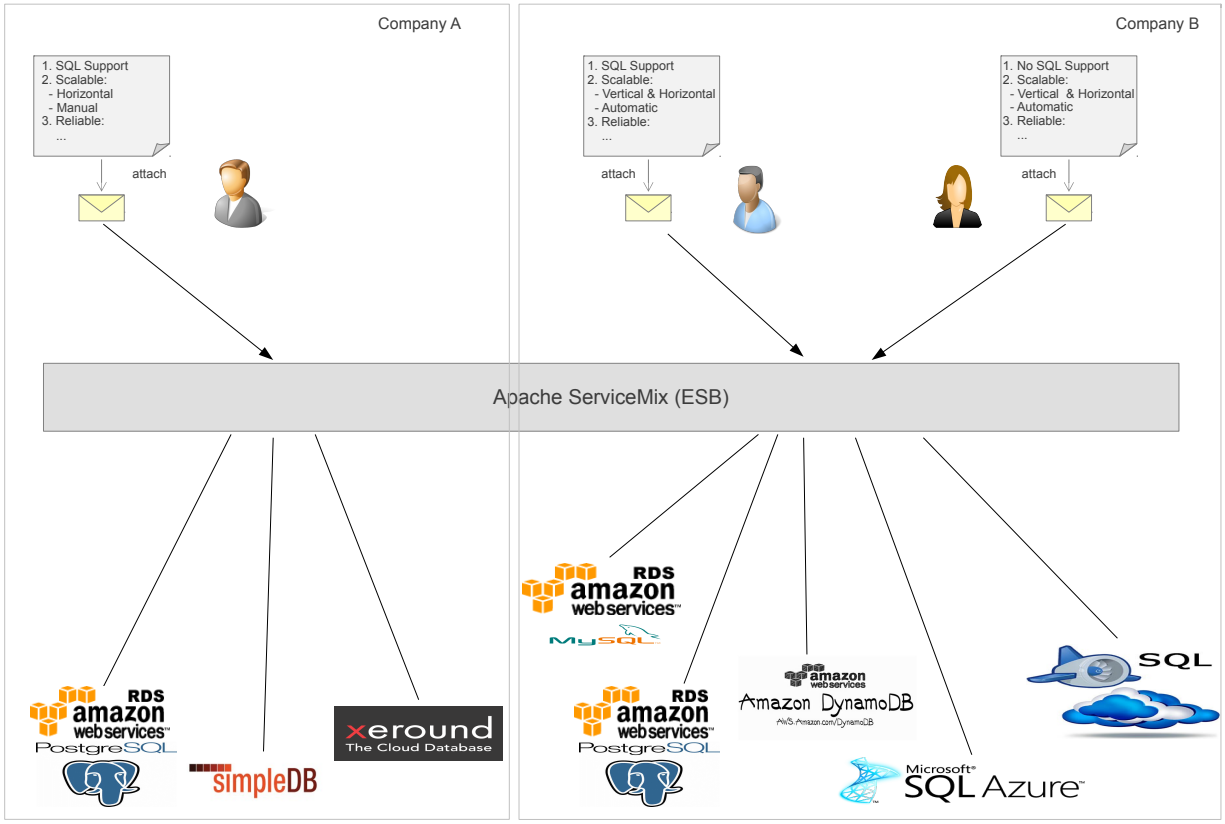

Figure 1.1.: Motivating Scenario.

1.2. Scope of Work

Our goal in this master's thesis is to specify concepts, designs and realize a prototype in order to provide Dynamic Service Discovery and Selection of CDHS based on WS-Policy in open-source Apache ServiceMix. As Dynamic Service Discovery and Selection increases flexibility of the application composed of reusable services as well as gives providers the possibility react faster on changes in the market, we develop the functionality for Apache ServiceMix. This work also supports multi-tenancy [AGJ+08] that helps to isolate data between tenants during service discovery and selection. Furthermore, we employ cache mechanism for data retrieval from the databases outside Apache ServiceMix to improve performance.

As service providers need a uniform policy language to specify the functional and non-functional properties of their Cloud data store services, a new policy language should be created and specified in this work. With the help of this policy language service providers express capabilities of their Cloud Data Hosting Solutions [SKLU11], whereas requesters specify their needs.

To register services with policies and to attach services to already existing services, JBI Multi-tenancy Multi-container Support [Muh11] Management Application interface and service registry are extended.

Out of scope is the design and implementation of different load-balancing strategies for service selection.

1.3. Outline

The current work is organised as follows:

- **Fundamentals, Chapter 2** – in this chapter, the relevant literatures that covers the fundamentals of the master's thesis are reviewed and specified.
- **Related Works, Chapter 3** – in this chapter, we investigate the works that have relation to our work and position our work towards this state of the art.
- **Domain-Specific WS-Policy Extension for CDHS, Chapter 4** – in this chapter, a new WS-Policy Assertion language is specified that is used to specify functional and non-functional properties of Cloud Data Hosting Solutions.
- **Concept and Specification, Chapter 5** – in this chapter, as a result of investigating the related works and surveying fundamentals, the conceptual solutions are found and specified for the specified functional and non-functional requirements of Dynamic Service Selection and Discovery support for multi-tenant Enterprise Service Bus (ESB).
- **Design, Chapter 6** – in this chapter, architectural and technological solution for the concepts and specified functional and non-functional requirements are designed.

- **Implementation and Validation, Chapter 7** – in this chapter, the challenges occurred during implementation are specified in more detail. In addition, the result of the implementation is validated.
- **Conclusion and Future Work, Chapter 8** – this chapter summarizes the outcomes of this work and suggests future extensions to the developed system.

1.4. Definitions and Conventions

The following definitions and abbreviations should be inspected for understanding the descriptions in this work. They are used throughout the document.

The following eXtensible Markup Language (XML) namespaces are used in this document and referenced by the listed prefix:

Prefix	Namespace	Specification
cdhs	http://iaas.uni-stuttgart/cdhs	This document
dsds	http://iaas.uni-stuttgart/dsds	This document
camel	http://camel.apache.org/schema/spring	[APA11a]
osgi	http://www.springframework.org/schema/osgi	[CHLP09]
soap	http://www.w3.org/2003/05/soap-envelope	[SOA07]
soap12	http://schemas.xmlsoap.org/wsdl/soap12	[WSD06]
sp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702	[OAS09]
wsdl	http://schemas.xmlsoap.org/wsdl	[WSD01]
wsp	http://www.w3.org/ns/ws-policy	[WSP07b]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	[OAS06]
xs	http://www.w3.org/2001/XMLSchema	[XSD04]
xsi	http://www.w3.org/2001/XMLSchema-instance	[XSD04]

Table 1.1.: XML namespaces referenced in this document via listed prefix.

List of Abbreviations

The following list contains abbreviations used in this document. Full names by convention not valid or not used anymore are marked as deprecated.

BC Binding Component

EAI Enterprise Application Integration

ESB	Enterprise Service Bus
JAR	Java Archive
Java EE 5	Java Platform, Enterprise Edition v. 5
JBI	Java Business Integration
JBIMulti2	JBI Multi-tenancy Multi-container Support
SmxDSDS	ServiceMix Dynamic Service Discovery and Selection Support
CDHS	Cloud Data Hosting Solutions
DSDS	Dynamic Service Discovery and Selection
JDBC	Java Database Connectivity
JMX	Java Management Extensions
JOnAS	Java Open Application Server
JVM	Java Virtual Machine
NMR	Normalized Message Router
OSGi	Open Services Gateway initiative *(deprecated)*
PaaS	Platform-as-a-Service
SaaS	Software-as-a-Service
SE	Service Engine
SU	Service Unit
SA	Service Assembly
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol *(deprecated)*
WSDL	Web Services Description Language
XML	eXtensible Markup Language
XSD	XML Schema Definition
GUI	Graphic User Interface
API	Application Program Interface
UI	User Interface
EIP	Enterprise Integration Patterns

2. Fundamentals

2.1. Cloud Computing

Our world is becoming more and more instrumented, interconnected and intelligent and pressures like workforce mobility and increasing productivity are placing greater demands on IT systems. To fulfil these challenges new paradigm called Cloud computing transforms the IT industry and changes way people work and companies operate. In particular, it enables the developers of software application to distribute their applications over the World Wide Web and to have reduced maintenance and installation efforts.

Cloud as a new evolving paradigm has on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured services essential characteristics. Cloud computing doesn't only provide software applications on the web, but also all computing resources, such as compute service, data store and networking. The definition of Cloud computing by National Institute of Standards and Technology (NIST) [NIS11] is "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models."

According to the NIST Cloud computing definition there are three following service models:

- *Software as a Service (SaaS)* Provider's applications are provided for consumers on the cloud infrastructure. Consumers can only use and configure this applications but they normally don't have access to manage and control underlying cloud infrastructure including, servers, operating systems, storage or even individual application capabilities.

- *Platform as a Service (PaaS)* This model provides to the consumer to deploy its applications onto the Cloud infrastructure. The consumer is not allowed to manage and control the underlying cloud infrastructures including, servers, operating systems, storage or even individual application capabilities but the consumer can configure the deployed applications and possible application hosting environment configurations.

- *Infrastructure as a Service (IaaS)* This enables the consumer to provide processing, storage, networks, and other fundamental resources. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications.

According to the NIST Cloud computing definition, the cloud applications are deployed differently to meet their requirements and priorities. The first deployment type *Private Cloud* that could be on-premise or off-premise cloud infrastructure that is only used by one organization. It may be owned, managed and operated by the organization, a third party, or some combination of them. The second one is *Community Cloud* that is used by a community of several organizations and it can be operated, managed, controlled by the organisations of the community or third party or some combinations of them. It exists on or off the premises of cloud the cloud provider. The third one is *Public Cloud* that is used by the general public and may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider. The last *Hybrid Cloud* is the combination of previously stated deployment models.

As it is clear from the IBM Cloud Reference Architecture 2.0 [OPG11] SOA services have all characteristics cloud services have but they are optional for SOA services. The Cloud computing architecture is based on SOA architecture, but it adds some additional features, like virtualization, security across business boundaries, and service management automation. Thus, development of cloud services are simply an extension of SOA services.

2.2. Service-Oriented Architecture

With increase of outsourcing of companies and the importance of business process reengineering have led to the emergence of Service-oriented Architecture (SOA) as an importance architectural style to the business information technology. Currently, SOA is widely used distributed architectural approach used by service providing and consuming enterprises. As a basis for SOA, a service is a logical representation of a reusable business activity, maybe composed out of other services, self-contained and hides the details behind it. Among many offered paradigms, SOA represents the easiest way to deploy and consume shared, reusable services [WCL+05].

SOA is an architectural style that supports service orientation. Service orientation is a way of thinking in terms of services and service-based development and the outcomes of services [SOAb]. SOA is a specific architectural style that is concerned with loose coupling and dynamic binding of services. The most important principles of SOA is represented in SOA Triangle [WCL+05]. As shown in the SOA triangle, *bind/publish/find*. First, a service provided by a provider needs to be described, for example, abstract definition, binding possibilities are described. Second, the service is registered or published to Service Discovery facility to allow consumer to find the service. Third, the consumers who want to use services searches for the service that meet their requirements with the help of Discovery Facility and finds suitable services. Finally, the consumer can bind to the found service and execute it.

SOA is implemented by Web Service Technology that has been selected as a leading SOA technology among other technologies, whereas ESB is the centrepiece of this implementation. The Figure 2.1 illustrates the high-level architecture of SOA stack. The bottom transport layer deals with transport protocols that connect to services and its requesters. Second bottom level,

which is on top of transport level, deals with XML and non-XML messages that are sent between requesters and services. The level on top of messaging layer deals with the description of Services and Interfaces in terms of operations and supported bindings with Web Services Description Language (WSDL) [WSD01] documents as well allows specification of policies to them to guarantee quality of services. The quality of service that the BUS supports are located in this level. In this level, security of Web services such as integrity, confidentiality, and non-repudiation as well as reliable transferring of messages and various kind of transactions support locate. The top layer represents various kind of virtual components of the Web services. In this layer atomic services represent services that are only the virtual component of a single service as far as requester's experience with the service is concerned. The services that composed of several services represent the composed services aimed to a certain business logic. Composed services that the service bus supports itself are choreographies and societies of services that cooperate with each other based on an agreement protocol to decide on the success of the cooperation at the end of the cooperation.

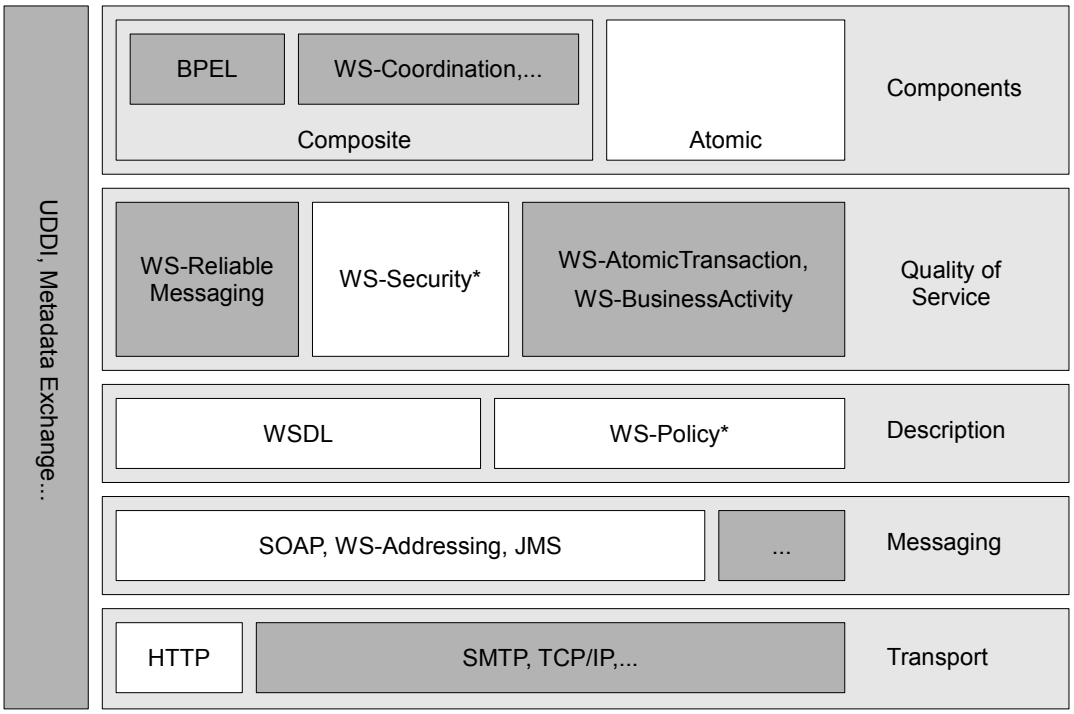

Figure 2.1.: SOA Stack

The structure of SOAP (version 1.2) [SOA07] messages is defined as an XML Information Set that support different transport protocols. Each SOAP message has a SOAP envelope in it that contains a SOAP header and a SOAP body. The message payload, which is in SOAP body, is defined with meta-data in a SOAP header. As a result, a SOAP message is processed accordingly on chain of nodes, with each chain node manipulating the message based on the

information in the SOAP header.

With the help of a WSDL (version 1.1) [WSD01] document an abstract and a concrete definition of a Web service can be specified. The abstract definition includes one or more port types that are the abstract interface of the service, whereas the concrete definition is used to map port types to concrete bindings and service endpoints. The WS-Policy [WSP07b] documents can be referenced in a WSDL document that define claims about QoSs or information contained in SOAP messages.

To sum up, Web Service Technology for SOA provides features to find services and their description as well as an agreement for the interaction between a requester and a service. Currently, SOA is the very important architectural paradigm that enhances efficiency, agility, and productivity of the enterprises where an enterprise solution logic is based on services.

2.3. Enterprise Service Bus

Several business and technical factors have led to the need for integration. Some organizations have generations of systems, legacy systems and mainframes, databases, enterprise applications and increasingly cloud and Software-as-a-Service (SaaS) applications and mobile applications. Achieving business objective requires integrating such applications. Over several years, a lot of work has been done for the integration of heterogeneous applications. First, integration was done manually, then Enterprise Application Integration (EAI) was employed. Finally with the emergence of SOA the ESB [Cha04] has been developed that is the middleware that enables communications among web-based applications. Traditional EAI supports methods and tools for applications systems focusing on EAI Patterns [HW03] and process-oriented integration. Whereas, SOA is a middleware platform focusing on messaging and transformation. Currently, EAI is provided as a service in an ESB.

According to Chappel [Cha04] "An Enterprise Service Bus (ESB) is a flexible connectivity infrastructure for integrating applications and services. An ESB can power your SOA by reducing the number, size, and complexity of interfaces between those applications and services". Key properties of ESB are different message-oriented middleware (e.g. synchronous/asynchronous, publish/subscribe, point-to-point), intelligent routing methods (e.g. content-based, rule-/policy-based, static routing, correlation), protocol transformation between requester and service, message transformation, secure and reliable messaging properties. It also allows service orchestration by compsing services available for the purpose of achieving a certain business logic. With the properties described, an ESB provides loosely-coupled communication between applications that allows applications changes take place independently without focusing on the integration of different applications. Additionally, an ESB allows you to add or delete services, or change the existing services with little impact or no impact to the use of existing services.

ESB as a distributed middleware platform can span beyond a single enterprise or business infrastructure. Each instance of an ESB can be distributed in different enterprises that is

connected each other. That means each instance can be independently used by a certain enterprise by integrating services of other enterprises on different locations.

Due to the importance of an ESB, it can be a building block to Platform-as-a-Service (PaaS) offerings in the Cloud. Therefore, we work in this master's thesis on the extension of the open-source ESB Apache ServiceMix for DSDS that is provided as a PaaS offering in the Cloud.

2.4. WS-Policy

WS-Policy [WSP07b] is an extensible framework that provides the means that are used to express the domain-specific description of capabilities, requirements, and other characteristics of Web Services-based system in policy documents. The capabilities, requirements, and other characteristics that manifest on the wire, for example authentication scheme, transport protocol and others are used for the selection of services by describing their privacy policies and QoS that do not have wire manifestation. Such properties can be expressed with simple declarative assertions and more sophisticated conditional assertions supplied by WS-Policy Framework.

The abstract policy model is the construct of *policy*, *policy alternative*, and *policy assertion* [WSP07b]. Policy assertion represents requirements, capabilities, and other characteristics of Web service-based systems. Policy alternative is potentially empty collection of policy assertions, that gives choice in other words. Policy is a potentially empty collection of policy alternatives. A policy with more that one alternatives indicate several choices, while a policy without alternative gives no choice.

A policy is used to convey conditions between service providers, service requesters, or Web service-based systems. Web service provider expose its policy under which conditions it provides its services. Service requesters might use these policies of the published services to choose the services that meet their needs. A Service requester chooses only one of the alternatives provided by service provider policy. As a scope of assertions is different for different properties (e.g. security, transaction, etc.) assertion authors provide domain-specific assertion language for a certain interaction scope. Therefore, a WS-Policy Assertion language that complies with WS-Policy framework is created. Created domain-specific policies are attached to the policy subjects (service, endpoint, operation, and message) [WSP07a].

As a building-block for policy documents, a policy expression has an XML Infoset representation that can either be in a normal form or compact form. *Normal form* of policy expression is key for interoperability and intersection that enumerates each of its alternatives that in turn enumerate each of its assertions. Whereas *compact form* of a policy is very verbose that expressing optional assertions, copying other policies, and not using nested policies. For detailed information please refer to WS-Policy framework specification [WSP07b].

Policy operators (*wsp:Policy, wsp:All and wsp:ExactlyOne*) are used in policy expression to convey requirements, capabilities, and other characteristics of entities that are collection of assertions. The schema outline for the *wsp:Policy* element in the compact form is as follows:

```
<wsp:Policy ... >
   ( <wsp:Policy ...>...</wsp:Policy> |
  <wsp:ExactlyOne>...</wsp:ExactlyOne> |
  <wsp:All>...</wsp:All> |
  <wsp:PolicyReference ... >...</wsp:PolicyReference> |
  ...
   )*
</wsp:Policy>
```

Intersection operation is carried out between the policies of two parties are to determine whether they have compatible alternatives. If parties agree on mutual alternatives, only then an interaction can take place. As we already mentioned, the policies in a normal form is needed for the intersection of policies. For that reason, there is also an important operation *normalization*. Normalization has to take place before intersection to normalize the policies of both parties for intersection.

Considering all capabilities of WS-Policy framework and comparing it with other policy languages, we have decided to utilize it as a base for our new WS-Policy Assertion language that will contain the assertions aimed to the description of functional and non-functional properties of CDHS in policy documents.

2.5. Extending an Open Source ESB for Multi-Tenancy Support Focusing on Administration and Management

In this section we describe the diploma thesis of Muhler [Muh11] shortly. In this diploma thesis, a multi-tenant management application system for the open-source ESB Apache ServiceMix [ASM] has been developed. The management application grants tenant users limited configuration access to the ESB's connectivity and integration services. The system enables data isolation between tenants for the management application and ESB message flows. Moreover, the management application can control clusters of ESB instances, supporting elasticity.

The management system is built on top of Apache ServiceMix [ASM] providing different Web Service Application Program Interface (API)s for its users. The provided Web service APIs are aimed to different roles of users like system administrator, tenant-administrator, and tenant-operator. System administrators are allowed to configure system clusters and grant cluster accesses to tenants and monitory the resource usage as well as assigns quotas of resources to the tenants. In contrast to system administrator, tenants consume the given quotas of resources to deploy service assemblies or services. Tenants also have different administrator and operator roles where tenant administrator is allowed to define roles and assign permissions to other tenants as well as create and assign contingents to the tenant operators. In turn, tenant operators use assigned contingents to them to deploy service assemblies or register services.

As the Management application is developed as a not part of the ESB, the information related to service assemblies, services and others should be stored in shared registries. For that purpose, service assembly and service data are stored in service registry database, whereas tenant related data is stored in tenant registry database. In addition to service and tenant registry databases, a new configuration registry database is created that stores cluster configuration data and enables access control to different roles as well as contingent assignment data. Each configuration data belonging to a certain tenant is isolated by having tenant identifier as primary key on its entities.

Integration of the Management web application to Apache ServiceMix is done by developing a new OSGi bundle that provides necessary management interfaces for Web application. The bundle consumes management messages containing management commands, such as install Java Business Integration (JBI) Component, install service assembly, coming from Management Web Application. Based on such received commands, service assemblies or services are deployed, where tenant identifier is added to their service endpoints enabling isolation of tenant services and components. Additionally, each message sent by tenants must also contain tenant context (see Sect. 2.6) that ensures data isolations between tenants.

In the current master's thesis, we develop DSDS support as well as Registry component with cache mechanism, which is in charge of accessing data from external databases that the described diploma thesis lacks.

2.6. Extending an Open Source ESB for Multi-Tenancy Support

Currently, Cloud computing is becoming more and more widespread and important in Information Technologies and most of the applications are being provided or migrated into the Cloud. Therefore, an ESB is also offered in a PaaS in the Cloud that is extended for multi-tenancy support in the master's thesis [Ess11]. In this work it is also investigated several open-source ESBs for the multi-tenancy awareness extension, eventually, the open-source ESB Apache ServiceMix is selected as the best candidate. Moreover, a new concept *tenant context* is conceived to develop multi-tenancy in the ESB.

Original implementation of Apache ServiceMix does not support multi-tenancy, therefore, its components need to be extended for multi-tenancy support. So called tenant context fragment attached included messages compromise tenant or tenant user related information to distinguish different tenants. To support multi-tenancy and provide tenant context, the data related to tenants and services need to be stored in a database. For this purpose, *Service Registry* and *Tenant Registry* databases are created, which store tenant isolated data. Service registry stores service related data, while tenant registry stores tenant related data.

Apache ServiceMix architecture modification is depicted in the Figure 2.2. It illustrates the main JBI components and highlights the changed/created components with bold border. These components are Binding Component (BC), such as SOAP over HTTP, JMS BC, and

E-Mail BC; Service Engine (SE), such as Content Enricher and Content Based Router that together provide Enterprise Integration Patterns (EIP) Tenant Router used to route the messages to the corresponding services/endpoints.

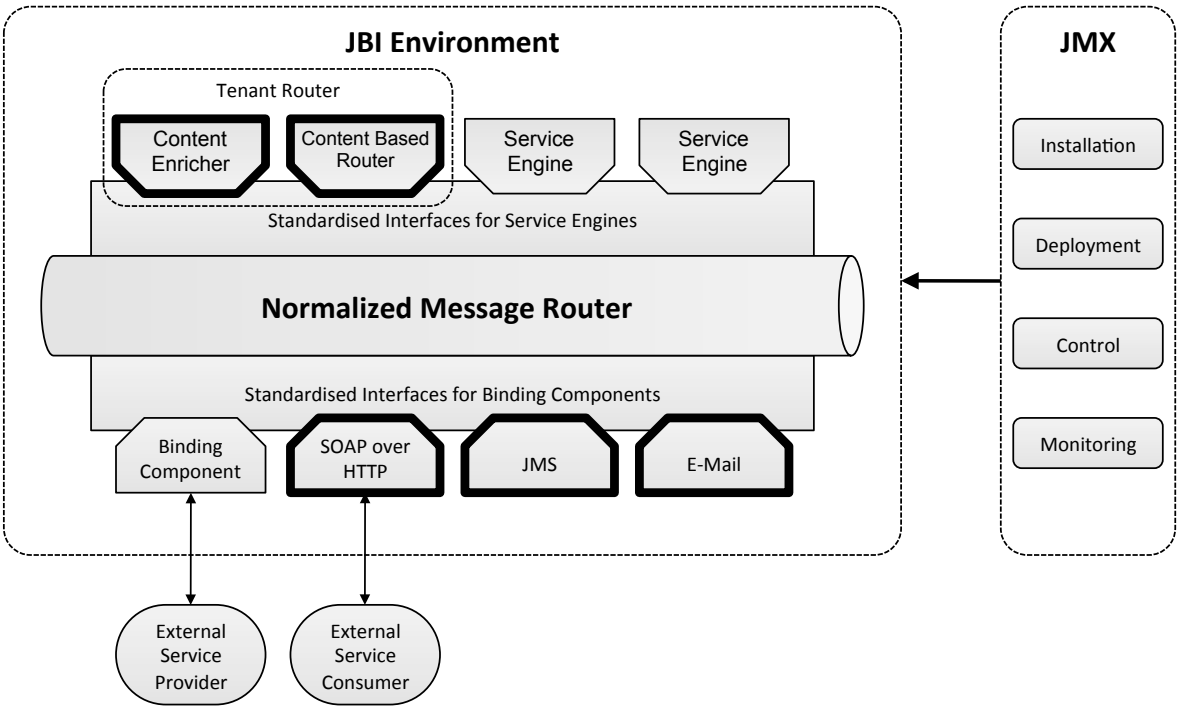

Figure 2.2.: Modifications to the Architecture of Apache ServiceMix

In the ESB, any provider, consumer, routing module or other component needs to be aware of a tenant when they process or serves tenant isolated messages. For that purpose, every tenant related message has to have a tenant context that uniquely identifies a tenant or tenant user the message belong to.

The BCs like servicemix-http, servicemix-jms, and servicemix-xmpp, which are in charge of connecting the ESB to outside world, are extended to support multi-tenancy by identifying tenants from a tenant context of a message. On the other hand, new service engines are created to support Tenant Router that consists of Message Enrichment and Message Router. As we already mentioned, Content Router and Content Based Router service engines together provide routing capability in the ESB, where Content Enricher adds the tenant specific data, which has actual service that handles the request of the specific tenant and tenant itself from service and tenant registries respectively to incoming messages. In contrast to content enricher, message router routes the message to the corresponding service/endpoint based on the data embedded in the message. The router routes the message with the help of Normalized Message Router (NMR), therefore, a message must be normalized before sending it to NMR and a tenant context must be added to each normalized message to allow messages' tenant isolation.

In this work the open-source ESB Apache ServiceMix is extended for multi-tenancy support in terms of communication. The main goal of this work is to make the open-source ESB Apache

ServiceMix multi-tenant aware by extending its BCs and creating new SEs for message routing, while keeping backward compatibility for non multi-tenant aware communication. However, in this work services are selected statically with defined routing rules rather than discovering and selecting/routing services dynamically by specifying a consumer's needs in messages.

2.7. Technologies

The following sections describe technologies that realize concepts of SOA and the ESB that this work builds up.

2.7.1. Java Business Integration

The JBI is the specification that defines a standards-based architecture. It can be used as the basis for Java-based integration products, in particular ESB, as the common characteristics of an ESB is abstract, which is earlier described (see Sect. 2.3). An ESB integrates different applications using diverse protocols, however, building blocks of the ESB itself are still vendor-specific. That means that each software vendor might design an own service container and define interfaces to connectivity and integration services. To provide common standards Java Community Process (JCP) created the JBI specification that standardizes the interoperating between service containers, connectivity services, and integration services [JBI05].

JBI model consists of JBI container that has SEs and BCs. The former hold services, while the latter is responsible for connectivity to existing applications and service outside the JBI environment. BCs allow JBI components to communicate over any protocol or transport like HTTP, JMS, and others, simply by plugging in the appropriate BC. BCs consumes messages and transfers them to NMR to be normalized that is then consumed by SEs. JBI components (SEs and BCs) connect with each other using the NMR, but NMR does not obligate the components with any format but is responsible for sending and receiving the messages, which enables to loose-coupling capability between components. NMR has several message exchange patterns like In-Only, Robust-In-Only, In-Out, and In-Optional-Out.

A JBI component can consume or provide services where provider provides a service, while consumer consumes the service. Providers describe services as per WSDL 1.1 [WSD01] or WSDL 2.0 [WSD07] specifications and declare them to the NMR. That means that each provide should publish its service description to the NMR as WSDL 1.1 or WSDL 2.0 document. As WSDL 2.0 document is split into an abstract and concrete part, the only abstract part of the WSDL is used inside the NMR. As a result, one needs to know abstract part to determine how to invoke a certain service specified in WSDL 2.0.

According to the Chappel [Cha04] JBI components can be themselves containers to Service Unit (SU)s can be deployed. The reason why deploying such artefact is that often integration problems cannot be solved by a single JBI component. SUs provide information about the services and their endpoints to the component. SUs are packaged and deployed on to target

components by Service Assembly (SA)s. SA is an archive file also has metadata file for the deployment.

The JBI also specifies a management framework that integration architects can use for configuring those services to cope with individual integration tasks. For this purpose, the JBI specification depends on Java Management Extensions (JMX) that is a standart Java means of managing and monitoring applications. To allow monitoring and management with JMX, each JBI component has to implement predefined Managed Bean (MBean) interfaces. By doing that administrators will be able to install or unistall such JBI components.

In this work we developed Dynamic Service Discovery and Selection functionality in a separate SE (see Sect. 7.1.1).

2.7.2. OSGi Framework

The OSGi framework [OSG11] is a module system and service platform for the Java programming language that implements a complete and dynamic component model. It runs modular applications and encourages resource sharing between component within a single Java Virtual Machine (JVM). There are executable and non-executable modules bundles in OSGi, which are packaged as Java Archive (JAR) files.

As a component either provides capabilities for other components or depends on other components, each OSGi bundle has meta-data that describes capabilities it provides and requirements it demands. Capabilities are Java packages the bundle provides of Java classes it provides, whereas requirements are Java packages the bundle depends and does not deliver itself. After a new bundle deployment, the OSGi framework initializes a network of Java class loaders, allowing bundles to use system packages, framework packages, and packages exported by other bundles.

As bundles should be administered, the OSGi framework defines the bundle lifecycle operations (install, update, start, stop, and uninstall). Therefore, executable bundles implement OSGi specific interfaces that allow the framework to start and stop the individual bundle. Moreover, a bundle can provide services to other bundles by registering service objects, of which methods can be access by direct invocation. OSGi services are just Java interfaces representing a conceptual contract between service providers and service clients. The services publishes themselves in the service registry, where service other bundles can look up use necessary services.

As a part of this work, we developed a OSGi bundle that is used to get the data from the databases outside Apache Servicemix and caches them if necessary (see Sect. 7.1.3).

2.7.3. Apache ServiceMix

In the current work, we are going to extend the open-source ESB Apache ServiceMix for Dynamic Service Discovery and Selection, which we call ServiceMix from now on. Apache ServiceMix is a flexible, open-source integration container that combines the features and

functionality of Apache ActiveMQ [AMQ], Camel [APA11a], ODE [AOD], Karaf [APA11b] into a powerful runtime platform one can use to build his/her own integration solutions. It is powered by OSGi (see Sect. 2.7.2).

The main features Apache ServiceMix are as follows:

- reliable messaging that is provided by Apache ActiveMQ
- messaging, routing and Enterprise Integration Patterns with Apache Camel
- WS-* and RESTful web services that is provided by Apache CXF
- loosely coupled integration between all the other components with Apache ServiceMix NMR including rich Event, Messaging and Audit API
- complete WS-BPEL engine is provided by Apache ODE
- OSGi-based server runtime powered by Apache Karaf that builds its kernel layer

The goal of ServiceMix is to provide ESB that implements JBI specification. The OSGi framework realizes the technology layer of Apache ServiceMix on top of the kernel layer, which complies with JBI specification. Moreover, ServiceMix includes large set of JBI Components that together provide ESB functionalities and features given. There two types of JBI components, BC and SE. The former support different protocols like HTTP, JMS, FTP, and SMTP. The latter is used to orchestrate services in ServiceMix by consuming the messages handed-off by BCs. SEs also wraps Apache Camel provides enterprise integration patterns [HW03], which includes XML transformations, content-based routing, message splitting, or message aggregation. The NMR is based on Apache ActiveMQ in ServiceMix.

An instance of ServiceMix can be administered via a command line console that is provided by Apache Karaf. The console not only provides management for OSGi bundles and services, but also introduce additional console commands for managing installed JBI components and deployed service assemblies. One can ServiecMix artifacts like OSGi bundles, JBI components, or SA putting them in deploy directory of installed ServiceMix. For the extension of ServiceMix, the developers are supplied with Maven [AMV] plugins that eases the creation of ServiceMix artefacts.

3. Related Works

3.1. Comparison of Policy Languages for the Usage in Cloud Computing

In the Student Report Software Engineering [SR12], number of already existing Policy Languages are observed for the usage in Cloud Computing. The policy languages for Cloud Computing are evaluated for the projects CloudCycle [CLC12] and MIGRATE [MIG12] based on the criterias of policy languages. TOSCA [OAS12] standard of OASIS Consortium is used as a foundation for both projects.

There are number of policy Languages to be analysed to find which ones are more suitable for Cloud Computing. As a result, some policy languages are excluded due to the lack of such features like unsufficient documentation, no XML support, language redundancy, and others. The policy languages that do not lack such features are observed further.

Analytic Hierarchy Process (AHP) [SR12] is employed to support decision making in big groups and to simplify the problem policy language decision making. There are three main criterias: User-Friendliness, Power (Strength) and Technical prerequisites which are used in AHP to rank the policy languages. These main criterias are subdivided into sub-criterias as well:

- User Friendly - Readability, Compactness, Documentation, Intuitiveness, Modularity, Comment Support;
- Power - Optional Requirements, Logical Operations, Set Operations, Dependability, Specification and Generalization, Expandability, Formal Model;
- Technical Prerequisites - Implementation, Programming Language Support, Tool Support;

Each criteria has "yes" or "no" value. Based on these criterias the decision making of the policy languages is accomplished by using the procedure shown in the student project.

According to the student project, WS-Policy [WSP07b] and Rei [Rei] are ranked as the best suitable policy languages for Cloud Computing as shown below. First one, WS-Policy is a policy language, W3C recommendation as of 2007, provides service providers ability to specify quality, version and security properties and evaluate them against consumer policies. Another one is the policy language Rei which is based on the latest standards of the semantic web. Rei was developed for very dynamic and distributed environments to give a semantic web an ability to define, use and evaluate policies without central management or predefined decision makings.

It is shown below first top six policy languages (taken from the student project) with their rank value and the reason why they are excluded:

1. WS-Policy 9,38
2. ~~Rei~~ 8,83 – is used for semantic web
3. ~~XACML~~ 8,12 – extensible XML schema to describe authorization and entitlement policies.
4. ~~PERMIS~~ 7,57 – PERMIS is developed to implement role-based access right in multi-user systems.
5. ~~SAML~~ 7,40 – standard for exchanging authentication and authorization data between security domains
6. ~~RuleML~~ 5,26 – has very weak extensibility

As a result of thoroughly learning the student project and investigating the policy languages available in the Web, we have come to decision to use WS-Policy language. As demonstrated in the student project the most of the policy languages don't fit CDHS property description because for example some of them for special purposes, some of them lack extensibility. Among them the WS-Policy language is selected as the most suitable language to express functional and non-functional properties of CDHS in policy documents.

3.2. WS-Policy and WSRF Extensions for Open Source Enterprise Service Bus

ProBus [Wie07] is the extension of the open-source ESB Apache Service Mix 3.3.1 providing the capability of Dynamic Service Discovery and Selection by employing WS-Policy and WSRF [OAS04]. With the help of ProBus, tenants (companies, enterprises) can only specify their requirements in a WS-Policy document then send a request the policy attached to it. The service which fulfils these requirements will be selected and bound automatically (or manually by tenants) by ProBus. There is no need to know for a tenant which service or resources being selected and bound, consequently, flexibility of the application composed of reusable services is increased and the providers have the possibility react faster on changes in the market.

The WS-Policy and WSRF are extended in ProBus for the purpose of enabling requests to select services with the help of policies extended to support specific requirements. The policies are attached to Policy Subjects by using XPointers [XPF03]. Service consumers can be located either inside or outside JBI environment. In order to determine whether the requirements of provider policy meet the requirements of consumer, the policies must be intersected. If the result of intersection gives at least one alternative (i.e. a provider meets a consumer requirement) then it has to be checked for rules compatibility defined by XPath [XPA99] expressions.

WSRF is also extended to enable ESB to process Policies related to resources and stateful Web Services. For that new assertions types are defined and formulated with help of XPath expressions.

Under two circumstances the post-processing is carried out. First one is aimed to assess the alternative(s) resulting from intersection if they match the rules defined. Second one is that the endpoint resources are evaluated against the policy requirements. In both cases the post-processing is carried out based on the rules defined with XPath expressions.

ProBus employs Apache WS-Commons Policy Framework [WSP07b] that provides general framework for the programmers to use WS-Policy. The operations that play important role such as merge, normalize and intersect to find corresponding services based on WS-Policies are built-in. The framework is integrated into Normalized Message Router (NMR) (see Fig. 3.1) as depicted in Figure 3.1.

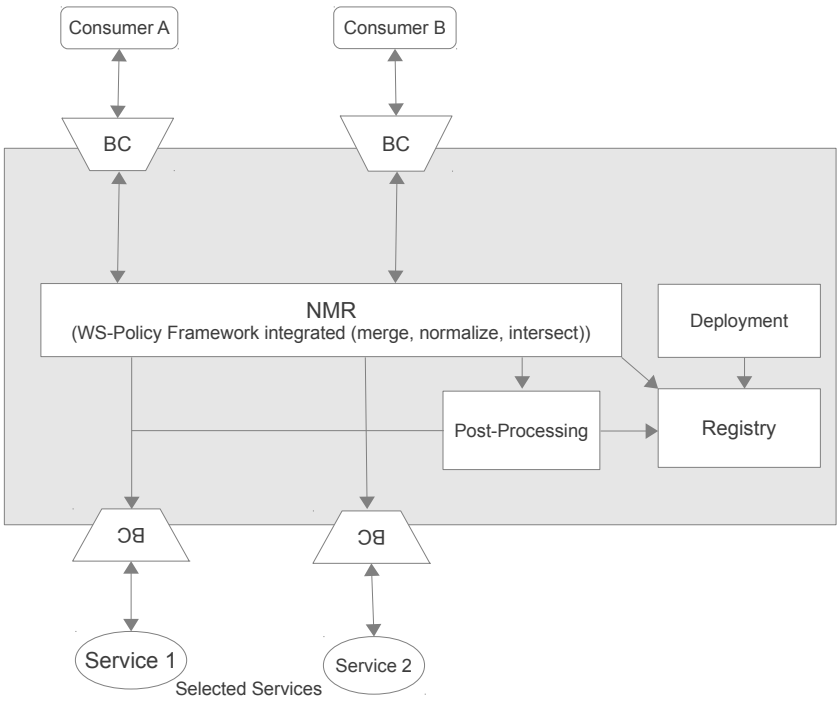

Figure 3.1.: ProBus Architecture.

In contrast to this diploma thesis, we integrate a similar WS-Policy framework and post-processing into a single JBI Component (SE) inside ServiceMix. Developing entire DSDS functionality in a single JBI Component allows us to deploy the component to other ESBs that support JBI. Furthermore, it makes backward compatibility support easier to configure by just ignoring DSDS JBI Component.

3.3. A User Driven Policy Selection Model

In the technical report [MF06], service selection based on non-functional properties of Web services are conceived. It is firstly created quality model for non-functional properties and then service selection model according to quality model is developed. It is also considered preferences of users that lead to choosing the best service by ranking services base on the user preferences.

The quality of Web service can be defined as the set of quality dimensions (qds) that are used to describe non-functional aspects of a Web service. Because of strong dependency on quality dimension, it is included so called *quality designer* who collects and organizes quality dimensions in a certain application domain. Quality designer creates so called QT (Quality Tree), which is a tree of all quality dimensions. For further details about quality model, which is not that related to our work, the work [MF06] can be referred.

Policy model that is used to requirements or capabilities of a Web service. Service provider attaches *service policy* to its service describing its offerings, whereas requesters specify their requirements in a *user policy*. WS Policy language is utilized to specify the service and user policies. Service policy includes only *pqds* (primitive quality dimensions) and it is defined when the service provider publishes the Web service. It is assumed that QT(Quality Tree) to the Web service being advertised exists. By using this QT service provider sets values of primitive quality dimensions. Likewise, requester also fills the values to the primitive quality dimension and derived quality dimension (dqd) of its user policy. The requester can also specify if all its requirements have to be met completely by service provider or partly. For this purpose, user can indicate quality dimensions as *mandatory* that must be met by service provider. In addition, requester can change quality dimension weights according to its own preferences. For example, some user may prefer to have cheaper price than having high quality. In this case, the requester puts more weight to price quality dimension.

Given user policy and several Web services with their QT, policy selection is used to discover the best service candidate that meet requester preferences best. It is assumed that service providers and requesters have the same QT model. The requester has UP(QT) (user policy) and customized quality tree (UQT), whereas service providers have SP(QT)s (service policies).

The selection process is composed of two main processes: *satisfiability evaluation* and *alternative ranking*. During satisfiability evaluation process it is verified that for all the requirements expressed in UP, there exists at least one of the service capabilities. This means that all the quality dimensions specified in UP must be included in SP too. If at least one quality dimension of UP is not offered by SP, in case all user quality dimensions indicated as mandatory, then the satisfiability evaluation fails. The UP quality dimensions indicated as non-mandatory may be ignored, that is service is not obliged to provide non-mandatory user requirements. The result of satisfiability evaluation process is matched policy (MP).

Alternative ranking process is responsible for the ranking of the alternatives in the MP resulted from satisfiability evaluation according to the preferences specified in the UQT. Where, input

policies are MP and UQT (customized quality tree for user preferences) and the result of the ranking is ranked policy (RP).

In this work a new approach to select Web services is proposed as a result of anaylysis of offered quality. Quality definition model is defined that is used to specify Web service provider quality and user desired quality. As opposed to our work, user preferences are specified by changing QT as well as they used mandatory and non-mandatory fields for quality dimensions.

3.4. Dynamic Service Selection Capability for Load Balancing in ESB

In the work Aimrudee [AJ10] Dynamic Service Selection Capability for Load Balancing in ESB is developed. Load-balancing is done between different services with the same function, not between service replicas. New *service type* concept is created that is used to group services with the same functionalities. This enables dynamic selection of the target service. The selection of the service from the service type is based on a balancing strategy.

Service type groups several services with the same functionalities. Each service in the service registry belongs to a service type. A service type is formed of *service type name*, *service signature*, and *service property*. The basic idea of incorporating the service type is a service of the same service type may be substituted with each other.

The idea of balancing mechanism is that a service is dynamically selected based on a service type. Balancing mechanism functionalities are splitted into several components. First one is *inbound router* that is provided by Mule [Mul] receives a request message. Another one is *message extractor* that is used to extract the service type from the message in this work. Third one is *service group recognizer* that receives a service type and finds services having this service type from the registry. It also can filter services if service property information is specified. Fourth one is *service registry* that stores the information including service type of services. Fifth one is *balance computing module* that is the main component responsible for sending the messages to the most suitable service, which has the lowest load. There is also *load info* that keeps load information of a service. *Load monitor* selects a service based on the data in a load info of a service and sends a message to *extended outbound router*. The extended outbound router, in turn, is used to send messages to a target service. It also handles failure situations such as a service is unreachable.

The selection of a actual destination service is based on balancing strategy. Several strategies such as round-robin, random, threshold, minimum, and last-load that are implemented in the work of Aimrudee.

In contrast our work, this work only considers load-balancing to select services dynamically but it does not select services based on their functional and non-functional properties (specified with WS-Policy) nor does it support multi-tenancy.

4. Domain-specific WS-Policy Extension for CDHS

In this chapter a new domain-specific policy assertion language for CDHS [SKLU11] will be introduced. The language provides assertions through which functional and non-functional properties of CDHS can be described. These assertions are listed in detail.

4.1. Introduction

The specification defines the new domain-specific policy assertion language for CDHS that allow the specification of capabilities, requirements and other characteristics of CDHS. With the help of the assertion language, functional and non-functional properties of CDHS can be specified in WS-Policy alternatives.

4.1.1. Goals

With the current extension to WS-Policy Language we propose a uniform language and format to specify functional and non-functional properties of CDHS.

4.1.2. Requirements

In meeting the goal the specification must address the following requirements:

The Means

Define the means to describe functional and non-functional properties of CDHS.

Extensibility

It must be possible to define new elements and new assertion construction.

Re-usability

The assertions created with this language should be re-usable.

Understandability

The policy language has to be understandable to the users in order to make creation and extension of policies easier.

4.2. CDHS Policy Assertions

In this section we list and describe all assertions in detail that constitute our current WS-Policy Assertion Language.

4.2.1. Scalability

Scalability assertion (see lines 9-37 in Listing A.1) of a CDHS is its ability to endure increasing workloads without decreasing an agreed service level when underlying resources are also increased. It identifies an assertion (<cdhsp:calability ...> ..</cdhsp:scalability>) in the policy schema. Scalability has also several sub-properties that are described below:

Automation Degree

This assertion indicates to which extent scalability is automated.

- **Manual** – this indicates that scalability is done manually by humans.
- **Automatic** – this indicates that scalability adjust automatically depending on the changes in a CDHS (e.g. load-balancin, performance, etc.).

Type

This assertion identifies how servers of a CDHS are scaled. Servers of a CDHS can both scale vertically (up/down) and horizontally (in/out).

- **Vertical** – This indicates that vertical scalability is used in a CDHS. Typically vertical scaling refers to adding more hardware to a system to improve processing capability, load-balancing.
- **Horizontal** – This indicates that horizontal scalability, also described scale-out, is used in a CDHS. Typically horizontal scaling refers to tying multiple independent computers/servers together to provide more processing power.

Degree

This assertion indicates if the number of (virtual) replicas is limited or not.

- **Virtually Unlimited** – this indicates that number of virtual replicas is not limited.
- **Limited** – this indicates that number of replicas is limited.

Time to Launch New Instance

This nested assertion indicates how much time is needed launch new instance of replica for scalability of a CDHS. The time (in milliseconds) that is needed launch new instance of replica for scalability of a CDHS is set with attribute *Milliseconds*.

Automatic Scalability Criterion

This nested assertion indicates the criteria of automatic scalability, which might be either system load or latency, or combination of both.

- **System Load** – this indicates that system load is main criteria for automatic scalability.
- **Latency** – this indicates that latency is main criteria for automatic scalability.

Transfer Limit

This nested assertion defines the maximum information volume being transferred to/from CDHS. The information can either incoming (in) or outgoing (out).

- **In** – This assertion defines the maximum information volume being transferred to CDHS. *Gigabyte* – This attribute is used to set the maximum information volume being transferred to CDHS.
- **Out** – This assertion defines the maximum information volume being transferred from CDHS. *Gigabyte* – This attribute is used to set the maximum information volume being transferred from CDHS.

4.2.2. Availability

This assertion (see lines 40-72 in Listing A.1) is used to describe the availability properties of CDHS. System called available if it is up and running and produces correct results. The availability of a system is the fraction of time it is available. Availability also includes several (nested) properties, which are *replication*, *replication type*, *replication method*, *replication location*, *self-healing*, *automatic fail-over*, and *degree*.

Replication

This identifies replication assertion. This assertion indicates that replication capability is supported.

Replication Type

This assertion indicates that Master-Master, Master-Slave, or combination of both replication types (method) are used in a CDHS. *Master-Master* This assertion indicates that Master-Master replication type (method) is used in a CDHS. *Master-Slave* This assertion indicates that Master-Slave replication type (method) is used in a CDHS.

Replication Method

This assertion identifies the replica connection/process methods such as synchronous, asynchronous, combination of both. *Synchronous* – This assertion indicates that the replica is connected/processed synchronously. *Asynchronous* – This assertion indicates that the replica is connected/processed asynchronously.

Replication Location

This assertion identifies replica locations.

- **Same DC** – This assertion indicates that a replica is located in the same data center with a CDHS.
- **Diff DC in Same Region** – This assertion indicates that a replica is located in different data center in the same region with a CDHS.
- **Diff DC in Diff Region** – This assertion indicates that a replica is located in different data center in different region with a CDHS.

Automatic Fail-Over

This assertion indicates whether automatic fail-over is available or not. Automatic fail-over means that in case of failure circumstances, the system automatically and dynamically avoids failures by using other replicas for example.

Availability Degree

This assertion indicates availability degree ranging from 0 to 100%. *MinRange* – This attribute is used to set minimum range of availability degree. *MaxRange* – This attribute is used to set maximum range of availability degree.

4.2.3. Recovery

This assertion (see lines 75-80 in Listing A.1) defines in which kind of failure situations the CDHS instance can be recovered.

- *Disaster Recovery* – a CDHS has a disaster recovery capability. Hence, it can adjust servers against disasters.

4.2.4. Security

This assertion (see lines 83-94 in Listing A.1) includes several sub-assertions as follows:

- **Storage Encryption** – this indicates whether the storage encryption is available or not.
- **Transfer Encryption** – this indicates whether transfer encryption for each data being transferred is available or not.
- **Firewall** – this indicates whether firewall for a CDHS is available or not. It is used to secure network access to the CDHS. CDHS may provide firewall settings to control the network access.
- **Authentication** – this indicates whether authentication for a CDHS is available or not. Authentication is the process of identifying someone or something is, in fact, who or what is declared to be. In different systems authentication is done with help of logon passwords and digital certificates.
- **Authorization** – this indicates whether authorization for a CDHS is available or not. CDHS users are given access rights as authorization is the process of giving someone permission to do or have something (e.g. HR group is authorised to access employee records).
- **Confidentiality** – this indicates whether confidentiality property of a CDHS is available or not. It is the term used to mean preventing the disclosure of information to unauthorized individuals or systems (credit card data, personal data, medical data, etc. should be protected).
- **Integrity** – this assertion indicates whether integrity of data in a CDHS is provided or not. The CDHS provide data integrity that prevents from undetected data modification.

4.2.5. Privacy

This assertion (see lines 97-112 in Listing A.1) includes privacy properties.

Secure Destruction

This assertion indicates that whether the CDHS provider destroy Personally identifiable information (PII) obtained by customers in a secure manner.

Complete Destruction

This assertion indicates that whether the information completely destroyed and can it be recovered.

Supported Privacy Policy Languages

- **p3p** [P3P] – This assertion indicates that P3P language is supported. The Platform for Privacy Preferences. P3P is a standard for communicating privacy practices and comparing them to the preferences of individuals.
- **xacml** [XAC] – This assertion indicates that XACML language is supported. The Extensible Access Control Markup Language together with its Privacy Profile is a standard for expressing privacy policies in a machine-readable language which a software system can use to enforce the policy in enterprise IT systems.
- **epal** [EPA] – This assertion indicates that EPAL language is supported. The Enterprise Privacy Authorization Language is very similar to XACML, but is not yet a standard.
- **wsPrivacy** – This assertion indicates that WS Privacy language is supported. "Web Service Privacy" will be a specification for communicating privacy policy in web services.

4.2.6. Location

This assertion (see lines 115-138 in Listing A.1) includes Cloud and geographic locations.

Choice

If choice is enabled then locations can be chosen. Otherwise, the location might be either single location or selected automatically.

Geographic Location

This assertion indicates the locations of a CDHS servers. The name of the locations can be set to the attribute *name* of the assertion.

Cloud Location

- **On-Premise** – this indicates that the Cloud infrastructure of a CDHS is located inside an organization.
- **Off-Premise** – this indicates that the Cloud infrastructure of a CDHS is located outside an organization.

4.2.7. Data Constraints

Amount of data stored in the CDHS may be limited in terms of the following (see lines 141-159 in Listing A.1):

- **MaxItemSize** – this indicates the maximum size in bytes of an item stored in a Cloud store, the size is set with *Bytes* attribute that in turn accepts integer values
- **RowSize** – this indicates the maximum size of each row stored in bytes in a CDHS, the size is set with *Bytes* attribute that in turn accepts integer values
- **FileSize** – this indicates the maximum size of each file stored in bytes in a CDHS, the size is set with *Bytes* attribute that in turn accepts integer values

And in terms of

- **MaxDomainSize** – this indicates the maximum size of each domain in gigabytes stored in a CDHS, the size is set with *Gigabyte* attribute that in turn accepts float values
- **TableSize** – this indicates the maximum size of each table in gigabytes stored in a CDHS, the size is set with *Gigabyte* attribute that in turn accepts float values
- **BucketSize** – this indicates the maximum size of each bucket in gigabytes stored in a CDHS, the size is set with *Gigabyte* attribute that in turn accepts float values

And in terms of

- **MaxItemNumberPerInstance** – this indicates the maximum number of items per CDHS instance, the number of items is set with *Number* attribute that in turn accepts integer values
- **MaxRowNumberPerInstance** – this indicates the maximum number of rows per CDHS instance, the number of rows is set with *Number* attribute that in turn accepts integer values
- **MaxFileNumberPerInstance** – this indicates the maximum number of files per CDHS instance, the number of files is set with *Number* attribute that in turn accepts integer values

And in terms of

- **MaxSizePerInstance** – maximum overall size of all domains, tables, and buckets per a CDHS inistance

- **PredefinedInstanceSize** – the size of a CDHS can be predefined if this property is enabled. And the following property means that Cloud computing/storage/database/etc. instance size is predefined.

4.2.8. Interoperability

This assertion includes all interoperability properties (see lines 162-254 in Listing A.1).

Data Portability

The CDHS consumers own their data and its platforms should make it easy and efficient to securely, consistently move customers data in and out. To indicate such properties as assertions the following portability capabilities are used:

- **import** – enables import of data from other Cloud providers and/or
- **export** – enables export of data from the current CDHS and/or
- **One-Way Synchronisation** – Data is expected to change in one location only. To reconcile the changes, the synchronization process copies changes in one direction only. For example, if data in consumer local servers is synchronised based on the data on CDHS but data on CDHS is not synchronised based on the data on consumer servers.
- **Two-Way Synchronisation** – the synchronization process copies data on both directions to reconcile changes as needed. Data is expected to change on both locations. Locations can be locations of Cloud consumer and Cloud providers or two Cloud providers. And/or none

Data Exchange Format

This assertion indicates the data exchange formats supported in the exchange with the other CDHS. The sub-assertions that constitute the this assertions are as follows:

- **XML** – this assertion indicates whether XML data exchange format is supported or not
- **JSON** – this assertion indicates whether JSON data exchange format is supported or not

Additionally, other proprietary or no proprietary data exchange formats can be added as the language allows extensibility.

Storage Access

This assertion indicates how the data of the CDHS is accessed. The variants of the data accesses of the CDHS are as follows:

- **SOA** – this assertion indicates whether services to access the data of the CDHS is accessed with SOA style web services or not
- **REST** – this assertion indicates whether services to access the data of the CDHS is accessed with RESTful style web services or not
- **SQL** – this assertion indicates whether the CDHS is accessed as plain SQL requests or not

ORM

This assertion indicates which Object Relational Mappers (ORM) are supported by the Cloud provider or community. The variants of ORMs are as follows:

- **JPA** – Java Persistence API
- **JDO** – Java Data Objects
- **LINQ** – LINQ
- and other Object Relational Mappings

Migration and Deployment Support

This assertion indicates whether the data migration and setup are supported by the Cloud provider.

Supported Integrated Development Environments (IDE)

This assertion indicates which developer Integrated Development Environments are supported by the Cloud provider or community. The variants of IDEs are as follows:

- **Eclipse** Eclipse IDE is supported
- **NetBeans** NetBeans IDE is supported
- **Visual Studio** VisualStudio IDE is supported
- **IntelliJIdea** IntelliJIdea IDE is supported and other IDEs are supported

Developer SDKs

This assertion indicates which developer Software Development Kids (SDK) are supported by the CDHS provider. The variants of SDKs are follows:

- **Java**
- **.Net**
- **PHP**
- **Ruby**
- **Python**
- or none of them

OS

This assertion indicates which Operation Systems are supported by the Cloud provider or community. The operations are Windows Versions, Linux Distributions, and the others.

4.2.9. Compatibility

Compatibility of the CDHS with the database products including their versions (see lines 257-270 in Listing A.1).

- **Product including Version** the CDHS can be compatible with MySQL, Postgre SQL, MSSQL, etc. versions

4.2.10. Storage

Storage assertion (see lines 273-294 in Listing A.1) is used to indicate how storage is accessed, what storage type is used as well as how transaction is supported in the CDHS.

Accessability

How CDHS storage is accessed via VPN or the other ways of accesses. Its sub-assertion VPN indicates whether it is possible to restrict access to the CDHS to a Virtual Private Network (VPN).

Storage Type

This assertion indicates the type of the CDHS. Its sub-assertions, RDBMS stands for Relational Database Management System, NoSQL stands for Not only SQL, Blob Store is for storing binary data and CDN stands for Content Delivery Network.

Transaction Support

This assertion indicates how transactions are supported. Is the Cloud data ACID (Atomicity, Consistency, Integrity, Durable) compliant?

4.2.11. Performance

This assertion (see lines 298-310 in Listing A.1) is composed of the key performance indicators of the CDHS.

Predictability Read/Write/Response

This assertion indicates whether the response time for read and write requests predictable.

Throughput

This assertion indicates how much throughput can be used at most in the CDHS. Throughput as an actual transfer volume is measured byte/seconds.

Latency

This assertion indicates how much is latency of a request to the CDHS. Latency is a measure of time delay experienced in a system. 0 - N milliseconds.

4.2.12. CAP

This assertion (see lines 313-327 in Listing A.1) is used to define CAP [Bre12] properties.

Consistency Model

This assertion indicates that which assertion consistency models are supported by the CDHS. The existing consistency models are as follows:

- **Strong Consistency** – After the update completes, any subsequent access will return the updated value.
- **Weak Consistency** – The system does not guarantee that subsequent accesses will return the updated value.
- **Eventual Consistency** – The storage system guarantees that if no new updates are made to the object, eventually all accesses will return the last updated value.

Availability in Case of Partitioning

This indicates that availability is kept although distributed databases are partitioned.

4.2.13. Flexibility

This assertion (see lines 330-336 in Listing A.1) defines flexibility of the schema of the data in a CDHS.

Schema

This assertion indicates whether a schema can be created for the data in the CDHS.

Schema Customizable

This assertion indicates whether the schema of the data in the CDHS can be customized or not.

4.2.14. Cloud Computing

This assertion includes (see lines 339-356 in Listing A.1) nested assertions that altogether define Cloud computing properties.

Service Model

This assertion indicates which Cloud service model the CDHS use. The service models variants are as following:

- **IaaS** – Provider's applications are provided for consumers on the Cloud infrastructure. Consumers can only use and configure this applications but they normally do not have access to manage and control underlying Cloud infrastructure including, servers, operating systems, storage or even individual application capabilities.
- **PaaS** – This model provides to the consumer to deploy its applications onto the Cloud infrastructure. The consumer is not allowed to manage and control the underlying Cloud infrastructures including, servers, operating systems, storage or even individual application capabilities but the consumer can configure the deployed applications and possible application hosting environment configurations.
- **SaaS** – This enables the consumer to provide processing, storage, networks, and other fundamental resources. The consumer does not manage or control the underlying Cloud infrastructure but has control over operating systems, storage, and deployed applications.

Deployment Model

This property contains Cloud deployment models that the CDHS provide.

- **Private Cloud** – is either on-premise or off-premise Cloud infrastructure that is only used by one organization. It may be owned, managed and operated by the organization, a third party, or some combination of them.
- **Public Cloud** – that is used by the general public and may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the Cloud provider.
- **Community Cloud** – that is used by a community of several organizations and it can be operated, managed, controlled by the organisations of the community or third party or some combinations of them. It exists on or off the premises of the Cloud provider.
- **Hybrid Cloud** – the combination of Private, Public, Community Cloud deployment models.

4.2.15. Management / Maintenance Effort

This assertion (see lines 359-365 in Listing A.1) indicates the properties used to define how efforts to management and maintenance are done.

Degree of Automation

How are updates of the CDHS managed? Are they installed automatically or is it a manual process (self-service)?

4.2.16. Monitoring

This assertion (see lines 368-381 in Listing A.1) indicates the Key Performance Indicators (KPIs) that are provided to monitor the CDHS instances.

Supported KPIs

This property indicates that what Key Performance Indicators are available to monitor a CDHS. For example, *Processing Load (CPU)*, *Data Transfer - Network (I/O)*, *Data Transfer - Disk (I/O)*, *Memory Load (RAM)*.

4.2.17. Backup

This assertion (see lines 384-413 in Listing A.1) includes to indicate the backup properties.

Backup Interval

This assertion indicates how often can backups get triggered. For example, *none*, *periodic(every 1 - x mins)*, and/or *on-demand*.

Interrupt of Access

This property indicates that whether the creation of a backup shortly interrupt access to the CDHS.

Number of Backups Kept

This assertion indicates that how many and/or how long backups are kept by the provider of the CDHS. For example, None, 1 - x Backups, and how long 1 - x Days.

Backup Method

What backup methods are supported by the provider of the CDHS. There are three common backup methods:

- **Snapshot** - states of a CDHS are stored at certain point in times.
- **File-Backup** - CDHS instance is backed up as a file. For example, as a compressed zip file or in other file formats.
- **Incremental Backup** - this method is used to back the data up that have changed since last backup.

4.2.18. Multi-tenancy

Multi-tenancy refers to software architecture where a software instance serves to multiple software clients (tenants) (see lines 416-427 in Listing A.1).

Multi-tenancy Capability

This property indicates whether multi-tenancy capability is supported by CDHS or not.

Multi-tenancy Type

There are two types of multi-tenancy:

- **Multiple Instances** – a CDHS platform creates a new instance for each CDHS client (tenant)
- **Native Multi-tenancy** – CDHS provides multi-tenancy that is one instance of a CDHS serves to multiple consumers (tenants)

5. Concept and Specification

This chapter composes requirements for a Dynamic Service Discovery and Selection functionality extension for Apache ServiceMix [ASM] that can be part of a PaaS platform, like *4CaaSt* [4Ca]. At first, an overview of the key functional requirements and components is given in Section 5.1. The following sections describe functional requirements in more detail and include a use case analysis. Moreover, non-functional requirements are listed in Section 5.7 giving guidance values for several software qualities.

5.1. System Overview

In the current extension of the open-source ESB Apache ServiceMix, we develop DSDS functionality for CDHS based on the created new WS-Policy Assertion language (Chapter 4). Tenant operators (service providers) register their services with the policy attachment that contain the service capabilities. In turn, sends requests with their requirements in the policy attached (see Sect. 5.3). DSDS SE is aimed to find and select a suitable service for the request by matching the consumer and provider service policies (see Sect. 5.2). The registered services and their policies are retrieved from the service registry database through a component that employs caching mechanism (see Sect. 5.4). Furthermore, while extending the ServiceMix for the dynamic discovery and selection, the multi-tenancy support is maintained that allows only discover the tenant services for its tenant users (customers). DSDS can also run not in a multi-tenant manner that discovers services for the messages without tenant-context.

We made the changes to the Web User Interface (UI) and Web Service API of the system developed on top of the ServiceMix that is JBIMulti2. These changes will be described in more detail in the coming sections.

5.1.1. Components

As described in the Section 2.7.3, ServiceMix complies with the JBI specification [JBI05]. Therefore, ServiceMix is designed as a JBI container that allows the installation of two types of JBI components. BC connect the JBI container to external endpoints via different protocols, whereas SE orchestrate the message flow inside the JBI container. Each JBI component uses service assemblies as configuration data that defines additional behaviour, such as message flows, message transformations, or bindings to external endpoints. Dynamic service discovery and selection support is developed in a separate SE.

The system in this master's supports DSDS while maintaining the multi-tenancy support of JBIMulti2 system that is developed by Muhler [Muh11]. The system is called *ServiceMix Dynamic Service Discovery and Selection Support (SmxDSDS)*.

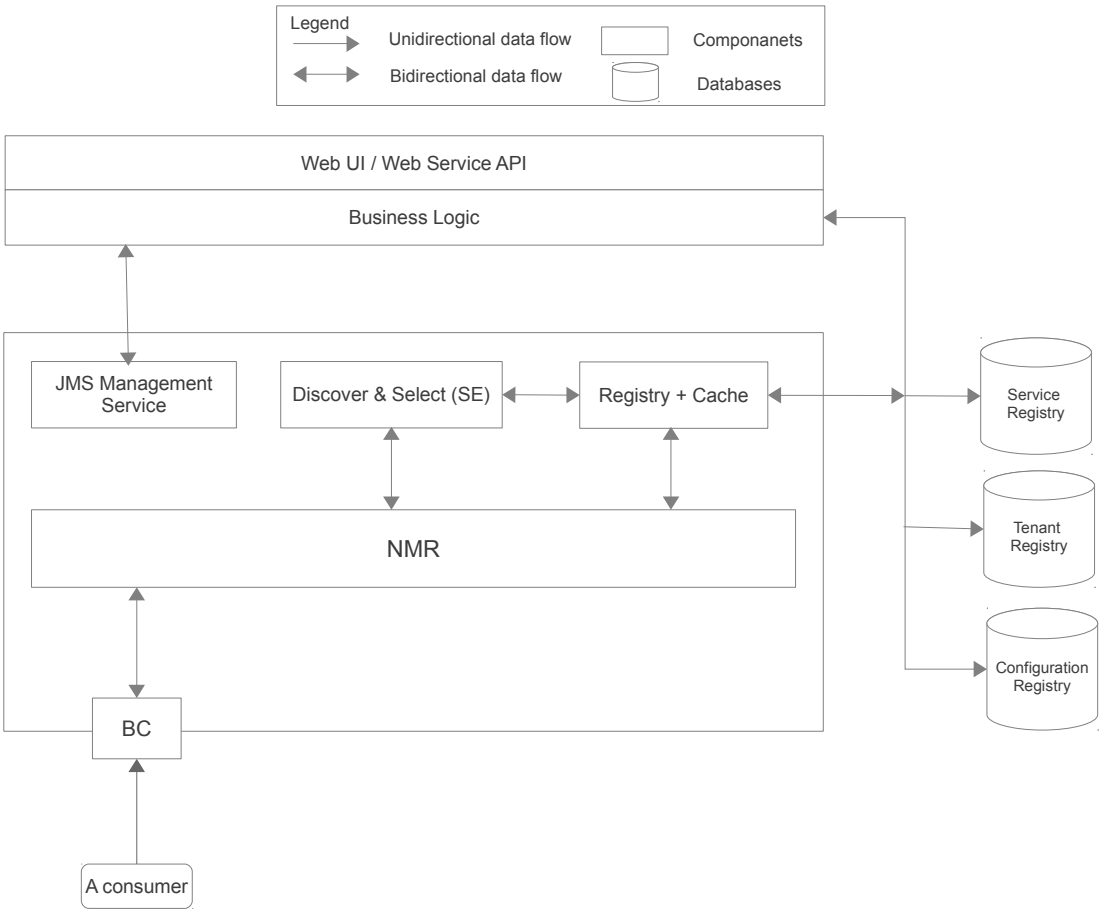

Figure 5.1.: Overview of the DSDS Extension to JBIMulti2. Arrows illustrate data flow between core components and resources.

DSDS component is a core part of ServiceMix Dynamic Service Discovery and Selection Support (SmxDSDS). The key operations are used to discover the suitable service candidates, such as service finding that includes policy normalization, intersection, and post-processing operations are carried out in this component. After discovering service candidates, the service selection is carried out that is configurable and extensible.

According to the diploma thesis Muhler [Muh11] the system relies on three databases: *Service Registry*, *Tenant Registry*, and *Configuration Registry*. In this work we make changes to service and configuration registries.

It is clear that the huge number of requests to SmxDSDS might affect to the performance of the ServiceMix adversely. Due to the performance reasons, we propose a separate component that accesses the data from the registries by utilizing cache mechanism.

Web Graphic User Interface (GUI) and Web API provide system administrators, tenant administrators, and tenant operators with the corresponding interfaces [Muh11]. In the current work, we expose new Web UI and Web API operations based on the use cases specified in the Section 5.5.

5.1.2. Scenarios

There are two main scenarios how SmxDSDS is used by the service consumers. First one is that DSDS enables consumer requests being processed dynamically. In this scenario the request process is comprised of only one step that is sending a request with the specified policy and the suitable service is discovered and selected by DSDS component. Second one is *Backward Compatibility Support* that means traditional way of working of the ServiceMix without DSDS and multi-tenancy support.

5.2. Dynamic Service Discovery and Selection

DSDS support that is developed for ServiceMix have several main steps. Firstly, after receiving the consumer and provider policy in *service discovery* step, the policies are intersected. Secondly, intersected policy is passed to *post-processing* step. Next step ranks service candidates that are result of preceding steps. Finally, the most suitable service is selected in the service selection step. All steps are explained in detail in the following section.

5.2.1. Service Discovery

As mentioned above the operation that used to find service candidates fulfilling the requirements specified in the consumer message policy. First, the component, upon receiving the message, retrieves the services of the tenant that the consumer belongs to. After that it evaluates the policy of the message to the policies of the retrieved (provider) services. To do so, the consumer policy and each provider policy are intersected to find a so-called *effective policy (intersected policy)*. An effective policy is a new policy that contains all alternatives of consumer and provider policy that are compatible. If the effective policy has at least one non-empty alternative, the service of the provider policy is considered to be compatible.

A consumer asks for its requirements while provider offer capabilities. Because of that the consumer policy should be the subset of the provider policy. In other words, all the service requests expressed in the requester policy, there exists at least one of the service offering capabilities that satisfies such a request. This means that all the assertions included in the requester policy must be included in the service provider policy as well. If at least one of the quality dimensions in the requester policy is not satisfied, then the process considers the two policies as not compliant [MF06].

To mean capability as opposed to requirement we add to every assertion in the service provider policy the attribute `wsp:Ignorable`. By marking a provider assertion with the

wsp:Ignorable attribute with a value of "true" the provider indicates that a requester may choose to either ignore such assertions or to consider them as a part of policy intersection. An assertion that may be ignored for policy intersection is called an ignorable assertion. Using the wsp:Optional attribute would be incorrect in this case, since it would indicate that the behaviour would not occur if the alternative without the assertion were selected [WSP07c].

Post-Processing

The intersected policy from the intersection has to be post-processed to evaluate attribute values of assertions. In the current work post-processing is carried out on the domain-specific assertion attributes of the effective policy to evaluate if the provider assertion attribute values fulfil the requester assertion attribute values. For instance, let's assume that the requestor has the assertion <cdhs:availabilityDegree cdhs:Percentage="98.9"/> in its policy, whereas the service provider has the assertion <cdhs:availabilityDegree cdhs:Percentage="99.99"/> in its policy. Obviously, the policies are compatible but their domain-specific assertions still need to be post-processed for cdhs:Percentage attribute values of the given assertions.

To provide domain-specific assertion processing, we propose to map a rule to each assertion. Specifying rules for domain-specific assertions in a separate file instead of hardcoding them in source code result in stronger re-usability. As it is obvious that the domain-specific attributes can have different types and their comparison logic is different. To provide wide range of types and comparison possibilities we employ XPath [XPA99] expressions. Thus, each rule mapped to a certain assertion qualified name (QName) is a XPath assertion that expresses evaluation logic of two compatible assertions.

Service Ranking

Different users have different preferences for the CDHS properties, therefore it is important for the properties to be specified from the perspective of service consumers. *Services Ranking* should be based on consumer preferred properties [YL04] specified in a policy document of a message. For example, the service consumer while describing its needs in a policy document can describe its preferences. This is done by prioritizing the required consumer requirements (i.e. adding optional priority attribute to each assertion).

We need service ranking in case no provider fulfils the complete requester requirements. Services are sorted based on a new term called *Service Rank Factor*. The sum of all the assertion priority values determine the rank factor value. As the priorities increase in ascending order (1 - the highest, 2, 3, 4 - the lowest), the service that has the minimum rank factor value is the most preferable service candidate, while the service with the maximum rank factor value is the least preferable service. An example is shown in the Figure 5.2 where Service B rank factor is the most preferable having the lowest Service Rank Factor value.

Service rank factor of a certain service is calculated as follows:

$$ServiceRankFactor = ass_1.p + ass_2.p + ... + ass_n.p$$

where, $ass_1, ass_2, ..., ass_n$ - are assertions; p - is priority.

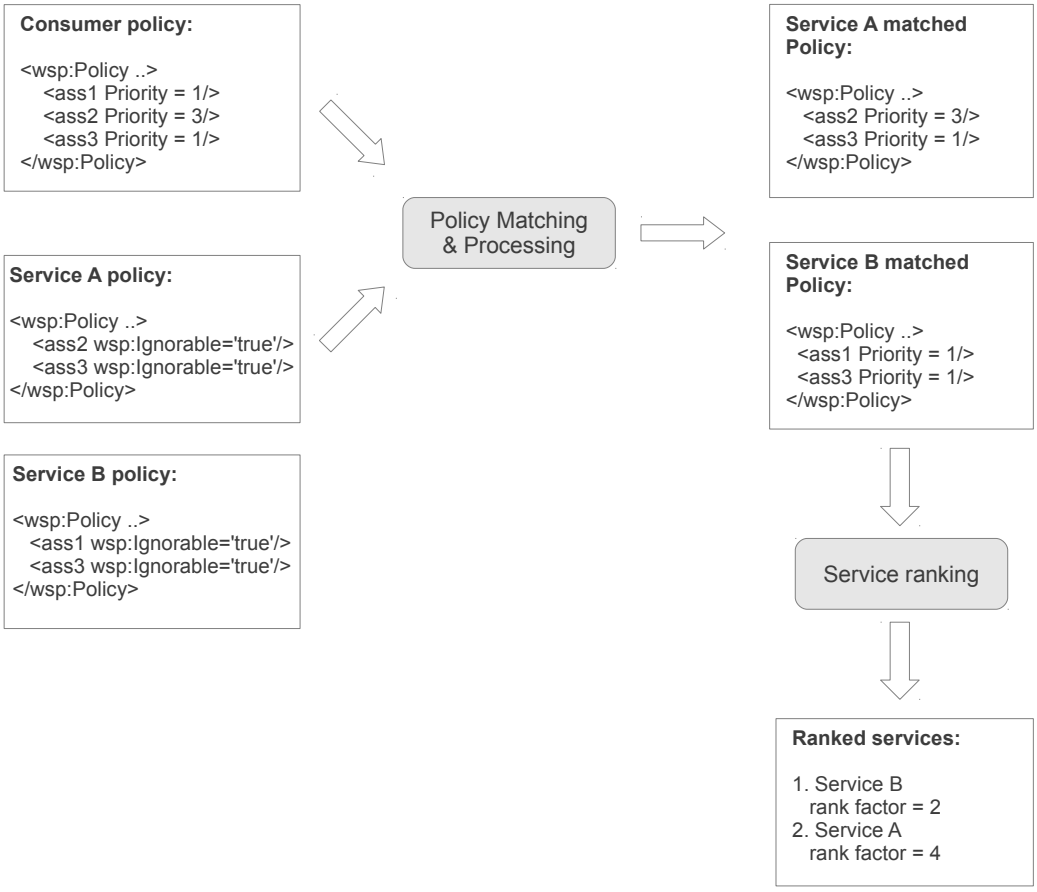

Figure 5.2.: Example for Service Ranking based on the Prioritization.

5.2.2. Service Selection

After discovering service candidates, the selection of the most suitable service is needed. To find the most suitable service the component runs service selection on the candidates found by the service discovery operation. Selection method is based on the method configured by system administrator. The selection methods can be extended by developers to add new selection methods like load-balancing selection that selects services based on service loads [AJ10].

5.3. Requester Policy Inclusion

A requester should include its policy to a message it sends to ServiceMix. Because of large number of assertions in WS-Policy Assertion language (see Chapter 4), a specified policy documents with this language may be big. Putting a big policy document in the message header deteriorates the message processing and in turn DSDS. As it can be currently sent only SOAP messages to ServiceMix to utilize DSDS, first solution could be to reference the requester policy from global repository. Another solution could be to include the requester policy as an attachment to the SOAP message.

In this work we add a policy document as an attachment to the message. First solution is not used because it leads consumers to carry out additional steps such as uploading their policies to the place where it can be referenced. To free this burden on a user, the attaching the policy to a SOAP message is used.

5.4. Cache

Current extensions to ServiceMix and to provide it as a PaaS with management application and multi-tenancy support in the Cloud lead to the connection of the ESB with the external databases that store services, tenants and configuration information. Increasing number of requests to external databases might affect to the performance of ServiceMix adversely. Therefore, ServiceMix needs to cache data retrieved from the external databases. To improve the performance, a separate component in ServiceMix is to be created that is responsible for data access and data caching of the data retrieved from the external databases. The reason to provide such functionalities in a separate component in ServiceMix is to make the component reusable.

The consistency of the data in the cache and databases must be ensured. If the data is changed via Management Web Application, such changes to the cache data in ServiceMix have to be made too. Temporary solution for this problem could be to set time-to-live to the cache data, which is realized in this work.

5.5. Use Cases

The use cases specified in this section identify interactions between the SmxDSDS and its users with Management and Administration Support. The interactions helps to discover the requirements of system administrator, tenant administrator, and tenant operator of the Management and Administration System of SmxDSDS.

We extended tenant administrator and tenant operator uses cases from the diploma thesis that are developed by Muhler [Muh11]. We specified use case extensions to the tenant administrator and tenant operator use cases by only specifying the changes with bold lines illustrated in the Figure 5.3 and 5.4 respectively.

It is obvious from the Tenant Administrator use cases that WS-Policy Assertion language and Rules are specific to tenants (companies). This means that each tenant can use SmxDSDS for its own WS-Policy extension and rules.

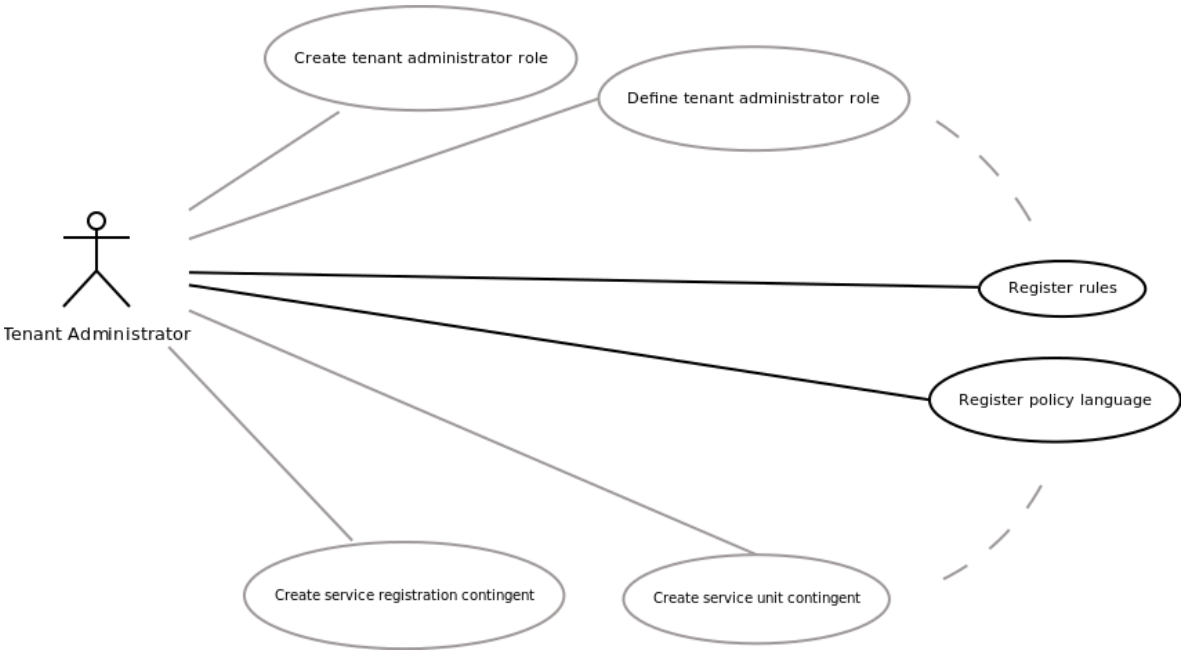

Figure 5.3.: Use case diagram for Tenant Administrator.

Name	**Register Policy Language**
Goal	The tenant administrator uploads the policy assertion language schema file in the configuration registry.
Actor	Tenant Administrator
Pre-Condition	The schema has to be specified as XML Schema Definition (XSD) file
Post-Condition	The schema is saved successfully
Post-Condition in Special Case	The schema is not saved.
Normal Case	1. The tenant administrator browse and selects WS-Policy Assertion language Schema, which is XSD file, and saves it.
Special Cases	1a. The file is not XSD file. a) The system shows an error message and aborts.

Table 5.1.: Description of Use Case *Register Policy Language*.

Name	**Register Rules**
Goal	The tenant administrator uploads the policy rules XML file in the configuration registry.
Actor	Tenant Administrator
Pre-Condition	The rules must comply to the rules schema (see Listing A.3) provided
Post-Condition	The rules XML is saved successfully.
Post-Condition in Special Case	The rules XML is not saved.
Normal Case	1. The tenant administrator browse and selects rules XML file and saves it.
Special Cases	1a. The rules XML does not comply with its rules schema (see Listing A.3). a) The system shows an error message and aborts.

Table 5.2.: Description of Use Case *Register Rules*.

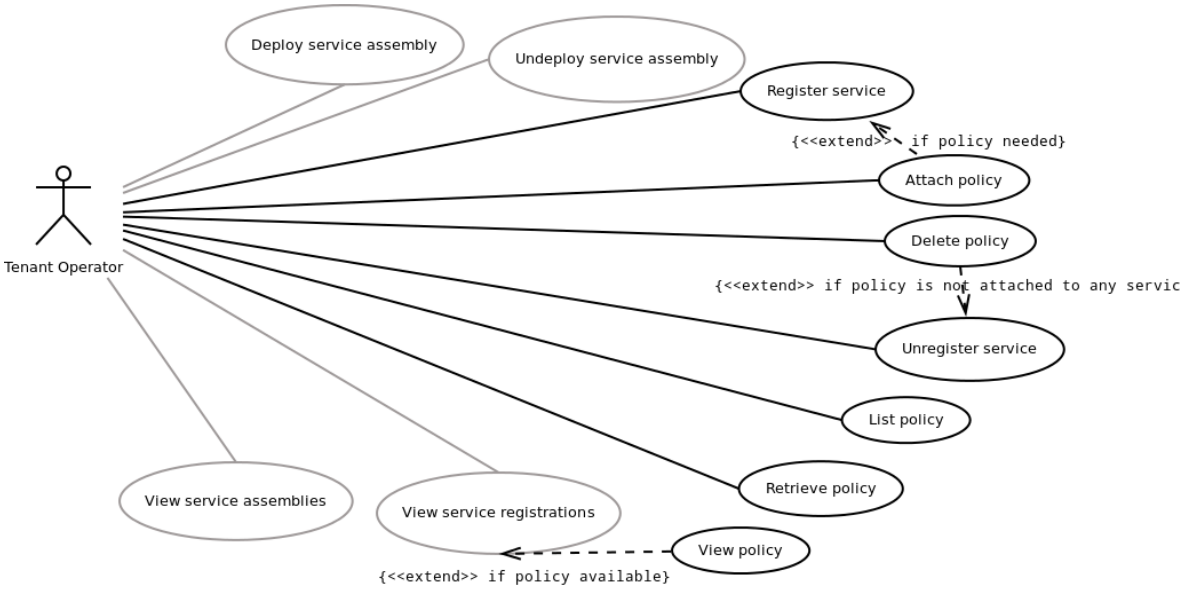

Figure 5.4.: Use case diagram for Tenant Operator.

Name	**Register Service**
Goal	The tenant operator wants to register a service using an existing service registration contingent. The tenant operator can register a service with either WSDL file or with the service name. During the service registration, the tenant operator can either to decide to attach a policy or not.
Actor	Tenant Operator
Pre-Condition	The tenant operator has the permission to use the service registration contingent.
Post-Condition	The service is registered and the number of available service registrations is decreased by one for the used service registration contingent. If the policy is attached, then it is saved and attached to the service.
Post-Condition in Special Case	The service is not registered and the service registration contingent is not used or service attached but the policy is neither attached nor saved.
Normal Case	1. The tenant operator enters either the location of a WSDL file or the service name, chooses the policy if it is decided so, and chooses the service registration contingent to use and confirms. 2. The system bundles a service assembly with the WSDL file, stores it to the service registry and updates the configuration manager.

Special Cases	1a. One of the files does not exist. a) The system shows an error message and aborts. 2a. The uploaded WSDL or policy file is not valid. a) The system shows an error message and aborts. 2b. Concurrently the service registration contingent is completely used for other service registrations. a) The system shows an error message and aborts. 2c. Concurrently the tenant operator has lost the permission to use the service registration contingent. a) The system shows an error message and aborts. 2d. The system can not finish the transaction with the service registry and the configuration registry. a) The system shows an error message and aborts.

Table 5.3.: Description of Use Case *Register Service*.

Name	**Attach Policy**
Goal	The tenant operator wants to attach a policy to a service.
Actor	Tenant Operator
Pre-Condition	The tenant operator has the permission to the service and the policy file has to be ready (specified).
Post-Condition	The policy is attached to the service.
Post-Condition in Special Case	The policy is neither attached nor saved to service registry.
Normal Case	1. The tenant operator chooses a service and browse the policy file location. 2. The system attaches the policy to the service and stores it in the service registry.
Special Cases	1a. The policy file doesn't exist. a) The system shows an error message and aborts. 2a. The service has been deleted by someone while attaching the policy. a) The system shows an error message and aborts. 2b. The uploaded file is no valid policy document file. a) The system shows an error message and aborts. 2c. Concurrently the tenant operator has lost the permission to use the service registration contingent. a) The system shows an error message and aborts. 2d. The system can not finish the transaction with the service registry. a) The system shows an error message and aborts.

Table 5.4.: Description of Use Case *Attach Policy*.

Name	**List Policy**
Goal	The tenant operator lists all the policies attached to its services.
Actor	Tenant Operator
Pre-Condition	There must be policies attached to its services
Post-Condition	The list of policies
Post-Condition in Special Case	There is no policy listed
Normal Case	1. The tenant requests all tenant-specific service policies.
Special Cases	

Table 5.5.: Description of Use Case *List Policy*.

Name	**Retrieve Policy**
Goal	The tenant operator retrieves a policy content.
Actor	Tenant Operator
Pre-Condition	The policy is attached to a service and exist in the service registry.
Post-Condition	The policy content is retrieved.
Post-Condition in Special Case	The policy content is not retrieved.
Normal Case	1. The tenant operator chooses a service and retrieves its policy.
Special Cases	1a. The policy has been deleted while requesting it. a) The system shows an error message.

Table 5.6.: Description of Use Case *Retrieve Policy*.

Name	**View Policy**
Goal	The tenant operator views a policy.
Actor	Tenant Operator
Pre-Condition	The policy is attached to a service and exits in the service registry.
Post-Condition	The policy information is retrieved.
Post-Condition in Special Case	The policy information is not retrieved.
Normal Case	1. The tenant operator chooses a policy of a service.
Special Cases	1a. The policy has been deleted while viewing it. a) The system shows an error message.

Table 5.7.: Description of Use Case *View Policy*.

Name	**Delete Policy**
Goal	The tenant operator wants to delete a policy.
Actor	Tenant Operator
Pre-Condition	The policy exists and tenant operator has the permission to the service.
Post-Condition	The policy is deleted.
Post-Condition in Special Case	The policy is not deleted.
Normal Case	1. The tenant operator chooses the service's policy and deletes it.
Special Cases	1a. The system can not finish the transaction with the service registry. a) The system shows an error message and aborts. 1b. The policy is attached to other service(s). a) The policy is not deleted from the service registry and show message reporting that it is attached to another service.

Table 5.8.: Description of Use Case *Delete Policy*.

5.6. Application Interfaces

This section describes a graphical user interface that makes the previously described functions of tenant-opeator available to humans. Moreover, a Web service API serves as interface to other applications and can be used to integrate JBIMulti2 into PaaS platform.

Web-based Graphical User Interface

In the current work, additional features for tenant-operator are added to the Web GUI already developed in Muhler [Muh11]. These additional functionalities are *attach policy*, *specify policy*, *edit policy*, *view policy*, and *delete policy* that are shown in the uses cases section 5.5.

Tenant Operator Content Panels

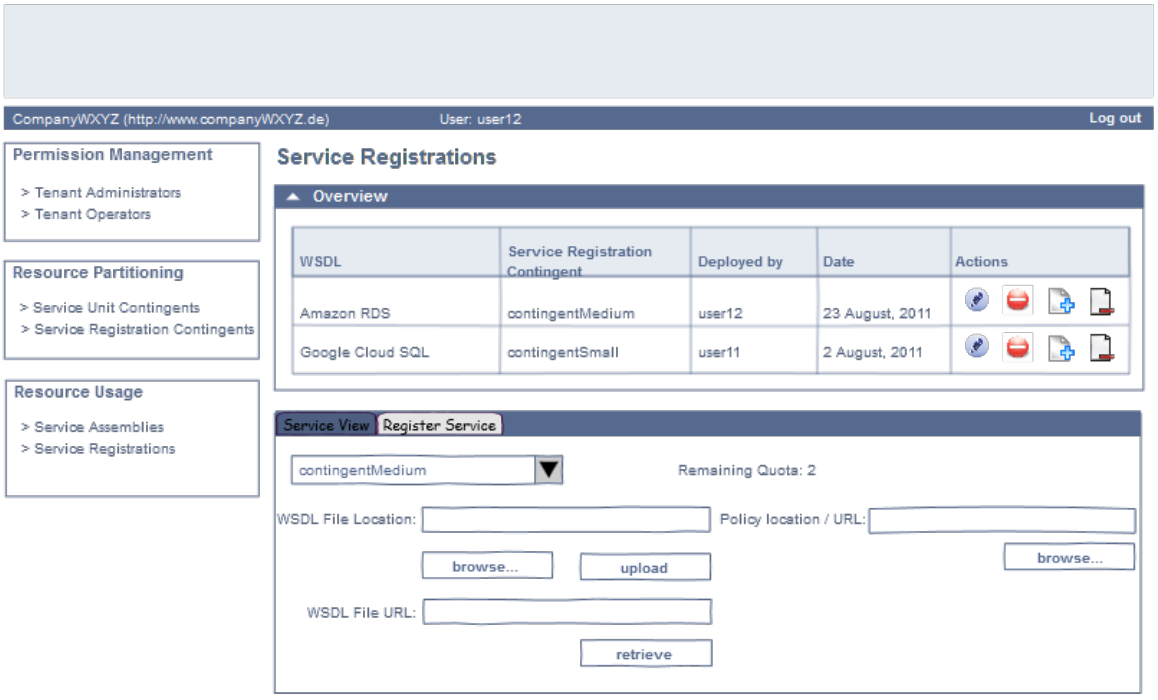

Figure 5.5.: Web UI: Sketch of Service Registrations Content Panel with Service Registration Tab.

The following content panels exist for tenant operators.

- *Service Assemblies Content Panel*: This is used to manage, deploy and undeploy service assemblies to ServiceMix [Muh11].
- *Service Registrations Content Panel* (see Fig. 5.5):

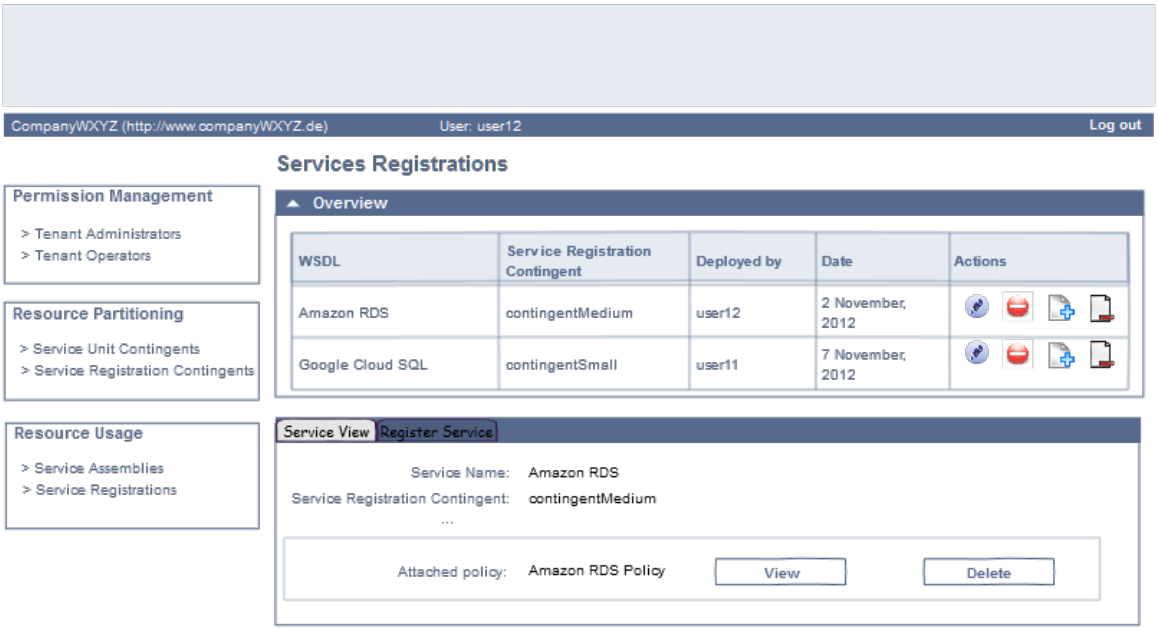

Figure 5.6.: Web UI: Sketch of Service Registrations Content Panel with Service View Tab.

- *Overview Content Panel* (see Fig. 5.5): The tenant operator can upload a service registration as a WSDL file, download a WSDL file, delete a service registration, add a policy, and delete the policy.
- *Register Service Tab* (see Fig. 5.5): The tenant operator can attach a policy to a service at the same wile registering the service.
- *Service View Tab* (see Fig. 5.6): The tenant operator can view all information of the service selected. The tenant operator can also view, edit, and delete the policy attached to the selected service.

Web Service API

We extended Web Service APIs of Muhler [Muh11] based on the use case analysis specified in the Section 5.5. The APIs are stateless and integrates WS-Security by claiming a valid tenant context on each request.

5.7. Non-functional Requirements

This section describes non-functional requirements that a productive version of SmxDSDS should satisfy. Considering the fact that SmxDSDS acts as a building block of a PaaS platform, we focused on different software qualities.

5.7.1. Extensibility

The added and extended components that provide key functionalities of the ESB for DSDS need to be extensible. The implementation of the current extension of ServiceMix must be extensible to allow further changes to the code.

5.7.2. Re-usability

The implemented components should be re-usable by providing good APIs and Interfaces. For example, DSDS in a separate component in ServiceMix allows it to be used by other components. Not only provide components re-usability but also the code should be re-usable.

5.7.3. Data Consistency

The data in the cache of SmxDSDS must be consistent with the data in the external databases.

5.7.4. Backward Compatibility

Backward compatibility meaning that even if a service binding is made dynamic, the hard-wired service binding (static selection & binding) style needs to stay as an option. In Addition, non-tenant awareness has to be supported meaning that requests without tenant context should be dealt as in a classical ESB.

5.7.5. Security

Security in terms of data isolation has to be considered between tenants.

5.7.6. Maintainability

The implementation should be well documented and the functionalities of the ESB should be decomposed into different components to maintain functional changes easily. As stated above the good APIs and Interfaces, which make the components re-usable, leverage maintainability too.

5.8. Special Cases

This section describes the special cases that may occur during the message/data flow in SmxDSDS. The cases are splitted up into places that have an impact on the message/data flow illustrated in the Figure 5.7. The columns in this figure illustrate the places where message/data is being processed. The special cases that need to be handled are marked with numbers and measures to be taken in these cases are given as follows:

- If the special cases 3) and one of 6), 7), and 8) occur, the whole list of services (tenant-specific services if a tenant context exists) is returned
- If the special cases 2) and 9) occur, no service is retuned
- If the special case 4) occurs, the partly satisfying services are ranked based on the rank factor (see Sect. 5.2.1)
- If the special case 7) or 9) occurs, there cannot be service candidates
- If the special case 5) occurs, the exception is thrown and dispatched to the requester

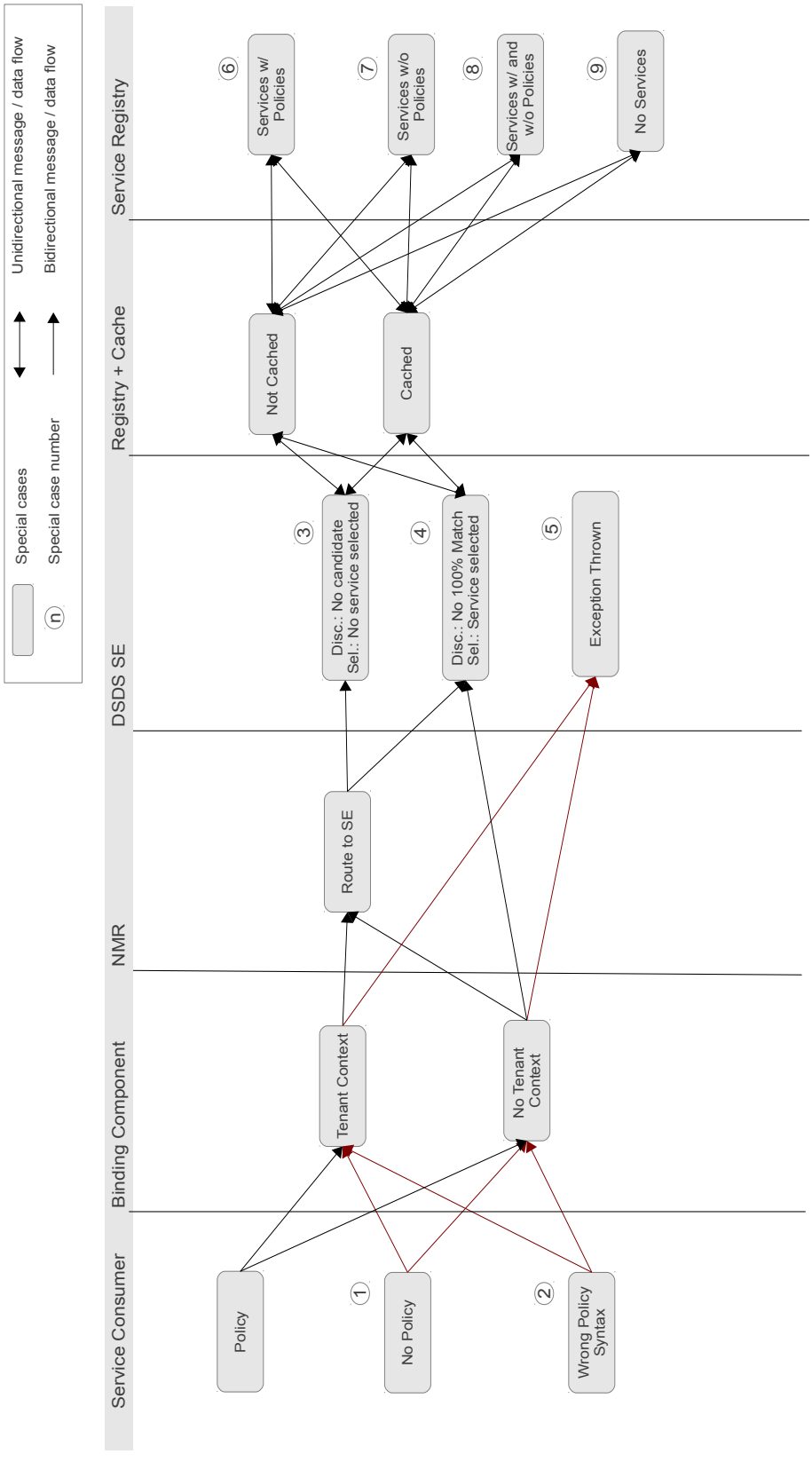

Figure 5.7.: Special Cases Diagram.

6. Design

This chapter describes an architectural and technological solution for the concepts and specified system requirements of Chapter 5. First, an overview of the system architecture and used technologies is given in Section 6.1. Then, the design of the main components are described.

6.1. Architectural Overview

As described in the Section 5.1, SmxDSDS is comprised of the new DSDS SE, Registry OSGi component as an intermediary between ServiceMix and the external databases. Moreover, connectivity of consumers to DSDS SE is provided with a HTTP consumer endpoint that supports SOAP messages. Policies are included to SOAP messages as attachments. The policy documents are specified with the new WS-Policy Assertion Language (see Chapter 4). The created/changed components in ServiceMix are illustrated with grey colors as illustrated in the Figure 6.1.

6.1.1. Components

As already mentioned, DSDS functionality to ServiceMix is being developed in a separate JBI Component (SE) in order to make DSDS functionality reusable and easier to configure the SE for backward compatibility. In turn, this component employs Apache Neethi [ANe12] that provides general framework for the programmers to use WS-Policy.

Registry component that plays a mediator role between external databases and the ServiceMix components is developed as a separate OSGi component. This component employs Java Database Connectivity (JDBC) PostgreSQL Driver [JDB] to connect to the external postgres databases. It also caches the data retrieved from the databases to improve the ServiceMix performance. The caching done by EHCache [Ehc] framework, which is a widely used open source Java distributed cache for general purpose caching, Java EE and light-weight containers.

Currently, in the Registry OSGi component, data retrieval services to get data from only service registry are provided. However, it can be also extended for tenant registry and configuration registry databases.

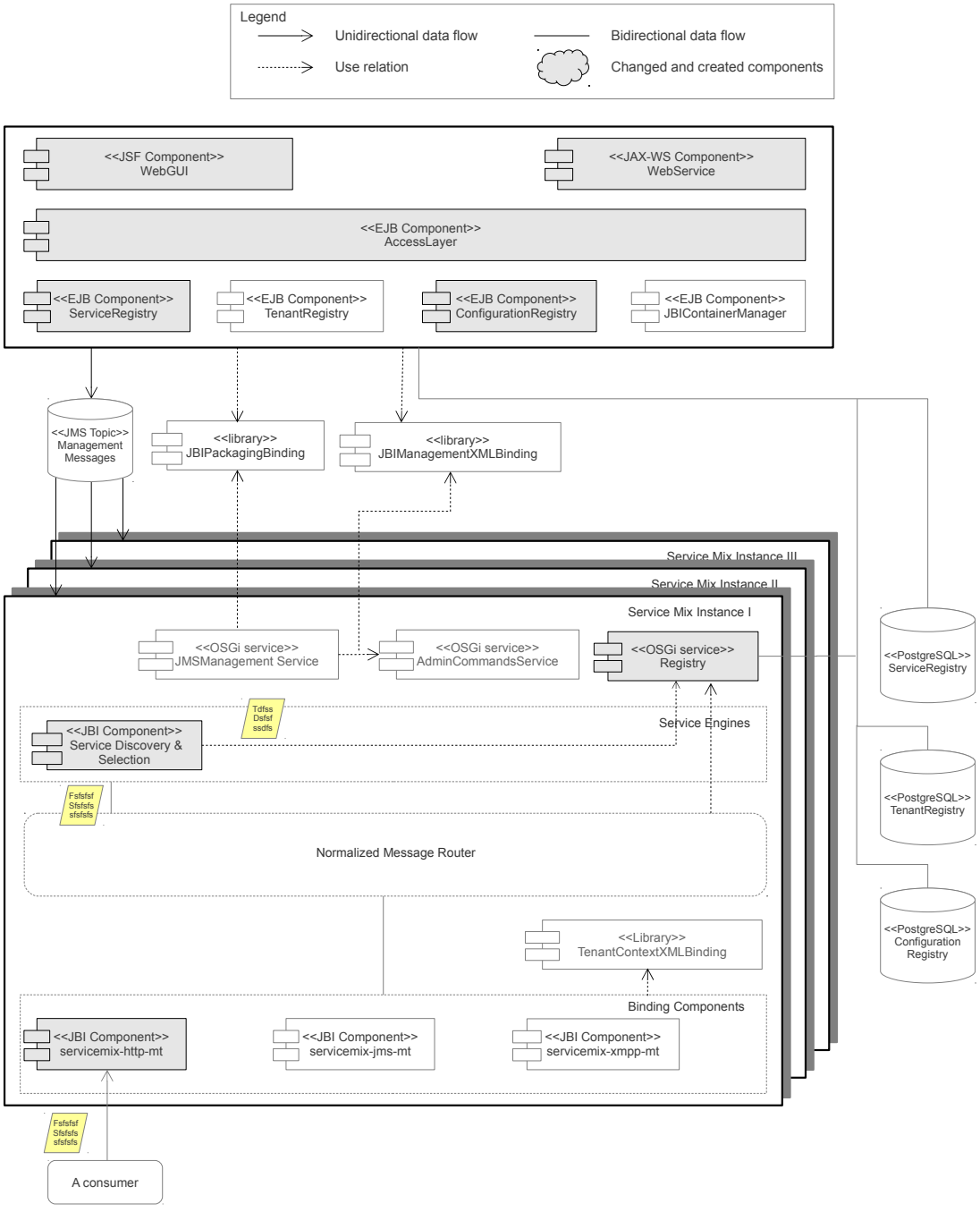

Figure 6.1.: Architecture.

6.1.2. Integration

DSDS functionality of SmxDSDS is provided to the external world through a HTTP consumer endpoint that consumes SOAP messages. The SU providing the endpoint connects to the service unit that configures DSDS SE. To access the necessary data from the service registry database, Registry OSGi component is created as an OSGi bundle, which is used by DSDS SE. JDBC PostgreSQL Driver connects the Registry OSGi component to the service registry database.

6.2. Extensions to ServiceMix

To provide the DSDS support to the ServiceMix, extensions to the ServiceMix are necessary. As the ServiceMix does not ship with DSDS support, DSDS is developed in a SE in the ServiceMix. Additionally, the Registry OSGi bundle is also developed in the ServiceMix that connects the SE to the databases.

6.2.1. Dynamic Service Discovery and Selection Service Engine

Upon receiving a consumer message DSDS SE starts to find the services that fulfil consumer needs in the attached policy. In order to do that it uses Registry OSGi component's *ServiceRegistry* class to get all available services that are specific to a tenant (if tenant context is included in the message) of the consumer. If the data exist in the cache of this component, the data is retrieved from the cache store. After receiving the services, the SE passes the services to *ServiceFinder* to discover and select services dynamically. Interactions between the main objects of DSDS SE are illustrated in the sequence diagram (see Fig. 6.2).

ServiceFinder object is used to discover the cloud data store service candidates depending on the requirements in the attached message policy. As we mentioned Apache Neethi [ANe12] is employed to take advantage of the capabilities provided for WS-Policy framework.

PostProcessing object is used to post-process intersected policy (intersected policy) found by service finder, to check if all attribute requirements are fulfilled.

ServiceRanker object is used to rank the service candidates based on the so called service rank factor (see Sect. 5.2.1). The service candidates are ranked unless the requester requirements are satisfied fully by these services, which is earlier described in Section 5.2.1.

The ServiceFinder object passes the found service candidates to *ServiceSelector* object that is responsible for service selection. There are several selection methods developed so far, namely, *first-met* and *random* service selection methods as depicted in the class diagram (see Fig. 6.3).

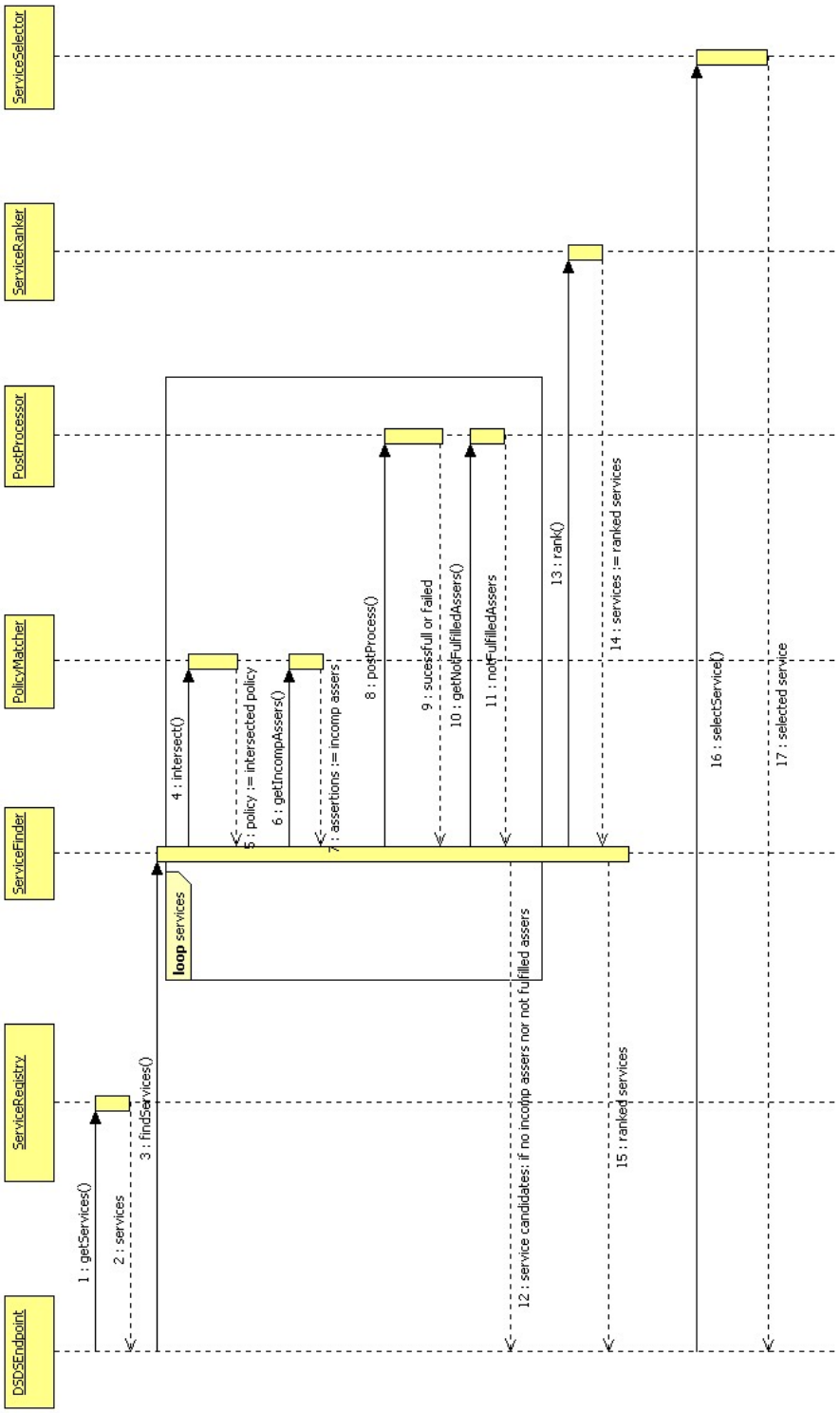

Figure 6.2.: Sequence Diagram of DSDS SE Portraying Basic Operations.

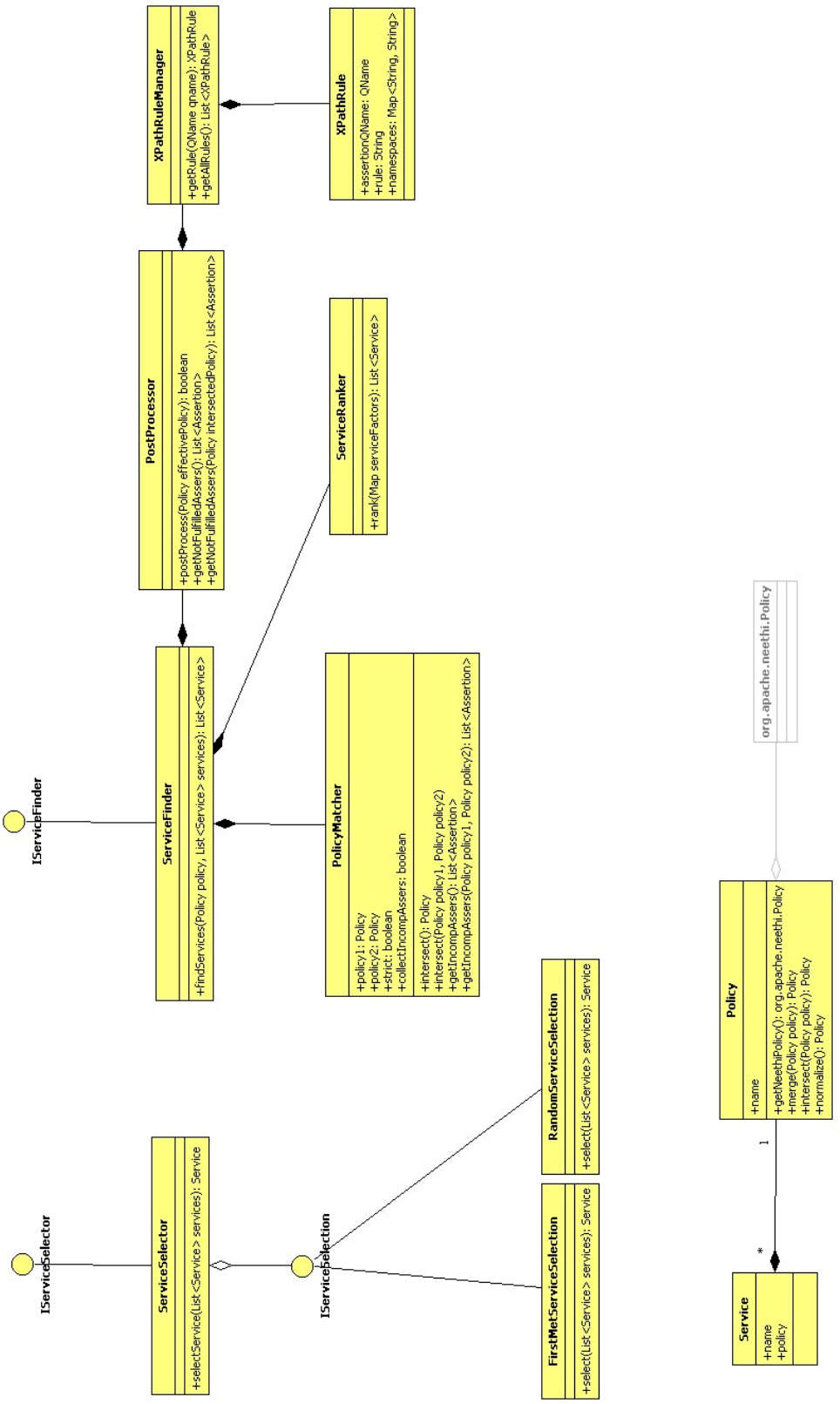

Figure 6.3.: Diagram of the Classes Used for DSDS.

6.2.2. Registry OSGi Bundle

As already mentioned Registry OSGi component is an intermediary between ESB and external databases that is used to retrieve data from the external databases and caches them. The Registry uses JDBC PostgresSQL Driver to connect to the databases. The driver allows Java programs to connect to a PostgreSQL database using standard and database independent Java code.

Large number of requests to the DSDS SE may lead to ServiceMix performance degradation. To prevent this, *Cache* capability is added to the Registry component that caches data coming from the service registry database to increase data access speed. Ehcache [Ehc] open-source Java caching tool is used that allows simple, fast, thread safe, standards based caching.

6.3. Web Application

According to Muhler [Muh11] the management application, further called Web application, has a three-tier-architecture, with a presentation layer, a business logic layer, and underlying resources. The presentation layer is separated into a Web GUI component and a Web service component. Whereas, the business logic is accessed via a superordinate AccessLayer that orchestrates use cases by calling the underlying business logic components: ServiceRegistry, TenantRegistry, ConfigurationRegistry, and JBIContainerManager. It is developed as a three-tier enterprise application utilizing Java Platform, Enterprise Edition v. 5 (Java EE 5) technology. This section describes the changes made to business logic layer and Web Service API.

Business logic access layer provides interfaces (comprising the use cases) for clients to access business logic. Access layer acts as a Session Facade [Mar02]. Three stateful *SystemAdminFacadeBean*, *TenantAdminFacadeBean*, and *TenantOperatorFacadeBean* provide a method for each use cases. In this work we added our use cases to *TenantOperatorFacadeBean* and *TenantAdminFacadeBean*.

Web Service API provide the same functionality as described in the use case analysis. Moreover, Web GUI is sketched and specified for future development, which also supplies operations based on the uses cases.

6.4. Database Schemes

In this section we describe our changes to the databases and their schemas developed by Muhler [Muh11].

6.4.1. Service Registry

As you see in Figure 6.4 a new entity *Policy* has been added to the service registry database schema. The policy entity store all information related to the attached policy of the service. Currently, policy entity has one-to-many relationship with service entity because a Cloud data store service can only have one policy document that describes its capabilities (several policies may have contradicting capabilities), but a policy can be attached to several services in the same scope of the tenant. The entity has *policyName* and *policyFile*, where policyName is used as an identifier while policyFile field is used to store policy document content.

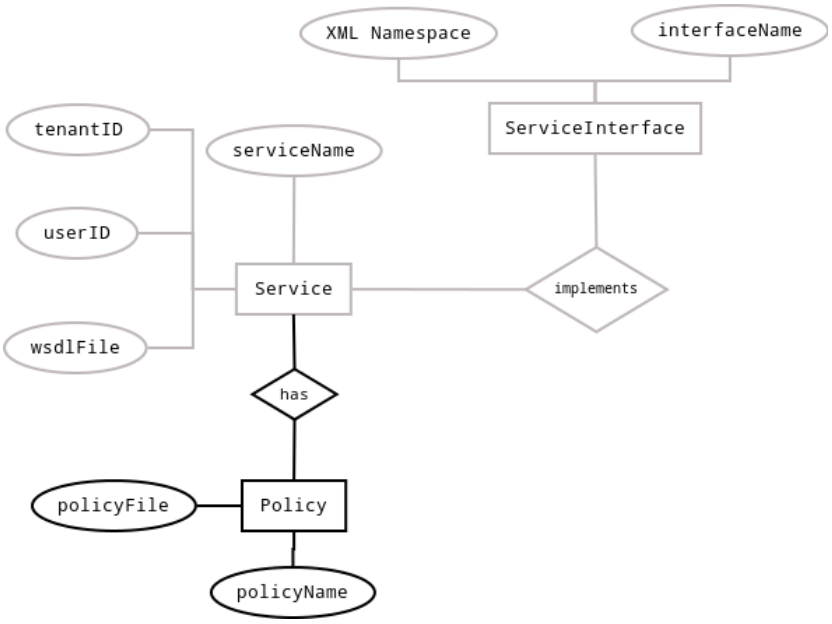

Figure 6.4.: Service Registry-Entity Relationship Diagram.

6.4.2. Tenant Registry

Regarding policy language XML schema and rules XML (see Fig. 5.3), they are stored in *KeyValuePair* entity of tenant registry database [Muh11]. The values 'POLICY_LANGUAGE' and 'RULES' can be the keys for policy language XML Schema file and rules XML file respectively, while the values mapped to these keys are content of the files.

7. Implementation and Validation

Various challenges occurred during the implementation of a prototype that complies to the concepts in Chapter 5 and conforms to the design solutions in Chapter 6. This chapter describes chosen implementation challenges in more detail. Moreover, the validation of the developed prototype is demonstrated.

7.1. Implementation

The implementation follows all the concepts (see Chapter 5) and the design solutions (see Chapter 6) that are developed in this work. The current prototype of SmxDSDS has been implemented as an extension to JBIMulti2 project, which is hierarchical Maven project. With the help of Apache Maven [AMV] developers can modularize software systems, restricting the dependencies between modules. The current prototype consists of two main modules that implement the extensions to Apache ServiceMix and extension to JBIMulti2 Web application. For the extensions of the ServiceMix we created a new JBI Component and new OSGi bundle with the help of Maven plugins. In addition, the third Maven module consists of SUs that expose the DSDS functionality of ServiceMix to consumers. Project files for the Integrated Development Environment (IDE) Eclipse have been generated with Maven and used during development. All Java source code was compiled with the Java Development Kit (JDK) 6.

7.1.1. Dynamic Service Discovery and Selection Service Engine

As the core component for SmxDSDS prototype, DSDS SE that is named *servicemix-dsds* is created and developed complying the design in the Section 6.2.1 and concepts in the Section 5.2 specified. In the pseudo-code given in Listing 1, the main part of DSDS algorithm is described, where in line 2 service candidates are discovered and in line 4 a suitable service is selected.

Algorithm 1 Main Part of Dynamic Service Discovery and Selection Algorithm

```
1: ...
2: serviceCandidates ← findServices(reqPolicy, services);
3: ...
4: selectedService ← select(serviceCandidates, selectionType);
5: ...
```

Service Discovery

Here service discovery implementation is described that are developed based on concepts specified in Section 5.2.1 and designs in Section 6.2.1. `IServiceFinder` interface provides `findService` method that discovers service candidates based on a policy and services passed as parameters to the method. First, it marks requester and service provider by putting `dsds:req="true"` for requester policy assertions and `dsds:prov="true"` for service provider assertions. The reason why marking is needed is that single effective policy document is passed to Post-Processing process to distinguish requester and provider assertions, which will be described shortly. Second, both policies are normalized and intersected to get an resulting effective policy. The intersection operation created in this work is not the same with the WS-Policy intersection because it goes to the end of nested assertions even if there incompatible child-assertions and collects incompatible assertions. If there is no alternatives of effective policy the policies are not compatible and cannot meet requester requirements at all. Otherwise, effective policy is passed to post-processing process. Then `PostProcessor` process the effective policy based on the attribute rules registered in the *rules* XML file. Finally, `ServiceFinder` returns service perfect candidates if there is at least one perfect match, otherwise it returns ranked service candidates that meet requester needs partly based on the fulfilled requirements. The pseudo-code of this whole steps is illustrated in Listing 2.

Intersection is operation provided by WS-Policy framework for combining two policies together to find an acceptable policy for both parties. In contrast to WS-Policy intersection algorithm, we developed here a new intersection algorithm that can collect incompatible assertions. The result of the new intersection algorithm is the combination of compatible policies. Adding such capabilities to the original intersection operation allows opportunity to rank service providers. In case of no perfect match, services that meet service requesters requirements partly have to be ranked based on how much they fulfil. Moreover, as mentioned we employ Apache Neethi framework to ease use of WS-Policy policy documents. We adapted the framework by making few changes to its source code.

PostProcessing is a class used to evaluate the defined XPath [XPA99] rules of assertion attributes. In this class we explicitly need to tackle the domain-specific post-processing issues in the bus. Therefore we define a general way how to define these domain-specific post-processing rules. As WS-Policy is rendered as XML, we choose XPath to describe these rules. It receives an intersected policy from Service Finder, then it evaluates attribute values of the same assertions based on the rules described in rules XML file. If all rules are evaluated to true then the post-processor return true, otherwise false. The post-processor can also keep and allow access to not fulfilled assertions that are not evaluated to true. Rules XML file registers all XPath rules for assertion attributes. Sample rules XML file is the Listing 7.1

Rules are created and registered by a tenant administrator to the domain-specific assertions and their attributes provided by the policy language schema. As you see in Listing 7.1, each XPath expression (rule) is mapped to a certain assertion qualified name (QName). In the XPath expression, `$reqAssertion` and `$provAssertion` prefixes are added to differentiate requester and provider part respectively, which are later replaced with corresponding assertion QNames. Rules XML file has to comply with the XML Schema given in Listing A.3.

Algorithm 2 Find Services

```
1: function FINDSERVICES(reqPolicy, services)
2:     for service: services do
3:         provPolicy ← service.getPolicy();
4:         if provPolicy = null then
5:             continue;
6:         end if
7:         markPolicy(reqPolicy, true);
8:         markPolicy(provPolicy, false);
9:         provPolicy.normalize();
10:        reqPolicy.normalize();
11:        polMatcher ← newPolicyMatcher(reqPolicy, provPolicy, strict = false, collIncompAssers = true)
12:        effPolicy ← polMatcher.intersect();
13:        if effPolicy alternatives exist then
14:            postProcessor ← newPostProcessor();
15:            boolean reqFulfilled ← postProcessor.postProcess(effPolicy);
16:            incompAssers ← polMatcher.getIncompAssers();
17:            notFulfilledAssers ← postProcessor.getNotFulfilledAssers();
18:            if reqFulfilled and incompAssers.isEmpy then
19:                serviceCandidates.add(service);
20:                existPerfectCandidate ← true ;
21:            else if not existPerfectCandidate and reqFulfilled then
22:                rankFactor ← rfc.calcRankFactor(policy, incompAssers);
23:                serviceFactor.put(service, rankFactor);
24:            else if not existPerfectCandidate and incompAssers.isEmpty then
25:                rankFactor ← rfc.calcRankFactor(policy, notFulfilledAssers);
26:                serviceFactor.put(service, rankFactor);
27:            else if not existPerfectCandidate then
28:                incompAssers.addAll(notFulfilledAssers);
29:                rankFactor ← rfc.calcRankFactor(policy, incompAssers);
30:                serviceFactor.put(service, rankFactor);
31:            end if
32:        end if
33:    end for
34:    if not existPerfectCandidate and not serviceFactor.isEmpty() then
35:        serviceCandidates ← ServiceRanker.rank(serviceFactor);
36:    end if
37:    return serviceCandidates;
38: end function
```

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <dsds:XPathRules xmlns:dsds="http://iaas.uni-stuttgart/dsds" xmlns:xsi="http://www.w3.org
       /2001/XMLSchema-instance"
3  xsi:schemaLocation="http://iaas.uni-stuttgart/dsds_rules.xsd"
4  xmlns:cdhs="http://iaas.uni-stuttgart/cdhs">
5
6    <dsds:Rule assertionQName="cdhs:timeToLaunchNewInstance">
7      <![CDATA[//$reqAssertion/@cdhs:Milliseconds < //$provAssertion/@cdhs:Milliseconds]]>
8    </dsds:Rule>
9
10   <dsds:Rule assertionQName="cdhs:latency">
11     <![CDATA[//$reqAssertion/@cdhs:Milliseconds < //$provAssertion/@cdhs:Milliseconds]]>
12   </dsds:Rule>
13
14   <dsds:Rule assertionQName="cdhs:throughput">
15     <![CDATA[//$reqAssertion/@cdhs:Milliseconds < //$provAssertion/@cdhs:Milliseconds]]>
16   </dsds:Rule>
17
18   <dsds:Rule assertionQName="cdhs:geographicLocation">
19     <![CDATA[//$reqAssertion/@cdhs:name = //$provAssertion/@cdhs:name]]>
20   </dsds:Rule>
21
22   <dsds:Rule assertionQName="cdhs:availabilityDegree">
23     <![CDATA[//$reqAssertion/@cdhs:Percentage <= //$provAssertion/@cdhs:Percentage]]>
24   </dsds:Rule>
25
26   <dsds:Rule assertionQName="cdhs:responseTime">
27     <![CDATA[//$reqAssertion/@cdhs:Milliseconds < //$provAssertion/@cdhs:Milliseconds]]>
28   </dsds:Rule>
29
30 </dsds:XPathRules>
```

Listing 7.1: Sample Rules XML File

We need *service ranking* in case no provider fulfils requester requirements fully as it is described earlier. `ServiceRanker` has the implementation of service ranking. Services are sorted based on the so called *service rank factor*. As already mentioned priority values ranges from 1 to 4 in decreasing importance. A service rank factor is calculated for each service (as shown in the lines 23, 26, 29 in Algorithm 2 that meet requester needs partly, and are sorted in increasing order of the rank factors, that are the services having lower rank factors are more preferable than the services having higher rank factors.

`DSDSEndpoint` is an entry class to DSDS SE. Upon receiving an incoming message, it starts DSDS process. Moreover, it takes into account the special cases, which are described in Section 5.8, that happen during DSDS process.

Service Selection

A service among the discovered services needs to be selected. In the current work we have developed only two service selection algorithms that are *First-Met* and *Random*. However, for

the future improvement new selection algorithms can be added by implementing `IServiceSelection` interface of which design is depicted in the class diagram 6.3.

In case of perfect match, one of the perfectly matched services are selected based on configured selection algorithm, which is selected with *first-met* algorithm in our work. In case of no perfect match, the most preferable service among the ranked services is selected that is first service in the list. The selection algorithm pseudo-code is given in Algorithm 3 in which selection method can be configured with `selectionType`.

Algorithm 3 Service Selection

```
1:  function SELECT(serviceCandidates, selectionType)
2:      if selectionType = FIRST-MET then
3:          selection ← newFirstMetServiceSelection();
4:          return selection.select(serviceCandidates);
5:      else if selectionType = RANDOM then
6:          selection ← newRandomServiceSelection();
7:          return selection.select(serviceCandidates);
8:      else if  then
9:          selection ← newFirstMetServiceSelection();
10:         return selection.select(serviceCandidates);
11:     end if
12: end function
```

7.1.2. DSDS Service/Endpoint

The SU *dsds-http-consumer-su* exposes DSDS service/endpoint to the users of ServiceMix. The service/endpoint should not be tenant-specific since it is a service for all users of ServiceMix. As each JBI Component defines what kind of configuration it relies on, the configuration has to be deployed as a SU to ServiceMix. The SU *dsds-http-consumer-su* together with *dsds-su* configuring the DSDS SE are directly (without management application) deployed in the SA *dsds-sa* as multiple service units targeting different JBI components can be deployed together as a SA. The reason for direct deployment is that as already mentioned, the service/endpoint is a global in SmxDSDS.

7.1.3. Registry Component

Currently, `servicemix-dsds` SE uses the services provided by `jbi.servicemix.registry` OSGi bundle for data access from service registry database with `ServiceRegistry` class. In turn, the bundle utilize *postgresql-9.1-901.jdbc3.jar* library to connect to Service Registry database. `ServiceRegistry` provides methods related only to Service Registry database that can be used to get tenant-specific or public services data, however, the component can be extended for the other databases in the future.

As earlier mentioned, caching mechanism is used to improve performance of SmxDSDS. For that purpose, we utilized Ehcache that is an open source, standards-based, and Java-based cache. Here we describe two main classes used to provide Caching, first one is `CacheFactory` that is in charge of creating an instance of *EhCacheWrapper* class, which is second one, used to cache all types of objects with their keys. The former class uses `CacheManager` class of Ehcache that controls creation of, access to, and removal of caches. The latter uses the cache objects classes implementing `Ehcache` interface. The `CacheFactory` creates the cache based on a configuration specified in a XML file. In this work, we specified the configurations in ehcache.xml file (see Listing 7.2).

```
1  <ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation
       =".../ehcache.xsd" dynamicConfig="true">
2    <diskStore path="java.io.tmpdir"/>
3    <cache name="jdbcCache"
4       maxEntriesLocalHeap="10000"
5       eternal="false"
6       timeToIdleSeconds="120"
7       timeToLiveSeconds="120"
8       maxEntriesLocalDisk="10000000"
9       diskExpiryThreadIntervalSeconds="120"
10      memoryStoreEvictionPolicy="LRU"
11      >
12       <persistence strategy="localTempSwap"/>
13     </cache>
14 </ehcache>
```

Listing 7.2: Cache Configuration.

Here we describe the used cache configurations in more detail (for further details refer to Ehcache documentation [Ehc]):

- diskStore (line 2 in Listing 7.2) – with this sub-property of the Ehcache configuration, the path to the directory where any required disk files will be created is configured.
- maxEntriesLocalHeap (line 4 in Listing 7.2) – This sets maximum entry number that is stored in a local heap. It can also be set with attribute *maxBytesLocalHeap* in bytes.
- eternal (line 5 in Listing 7.2) – if this attribute is set to "true", it overrides *timeToLiveSeconds* and *timeToIdleSeconds* so that no expiration can take place.
- timeToIdleSeconds (line 6 in Listing 7.2) – The maximum number of seconds an element can exist in the cache without being accessed
- timeToLiveSeconds (line 7 in Listing 7.2) – The maximum number of seconds an element can exist in the cache regardless of use
- maxEntriesLocalDisk (line 8 in Listing 7.2) – It is used to control the size of the swap-to-disk area in the local disk

- diskExpiryThreadIntervalSeconds (line 9 in Listing 7.2) – One thread per cache is used to remove expired elements. This optional attribute sets the interval between runs of the expiry thread.
- memoryStoreEvictionPolicy (line 10 in Listing 7.2) – This optional attribute is used to set eviction policy: LFU(Last Frequently Used), LRU(Last Recently Used), and FIFO(First In First Out).
- persistence (line 12 in Listing 7.2) – with this sub-property of the cache configuration, a RestartStore, which provides fast restartability and options for cache persistence, strategy can be set. After any restart, data that was last in the cache will automatically load from disk into the RestartStore, and from there the data will be available to the cache.

7.1.4. Changes to Management Application

We already mentioned that we change the Web service API by adding methods same with the uses cases (see Sect. 5.5) for tenant-operator and tenant-administrator.

Now Web Service API users can attach a policy document while registering a service with the help of *register service* Web service. *Register service* Web service can register a service either with a policy or without policy. The management application users might want to attach a policy to a service later with the help of attach policy, which accepts service name and policy file as base64 binary format [FB96]. The last Web service *delete policy* is used to delete a policy document from the database. In turn, a tenant administrator with the Web services provided for tenant-admin can now register policy language and post-processing rules.

7.2. Validation

We need to validate implemented prototype of SmxDSDS to check if it gives desired results. For this purpose, we use the Scenario (see Sect. 1.1) to validate the prototype.

7.2.1. Initialization

We are provided with an Ubuntu Virtual Machine (VM) in the Cloud that has installed ServiceMix instance and the Java EE 5 certified application server Java Open Application Server (JOnAS) 5.2.2 [OWJ] in which the configured databases Tenant Registry, Service Registry, and Configuration Registry are created on PostgreSQL 9.1.1 [PSQ]. We only need to start JOnAS application server, Postgres server, the ServiceMix instance to start using JBIMulti2 and SmxDSDS.

The validation of the prototype of SmxDSDS occurs on a single ServiceMix instance in the VM. As DSDS is provided as a global service/endpoint in the ServiceMix the core component of the prototype DSDS SE and SUs that configure it are directly deployed to the ServiceMix

instance without JBIMulti2. Moreover, OSGi bundle should be deployed along with DSDS SE because the SE depends on the bundle.

We assume that there is no service registered in the ServiceMix. Therefore, via JBIMulti2 tenants need to register the Cloud data store services with their policy documents that express their capabilities. The syntax of the policies are based on a certain policy language, in our work, the policies must comply with WS-Policy Assertion language. A tenant admin registers the XML Schema of this policy language as well as the rules used for post-processing.

Once the DSDS SE and its SUs are deployed in the ServiceMix plus the set of Cloud data store services are registered, we can validate SmxDSDS by sending request messages with user policies. To interact with the provided DSDS service/endpoint we use soapUI 3.6.1 [SOAa], a graphical SOAP-based Web services testing tool. The validation is described in the following section in more detail.

7.2.2. Dynamic Service Discovery and Selection Validation

Here we demonstrate general use of SmxDSDS. In this example we specified two different policies user of tenants in the Listings 7.3, 7.6 that have the requirements of different users of tenants. The tenant of both users registered five different services with five different policies. The registered services are Amazon RDS MySQL with the policy in the Listing 7.4, Amazon RDS PostgreSQL with the policy in the Listing 7.5, Amazon Dynamo DB with policy in the Listing 7.7, Google Cloud SQL with the policy in the Listing A.4, and SQL Database with in the policy the Listing A.5. The policies are sample (excerpts taken from real policies) policies to make them to a reader easier to read and understand because the real policy documents of the registered services are big.

First, we validate SmxDSDS with first user policy in the Listing 7.3. As it is illustrated in the Figure 7.3, on the bottom left corner we can see the user policy attached. On the right-hand side, the response returned discovered service, which is Amazon RDS MySQL. We now explain why the service Amazon RDS MySQL is selected. If you look at the provider policies (see Sect. A.3) the Amazon RDS MySQL and Amazon RDS PostgreSQL services have capabilities that can fulfil the requirements in the user policy better than the other services. These two services are ranked top by `ServiceRanker` where Amazon RDS MySQL is ranked the best. If you look at the provider policies, `<cdhs:scalabilityType ..>` (the line 33 in the Listing 7.4) assertion is provided in Amazon RDS MySQL but not in Amazon RDS PostgresSQL, while the assertions `<cdhs:degree ..>` and `<cdhs:automaticFailover ..>` (the lines 18 - 22 and 33 in the Listing 7.5) are provided in Amazon RDS PostgresSQL but not in Amazon RDS MySQL. Additionally, `<cdhs:availabilityDegree cdhs:Percentage='99.99'>` (the line 33 in the Listing 7.4) of Amazon RDS MySQL service fulfil the user assertion `<cdhs:availabilityDegree cdhs:Percentage='99.98'>`(the line 38 in the Listing 7.3) but the user assertion `<availabilityDegree Percentage='99.97'>` (the line 33 in the Listing 7.5) of Amazon RDS Postgres SQL does not fulfil. Now if you look at the user policy, the user has the highest priority values for assertions `<scalabilityType cdhs:Priority='1'>` and `<cdhs:availabilityDegree cdhs:Priority='1'` (the lines 16-21 and 38 in the Listing 7.3),

however, it has lower priorities for the assertions <cdhs:degree cdhs:Priority='3'> and <cdhs:automaticFailover cdhs:Priority='2'> (the lines 23-27 and 37 in the Listing 7.3). Thus, Amazon RDS MySQL meets requirements of the user that have higher priorities than the requirements Amazon RDS PostgresSQL meets.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsp:Policy xmlns:wsp="http://www.w3.org/ns/ws-policy"
3      xmlns:cdhs="http://iaas.uni-stuttgart/cdhs"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://www.w3.org/ns/ws-policy http://www.w3.org/2007/02/ws-policy.
           xsd">
6
7      <cdhs:scalability>
8          <wsp:Policy>
9
10             <cdhs:automationDegree cdhs:Priority="2">
11                 <wsp:Policy>
12                     <cdhs:manual/>
13                 </wsp:Policy>
14             </cdhs:automationDegree>
15
16             <cdhs:scalabilityType cdhs:Priority="1">
17                 <wsp:Policy>
18                     <cdhs:vertical/>
19                     <cdhs:horizontal/>
20                 </wsp:Policy>
21             </cdhs:scalabilityType>
22
23             <cdhs:degree cdhs:Priority="3">
24                 <wsp:Policy>
25                     <cdhs:limited/>
26                 </wsp:Policy>
27             </cdhs:degree>
28
29             <cdhs:timeToLaunchNewInstance cdhs:Milliseconds="2100000" cdhs:Priority="3"/>
30
31         </wsp:Policy>
32     </cdhs:scalability>
33
34     <!-- Availability -->
35     <cdhs:availability>
36         <wsp:Policy>
37             <cdhs:automaticFailover cdhs:Priority="2"/>
38             <cdhs:availabilityDegree Percentage="99.98" cdhs:Priority="1"/>
39         </wsp:Policy>
40     </cdhs:availability>
41
42     <!-- Security -->
43     <cdhs:security>
44         <wsp:Policy>
45             <cdhs:storageEncryption cdhs:Priority="1"/>
46             <cdhs:transferEncryption cdhs:Priority="1"/>
47             <cdhs:firewall cdhs:Priority="3"/>
```

```
48              <cdhs:authentication cdhs:Priority="1"/>
49              <cdhs:confidentiality cdhs:Priority="1"/>
50              <cdhs:integrity cdhs:Priority="1"/>
51          </wsp:Policy>
52      </cdhs:security>
53
54      <!-- Data Constraints -->
55      <cdhs:dataConstraints>
56          <wsp:Policy>
57              <cdhs:maxSizePerInstance cdhs:Gigabyte="1024" cdhs:Priority="2"/>
58              <cdhs:predefinedInstanceSize cdhs:Priority="2"/>
59          </wsp:Policy>
60      </cdhs:dataConstraints>
61
62      <!-- Cloud Computing -->
63      <cdhs:cloudComputing>
64          <wsp:Policy>
65              <cdhs:serviceModel cdhs:Priority="2">
66                  <wsp:Policy>
67                      <cdhs:paas/>
68                  </wsp:Policy>
69              </cdhs:serviceModel>
70
71              <cdhs:deploymentModel cdhs:Priority="1">
72                  <wsp:Policy>
73                      <cdhs:public/>
74                  </wsp:Policy>
75              </cdhs:deploymentModel>
76          </wsp:Policy>
77      </cdhs:cloudComputing>
78
79
80  </wsp:Policy>
```

Listing 7.3: Company B first user policy.

The following policy is attached to Amazon RDS MySQL service:

```
1   <wsp:Policy Name="http://iaas.uni-stuttgart/cdhs/AmazonRDS-MySQL" wsu:Id="AmazonRDSMySQL"
2       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3       xsi:schemaLocation="http://www.w3.org/ns/ws-policy http://www.w3.org/2007/02/ws-policy.
            xsd"
4       xmlns:wsp="http://www.w3.org/ns/ws-policy"
5       xmlns:cdhs="http://iaas.uni-stuttgart/cdhs"
6       xmlns:xs="http://www.w3.org/2001/XMLSchema"
7       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility
            -1.0.xsd">
8
9           <!-- Scalability -->
10          <cdhs:scalability wsp:Ignorable="true">
11              <wsp:Policy>
12                  <cdhs:automationDegree wsp:Ignorable="true">
13                      <wsp:Policy>
14                          <cdhs:manual/>
```

```
15                      </wsp:Policy>
16                  </cdhs:automationDegree>
17
18                  <cdhs:scalabilityType wsp:Ignorable="true">
19                      <wsp:Policy>
20                          <cdhs:vertical/>
21                          <cdhs:horizontal/>
22                      </wsp:Policy>
23                  </cdhs:scalabilityType>
24
25                  <cdhs:timeToLaunchNewInstance cdhs:Milliseconds="2100000" wsp:Ignorable="
                        true"/> <!-- 35 * 60 * 100 milliseconds = 35 minutes -->
26
27              </wsp:Policy>
28          </cdhs:scalability>
29
30          <!-- Availability -->
31          <cdhs:availability wsp:Ignorable="true">
32              <wsp:Policy>
33                  <cdhs:availabilityDegree Percentage="99.99" wsp:Ignorable="true"/>
34              </wsp:Policy>
35          </cdhs:availability>
36
37          <!-- Security -->
38          <cdhs:security wsp:Ignorable="true">
39              <wsp:Policy>
40                  <cdhs:storageEncryption wsp:Ignorable="true"/>
41                  <cdhs:transferEncryption wsp:Ignorable="true"/>
42                  <cdhs:firewall wsp:Ignorable="true"/>
43                  <cdhs:authentication wsp:Ignorable="true"/>
44                  <cdhs:confidentiality wsp:Ignorable="true"/>
45                  <cdhs:integrity wsp:Ignorable="true"/>
46              </wsp:Policy>
47          </cdhs:security>
48
49          <!-- Data Constraints -->
50          <cdhs:dataConstraints wsp:Ignorable="true">
51              <wsp:Policy>
52                  <cdhs:maxSizePerInstance cdhs:Gigabyte="1024" wsp:Ignorable="true"/>
53                  <cdhs:predefinedInstanceSize wsp:Ignorable="true"/>
54              </wsp:Policy>
55          </cdhs:dataConstraints>
56
57          <!-- Cloud Computing -->
58          <cdhs:cloudComputing wsp:Ignorable="true">
59              <wsp:Policy>
60                  <cdhs:serviceModel wsp:Ignorable="true">
61                      <wsp:Policy>
62                          <cdhs:paas/>
63                      </wsp:Policy>
64                  </cdhs:serviceModel>
65              </wsp:Policy>
66          </cdhs:cloudComputing>
67
68  </wsp:Policy>
```

Listing 7.4: Amazon RDS MySQL Service Provider Policy.

The following policy is attached to Amazon RDS Postgres SQL service:

```
<wsp:Policy Name="http://iaas.uni-stuttgart/cdhs/AmazonRDS-PostgresSQL" wsu:Id="
    AmazonRDSPostgresSQL"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/ns/ws-policy http://www.w3.org/2007/02/ws-policy.
        xsd"
    xmlns:wsp="http://www.w3.org/ns/ws-policy"
    xmlns:cdhs="http://iaas.uni-stuttgart/cdhs"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility
        -1.0.xsd">

        <!-- Scalability -->
        <cdhs:scalability wsp:Ignorable="true">
            <wsp:Policy>
                <cdhs:automationDegree wsp:Ignorable="true">
                    <wsp:Policy>
                        <cdhs:manual/>
                    </wsp:Policy>
                </cdhs:automationDegree>

                <cdhs:degree wsp:Ignorable="true">
                    <wsp:Policy>
                        <cdhs:limited/>
                    </wsp:Policy>
                </cdhs:degree>

                <cdhs:timeToLaunchNewInstance cdhs:Milliseconds="2100000" wsp:Ignorable="
                    true"/> <!-- 35 * 60 * 100 milliseconds = 35 minutes -->

            </wsp:Policy>
        </cdhs:scalability>

        <!-- Availability -->
        <cdhs:availability wsp:Ignorable="true">
            <wsp:Policy>
                <cdhs:automaticFailover wsp:Ignorable="true"/>
                <cdhs:availabilityDegree Percentage="99.97" wsp:Ignorable="true"/>
            </wsp:Policy>
        </cdhs:availability>

        <!-- Security -->
        <cdhs:security wsp:Ignorable="true">
            <wsp:Policy>
                <cdhs:storageEncryption wsp:Ignorable="true"/>
                <cdhs:transferEncryption wsp:Ignorable="true"/>
                <cdhs:firewall wsp:Ignorable="true"/>
                <cdhs:authentication wsp:Ignorable="true"/>
                <cdhs:confidentiality wsp:Ignorable="true"/>
```

```
45                  <cdhs:integrity wsp:Ignorable="true"/>
46              </wsp:Policy>
47          </cdhs:security>
48
49          <!-- Data Constraints -->
50          <cdhs:dataConstraints wsp:Ignorable="true">
51              <wsp:Policy>
52                  <cdhs:maxSizePerInstance cdhs:Gigabyte="1024" wsp:Ignorable="true"/>
53                  <cdhs:predefinedInstanceSize wsp:Ignorable="true"/>
54              </wsp:Policy>
55          </cdhs:dataConstraints>
56
57          <!-- Cloud Computing -->
58          <cdhs:cloudComputing wsp:Ignorable="true">
59              <wsp:Policy>
60                  <cdhs:serviceModel wsp:Ignorable="true">
61                      <wsp:Policy>
62                          <cdhs:paas/>
63                      </wsp:Policy>
64                  </cdhs:serviceModel>
65
66              </wsp:Policy>
67          </cdhs:cloudComputing>
68
69  </wsp:Policy>
```

Listing 7.5: Amazon RDS Postgres SQL Service Provider Policy.

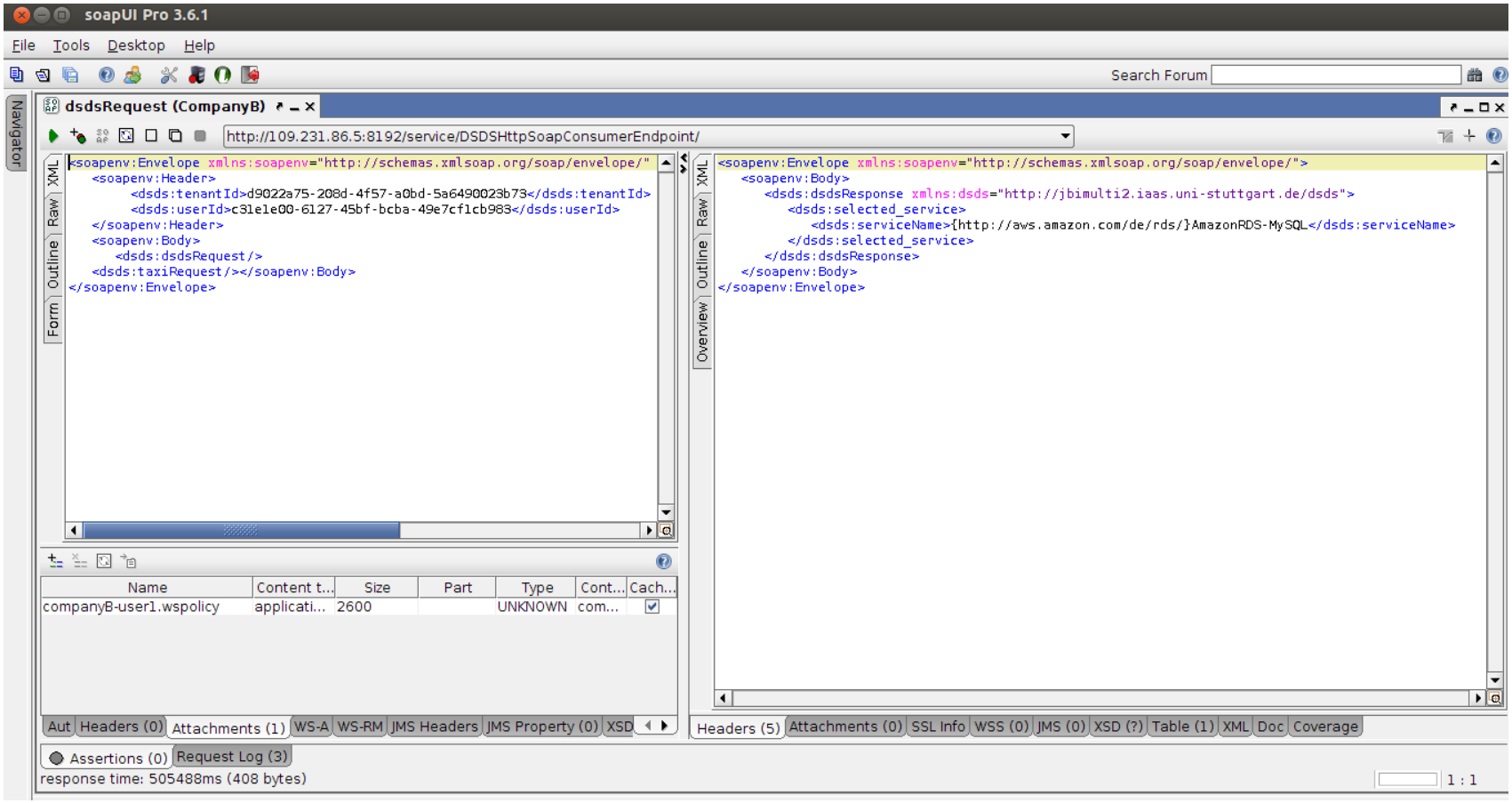

Figure 7.1.: DSDS Request executed by a tenant user with soapUI Pro 3.6.1 [SOAa]. On the left-hand side, top part is the SOAP request message and the bottom part is the message policy attachment. The response contains the discovered service.

Second user policy in the Listing 7.6 is used for validation. From the Figure 7.2, it is shown on the right-hand side that Amazon DynamoDB is the selected service. Let's explain why it is discovered as the best service for the user requirements. If you look at the provider policies (see Sect. A.3), the Amazon Dynamo DB service meets all the user requirements, while others only meet them partly. In this case, ranking is not even needed, because Amazon Dynamo DB perfectly fulfills the requirements.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy xmlns:wsp="http://www.w3.org/ns/ws-policy"
    xmlns:cdhs="http://iaas.uni-stuttgart/cdhs"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/ns/ws-policy http://www.w3.org/2007/02/ws-policy.
        xsd">

    <!-- Scalability -->
    <cdhs:scalability>
        <wsp:Policy>
            <cdhs:automaticScalabilityCriterion>
                <wsp:Policy>
                    <cdhs:systemLoadCriteria cdhs:Priority="2"/>
                    <cdhs:latencyCriteria cdhs:Priority="1"/>
                </wsp:Policy>
            </cdhs:automaticScalabilityCriterion>
        </wsp:Policy>
    </cdhs:scalability>

    <!-- Security -->
    <cdhs:security>
        <wsp:Policy>
            <cdhs:transferEncryption cdhs:Priority="1"/>
            <cdhs:authentication cdhs:Priority="1"/>
        </wsp:Policy>
    </cdhs:security>

    <!-- Performance -->
    <cdhs:performance>
        <wsp:Policy>
            <cdhs:latency Milliseconds="9" cdhs:Priority="2"/>
        </wsp:Policy>
    </cdhs:performance>

</wsp:Policy>
```

Listing 7.6: Company B second user policy.

The following policy is attached to Amazon Dynamo DB service:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy xmlns:wsp="http://www.w3.org/ns/ws-policy" Name="http://iaas.uni-stuttgart/cdhs/
    AmazonDynamoDB"
    xmlns:cdhs="http://iaas.uni-stuttgart/cdhs"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
5       xsi:schemaLocation="http://www.w3.org/ns/ws-policy_http://www.w3.org/2007/02/ws-policy.
            xsd">
6
7       <!-- Scalability -->
8       <cdhs:scalability wsp:Ignorable="true">
9           <wsp:Policy>
10              <cdhs:automaticScalabilityCriterion wsp:Ignorable="true">
11                  <wsp:Policy>
12                      <cdhs:systemLoadCriteria wsp:Ignorable="true"/>
13                      <cdhs:latencyCriteria wsp:Ignorable="true"/>
14                  </wsp:Policy>
15              </cdhs:automaticScalabilityCriterion>
16          </wsp:Policy>
17      </cdhs:scalability>
18
19      <!-- Security -->
20      <cdhs:security wsp:Ignorable="true">
21          <wsp:Policy>
22              <cdhs:transferEncryption wsp:Ignorable="true"/>
23              <cdhs:authentication wsp:Ignorable="true"/>
24          </wsp:Policy>
25      </cdhs:security>
26
27      <!-- Performance -->
28      <cdhs:performance wsp:Ignorable="true">
29          <wsp:Policy>
30              <cdhs:latency Milliseconds="10" wsp:Ignorable="true"/>
31          </wsp:Policy>
32      </cdhs:performance>
33
34  </wsp:Policy>
```

Listing 7.7: Amazon Dynamo DB Service Provider Policy.

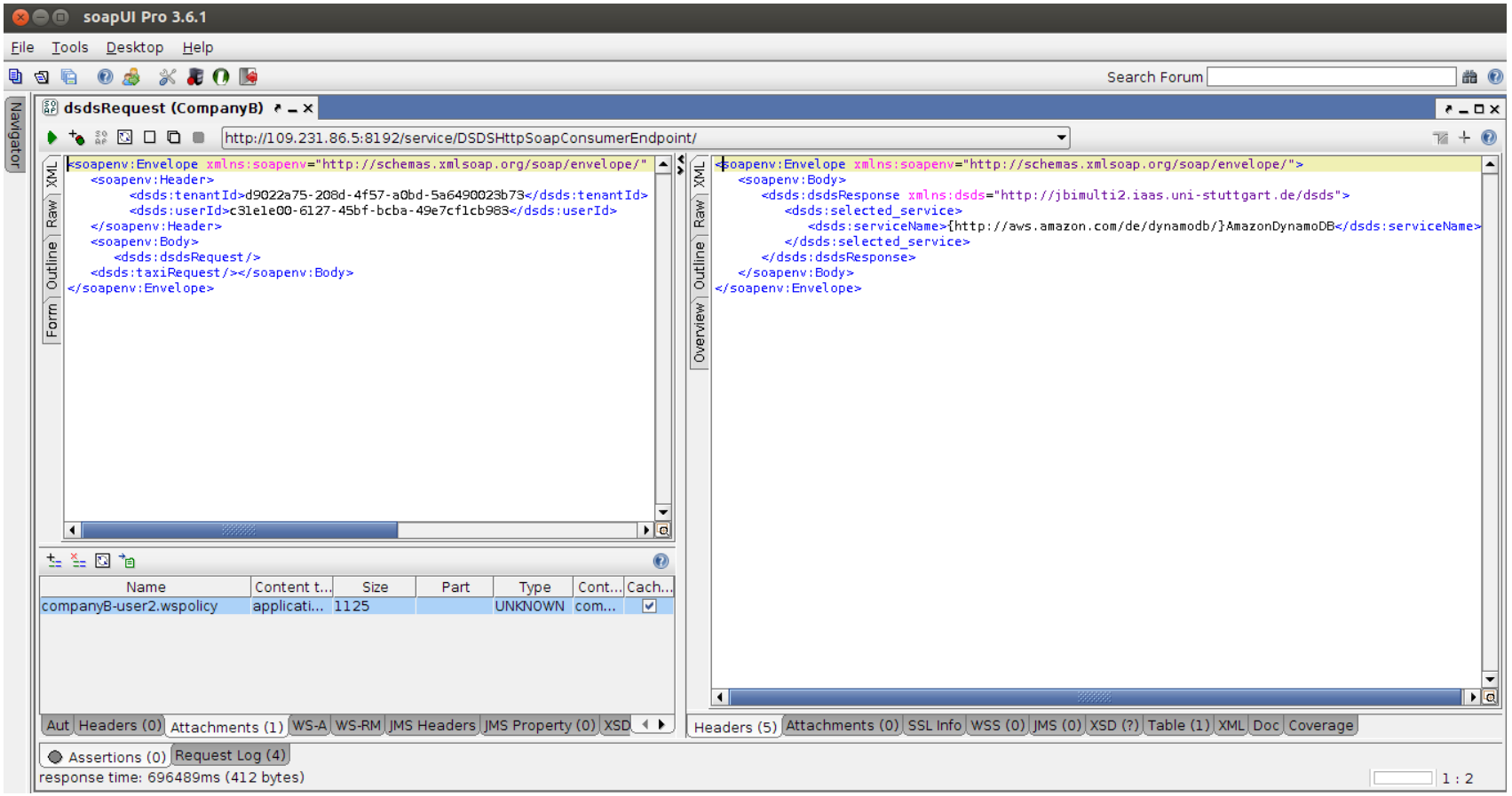

Figure 7.2.: DSDS Request executed by a tenant user with soapUI Pro 3.6.1 [SOAa]. On the left-hand side, top part is the SOAP request message and the bottom part is the message policy attachment, while on the right-hand side the corresponding SOAP response message is displayed. The response contains the discovered service.

In this case, a user of company A sends a request attaching the same policy with first user of company B. Tenant A (Company A) registered three Cloud data store services, which are Amazon RDS PostgreSQL with the policy in the Listing 7.4 as well as Amazon Simple DB and Xeround without policies. If you noticed, Amazon RDS PostgreSQL is available in both tenants (companies). As we stated, the user of tenant A and first user of tenant B sends request with the same policies (see Listings 7.3, 7.8). In doing so, if the data isolation was not provided, for the request of the user of company A, the Cloud data service Amazon RDS MySQL would be discovered. However, the result is Amazon RDS PostgreSQL because Amazon RDS MySQL is not registered by tenant A. So, we demonstrated multi-tenancy support.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsp:Policy xmlns:wsp="http://www.w3.org/ns/ws-policy"
3      xmlns:cdhs="http://iaas.uni-stuttgart/cdhs"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://www.w3.org/ns/ws-policy http://www.w3.org/2007/02/ws-policy.
           xsd">
6
7      <cdhs:scalability>
8          <wsp:Policy>
9
10             <cdhs:automationDegree cdhs:Priority="2">
11                 <wsp:Policy>
12                     <cdhs:manual/>
13                 </wsp:Policy>
14             </cdhs:automationDegree>
15
16             <cdhs:scalabilityType cdhs:Priority="1">
17                 <wsp:Policy>
18                     <cdhs:vertical/>
19                     <cdhs:horizontal/>
20                 </wsp:Policy>
21             </cdhs:scalabilityType>
22
23             <cdhs:degree cdhs:Priority="3">
24                 <wsp:Policy>
25                     <cdhs:limited/>
26                 </wsp:Policy>
27             </cdhs:degree>
28
29             <cdhs:timeToLaunchNewInstance cdhs:Milliseconds="2100000" cdhs:Priority="3"/>
30
31         </wsp:Policy>
32     </cdhs:scalability>
33
34     <!-- Availability -->
35     <cdhs:availability>
36         <wsp:Policy>
37             <cdhs:automaticFailover cdhs:Priority="2"/>
38             <cdhs:availabilityDegree Percentage="99.98" cdhs:Priority="1"/>
39         </wsp:Policy>
40     </cdhs:availability>
```

```
41
42      <!-- Security -->
43      <cdhs:security>
44          <wsp:Policy>
45              <cdhs:storageEncryption cdhs:Priority="1"/>
46              <cdhs:transferEncryption cdhs:Priority="1"/>
47              <cdhs:firewall cdhs:Priority="3"/>
48              <cdhs:authentication cdhs:Priority="1"/>
49              <cdhs:confidentiality cdhs:Priority="1"/>
50              <cdhs:integrity cdhs:Priority="1"/>
51          </wsp:Policy>
52      </cdhs:security>
53
54      <!-- Data Constraints -->
55      <cdhs:dataConstraints>
56          <wsp:Policy>
57              <cdhs:maxSizePerInstance cdhs:Gigabyte="1024" cdhs:Priority="2"/>
58              <cdhs:predefinedInstanceSize cdhs:Priority="2"/>
59          </wsp:Policy>
60      </cdhs:dataConstraints>
61
62      <!-- Cloud Computing -->
63      <cdhs:cloudComputing>
64          <wsp:Policy>
65              <cdhs:serviceModel cdhs:Priority="2">
66                  <wsp:Policy>
67                      <cdhs:paas/>
68                  </wsp:Policy>
69              </cdhs:serviceModel>
70
71              <cdhs:deploymentModel cdhs:Priority="1">
72                  <wsp:Policy>
73                      <cdhs:public/>
74                  </wsp:Policy>
75              </cdhs:deploymentModel>
76          </wsp:Policy>
77      </cdhs:cloudComputing>
78
79
80  </wsp:Policy>
```

Listing 7.8: Company A user policy.

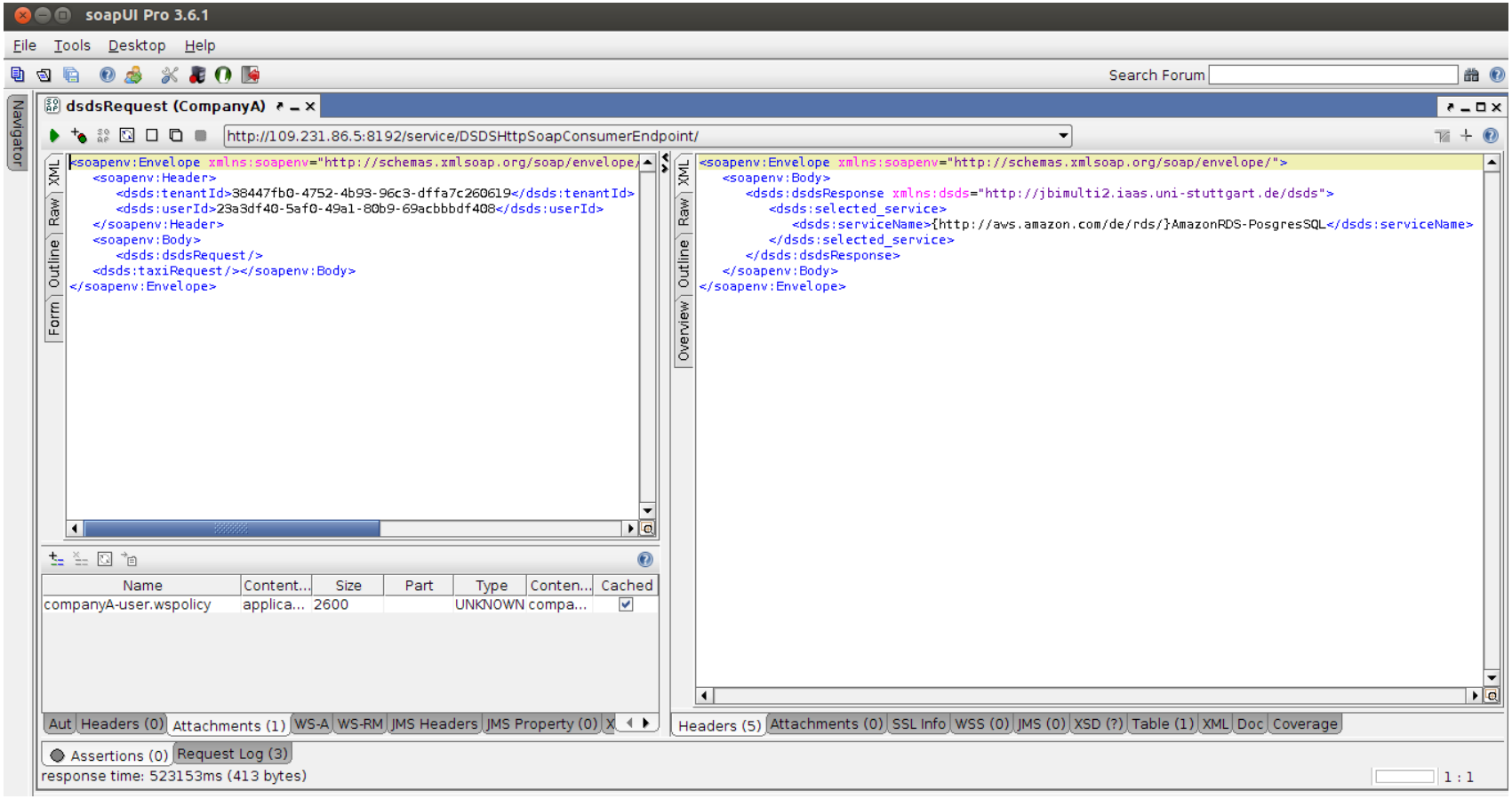

Figure 7.3.: DSDS Request executed by a tenant user with soapUI Pro 3.6.1 [SOAa]. On the left-hand side, top part is the SOAP request message and the bottom part is the message policy attachment. The response contains the discovered service.

8. Conclusion and Future Work

In Chapter 2 we have presented relevant fundamentals to this work. As a result of investigating related works to DSDS, we have introduced concepts for DSDS extension to the open-source ESB Apache ServiceMix that can be part of a PaaS platform. The emphasize was on conceiving good solutions to DSDS in ServiceMix, while maintaining multi-tenancy support and backward compatibility support. As a core part of this work, DSDS support is conceived that is applied discover and select a proper CDHS for the needs of the consumer. The main steps of SmxDSDS are Policy Matching, Post-Processing, Service Ranking, and Service Selection. For the policy matching, a new policy intersection algorithm is conceived that complies with the specification of WS-Policy framework and for the Post-Processing step we conceive solutions and created rules that can be used to post-process the domain-specific assertions and their attributes of intersected policy. We conceived Service ranking that is based on the resulting incompatible and not-fulfilled assertions from previous steps and according to them, it ranks services how much CDHS meet consumer needs. Here we only developed two different selection methods like First-Met and Random selection.

As CDHS providers should express their offer with uniform policy language, while CDHS consumers should specify their needs as policies relying on the compatible policy language. For this purpose, we analysed all existing policy languages and also used the student project [SR12] to determine which policy language is the most suitable for the expression of functional and non-functional properties of CDHS. Based on the outcome of the student project and other relevant works we decided to use WS-Policy as a basis to our new policy language. As a result, we extended WS-Policy to create a new WS-Policy Assertion Language that consists the assertions used to specify functional and non-functional properties of CDHS as policy documents. To provide a valid assertion set for the language, we investigated the functional and non-functional properties of all Cloud Data Hosting Solutions available.

As SmxDSDS retrieves the data for DSDS and multi-tenancy support from the databases outside ServiceMix, with the increase of requests, it is likely that the performance deteriorate. To prevent this, we proposed to build Cache mechanism in an intermediary component of SmxDSDS that is in charge of getting the data from external databases.

Based on the conceived concepts, we designed the system in Chapter 6 that describes design and technology solutions for the developed concepts to implement the prototype. Extensions to ServiceMix have been designed for DSDS support, where DSDS support is designed to be developed in a separate JBI Component (SE) allowing re-usability and support for backward compatibility. In contrast, the intermediary component, which is responsible for getting the data from databases and for their caching, is designed to be developed as an OSGi bundle. In addition, design extensions to the database schemes are made that enable a service and its policy handling. As for technology solutions, we designed to employ Apache Neethi

Framework [ANe12] for DSDS support to take advantage of the capabilities provided for the WS-Policy Framework. We also decided to use Ehcache [Ehc] open-source and standard-based cache library in the OSGi bundle.

A prototype that complies to these design and technology solutions has been implemented and validated in Chapter 7. In the validation, there are given two companies with their consumers that uses SmxDSDS. First, it is demonstrated how DSDS works for user preferences in case user requirements are not completely met. Second, the case when user requirements are completely met is demonstrated. Finally, the case that demonstrates the data isolation between tenants is validated.

For the future work, the WS-Policy Assertion Language and ranking in DSDS can be improved. Currently, priorities are considered to be set by consumers to only child assertions that should be improved for all assertion granularities. The policy validation is not done yet, so the policy validation algorithm can be created for validation of policy documents because we use WS-Policy construct <wsp:Policy>..</wsp:Policy> that is extremely extensible. New service selection methods can be added and configured, such as choosing services based on their load. We did not realize Web GUI either but we designed and specified it. Currently, the OSGi component is only developed to get data from Service Registry database. However, it can be easily extended for other databases like tenant registry, configuration registry. Cache utilization can also be improved by taking advantage of all the relevant capabilities of Ehcache.

Appendix A.

Interface Definitions

This chapter lists XSDs that WS-Policy Assertion language and Rules XML file must conform. Furthermore, there are given several policy documents of Cloud data stores that are used for validation.

A.1. WS-Policy Assertion Language Interface

This section illustrates XSD and syntax definition of WS-Policy Assertion language.

```
<wsp:Policy xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:cdhs="http://iaas.uni-stuttgart/cdhs"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!--  "..." is used to indicate a point of extensibility -->
  <!-- "?" means the element is optional (0 or 1) in terms of user spesification. It is user
       choice whether to specify the assertion or not -->

  <!-- Scalability -->
  <cdhs:scalability (cdhs:Priority="xs:integer")? ... >
    <wsp:Policy>
      <cdhs:automationDegree (cdhs:Priority="xs:integer")? ... >
        <wsp:Policy>
          (<cdhs:manual ... /> | <cdhs:automatic ... /> | (<cdhs:manual ... /> <
              cdhs:automatic ... />) | ...)
        </wsp:Policy>
      </cdhs:automationDegree>
      <cdhs:scalabilityType (cdhs:Priority="xs:integer")? ... >
        <wsp:Policy>
          (<cdhs:vertical ... /> | <cdhs:horizontal ... /> | (<cdhs:vertical ... /> <
              cdhs:horizontal ... />))
        </wsp:Policy>
      </cdhs:scalabilityType>
      <cdhs:degree (cdhs:Priority="xs:integer")?  ... >
        <wsp:Policy>
          (<cdhs:virtuallyUnlimited ... /> | <cdhs:limited cdhs:entityQuantity="xs:integer"
              cdhs:entityType="xs:string"/> | ...)
        </wsp:Policy>
      </cdhs:degree>?
```

```
26        <cdhs:timeToLaunchNewInstance cdhs:Milliseconds="xs:integer" (cdhs:Priority="
              xs:integer")? ... />?  <!-- time in the range (0 - N) minutes -->
27        <cdhs:automaticScalabilityCriterion (cdhs:Priority="xs:integer")? ... >
28          <wsp:Policy>
29            <cdhs:systemLoadCriteria ... />?
30            <cdhs:latencyCriteria ... />?
31            ...
32          </wsp:Policy>
33        </cdhs:automaticScalabilityCriterion>?
34        <cdhs:transferLimit cdhs:Gigabyte="xs:float" (cdhs:Priority="xs:integer")? ... />
35        ...
36      </wsp:Policy>
37    </cdhs:scalability>
38
39    <!-- Availability -->
40    <cdhs:availability  (cdhs:Priority="xs:integer")? ... >
41      <wsp:Policy>
42        <cdhs:replication (cdhs:Priority="xs:integer")? ... >
43          <wsp:Policy>
44            <cdhs:replicationType (cdhs:Priority="xs:integer")? ... >
45                <wsp:Policy>
46                  <cdhs:masterSlave  ... />?
47                  <cdhs:masterMaster  ... />?
48                  ...
49                </wsp:Policy>
50            </cdhs:replicationType>
51            <cdhs:replicationMethod (cdhs:Priority="xs:integer")? ...>
52              <wsp:Policy>
53                <cdhs:synchronous  ... />?
54                <cdhs:aynchrounous  ... />?
55                ...
56              </wsp:Policy>
57            </cdhs:replicationMethod>
58            <cdhs:replicationLocation (cdhs:Priority="xs:integer")? ... >
59              <wsp:Policy>
60                <cdhs:sameDC  ... />?
61                <cdhs:diffDCinSameRegion  ... />?
62                <cdhs:diffDCinDiffRegion  ... />?
63                ...
64              </wsp:Policy>
65            </cdhs:replicationLocation>
66          </wsp:Policy>
67        </cdhs:replication>?
68        <cdhs:automaticFailover (cdhs:Priority="xs:integer")? ... />?
69        <cdhs:availabilityDegree cdhs:Percentage="xs:float" (cdhs:Priority="xs:integer")? ...
              />  <!-- in the range of (0 - 100)% -->
70        ...
71      </wsp:Policy>
72    </cdhs:availability>
73
74    <!-- Recovery -->
75    <cdhs:recovery (cdhs:Priority="xs:integer")? ... >
76      <wsp:Policy>
77        <cdhs:disasterRecovery (cdhs:Priority="xs:integer")? ... />?
78        ...
```

```
79      </wsp:Policy>
80    </cdhs:recovery>
81
82    <!-- Security -->
83    <cdhs:security (cdhs:Priority="xs:integer")? ... >
84      <wsp:Policy>
85        <cdhs:storageEncryption (cdhs:Priority="xs:integer")? ... />?
86        <cdhs:transferEncryption (cdhs:Priority="xs:integer")? ... />?
87        <cdhs:firewall (cdhs:Priority="xs:integer")? ... />?
88        <cdhs:authentication (cdhs:Priority="xs:integer")? ... />?
89        <cdhs:confidentiality (cdhs:Priority="xs:integer")? ... />?
90        <cdhs:integrity (cdhs:Priority="xs:integer")? ... />?
91        <cdhs:authorization (cdhs:Priority="xs:integer")? ... />?
92        ...
93      </wsp:Policy>
94    </cdhs:security>
95
96    <!--  Privacy -->
97    <cdhs:privacy (cdhs:Priority="xs:integer")? ... >
98      <wsp:Policy>
99        <cdhs:secureDestruction  ... />?
100       <cdhs:completeDestruction  ... />?
101       <cdhs:supportedPrivacyPolicyLanguages  ... >
102         <wsp:Policy>
103           <cdhs:p3p  ... />?
104           <cdhs:xamcl  ... />?
105           <cdhs:epal  ... />?
106           <cdhs:wsPrivacy  ... />?
107           ...
108         </wsp:Policy>
109       </cdhs:supportedPrivacyPolicyLanguages>
110       ...
111     </wsp:Policy>
112   </cdhs:privacy>
113
114   <!-- Location -->
115   <cdhs:location (cdhs:Priority="xs:integer")? ... >
116     <wsp:Policy>
117       <cdhs:choice (cdhs:Priority="xs:integer")? ... >
118         <wsp:Policy>
119           <cdhs:yes ... >
120             <cdhs:geographicLocation cdhs:name="xs:string" (cdhs:Priority="xs:integer")? ...
                    />? <!-- Region/Country/Continent -->
121           </cdhs:yes>
122           |
123           <cdhs:no ... >
124             <wsp:Policy>
125               (<cdhs:singleLocation ... /> | <cdhs:automaticallySelected ... />)
126             </wsp:Policy>
127           </cdhs:no>
128         </wsp:Policy>
129       </cdhs:choice>? <!-- If choice is not specified, it means the CDHS is either located
              in a sigle place or selected automatically -->
130       <cdhs:cloudLocation (cdhs:Priority="xs:integer")? ... >
131         <wsp:Policy>
```

```
132           <onPremise ... />?
133           <offPremise ... />?
134         </wsp:Policy>
135       </cdhs:cloudLocation>
136       ...
137     </wsp:Policy>
138   </cdhs:location>
139
140   <!-- Constraints on the data to be migrated/created -->
141   <cdhs:dataConstraints (cdhs:Priority="xs:integer")? ... >
142     <wsp:Policy>
143       <cdhs:maxItemSize cdhs:Bytes="xs:integer" ... />?
144       <cdhs:rowSize cdhs:Bytes="xs:integer" ... />?
145       <cdhs:fileSize cdhs:Bytes="xs:integer" ... />?
146
147       <cdhs:maxDomainSize cdhs:Gigabyte="xs:float" ... />?
148       <cdhs:tableSize cdhs:Gigabytes="xs:float" ... />?
149       <cdhs:bucketSize cdhs:Gigabyte="xs:float" ... />?
150
151       <cdhs:maxItemNumberPerInstance cdhs:Number="xs:integer" ... />?
152       <cdhs:maxRowNumberPerInstance cdhs:Number="xs:integer" ... />?
153       <cdhs:maxFileNumberPerInstance cdhs:Number="xs:integer" ... />?
154
155       <cdhs:maxSizePerInstance cdhs:Gigabyte="xs:float" ... />?
156       <cdhs:predefinedInstanceSize/>?
157       ...
158     </wsp:Policy>
159   </cdhs:dataConstraints>
160
161   <!-- Interoperability -->
162   <cdhs:interoperability (cdhs:Priority="xs:integer")? ... >
163     <wsp:Policy>
164       <cdhs:dataPortability (cdhs:Priority="xs:integer")? ... >
165         <wsp:Policy>
166           <cdhs:import ... />?
167           <cdhs:export ... />?
168           <cdhs:importAndExport ... />?
169           <cdhs:oneWaySynchronization ... />?
170           <cdhs:twoWaySynchronization ... />?
171           ...
172         </wsp:Policy>
173       </cdhs:dataPortability>?
174       <cdhs:dataExchangeFormat (cdhs:Priority="xs:integer")? ...>
175         <wsp:Policy>
176           <cdhs:xml ... />?
177           <cdhs:json ... />?
178           <cdhs:proprietory ... />?
179           ...
180         </wsp:Policy>
181       </cdhs:dataExchangeFormat>
182       <cdhs:storageAccess (cdhs:Priority="xs:integer")? ... >
183         <wsp:Policy>
184           <cdhs:soa ... />?
185           <cdhs:rest ... />?
186           <cdhs:sql ... />?
```

```
187           <cdhs:proprietory ... />?
188           ...
189         </wsp:Policy>
190       </cdhs:storageAccess>
191       <cdhs:orm (cdhs:Priority="xs:integer")? ... >
192         <wsp:Policy>
193           <cdhs:jpa ... />?
194           <cdhs:jdo ... />?
195           <cdhs:linq  ... />?
196           ...
197         </wsp:Policy>
198       </cdhs:orm>
199       <cdhs:migrationAndDeploymentSupport (cdhs:Priority="xs:integer")? ... />?
200       <cdhs:supportedIDEs (cdhs:Priority="xs:integer")? ... >
201         <wsp:Policy>
202           <cdhs:eclipse ... />?
203           <cdhs:netBeans ... />?
204           <cdhs:visualStudio ... />?
205           <cdhs:intelliJIdea ... />?
206           ...
207         </wsp:Policy>
208       </cdhs:supportedIDEs>
209       <cdhs:developerSDKs (cdhs:Priority="xs:integer")? ... >
210         <wsp:Policy>
211           <cdhs:java ... />?
212           <cdhs:dotNet ... />?
213           <cdhs:php ... />?
214           <cdhs:ruby ... />?
215           <cdhs:python ... />?
216           <cdhs:standartSupport ... />? <!-- none needed -->
217           <cdhs:none ... />?
218           ...
219         </wsp:Policy>
220       </cdhs:developerSDKs>?
221       <cdhs:OS cdhs:name="xs:string" (cdhs:Priority="xs:integer")? ...>
222         <wsp:Policy>
223           <cdhs:windows ... >
224             <wsp:Policy>
225               <cdhs:windowsXP ... />?
226               <cdhs:windows7 ... />?
227               <cdhs:windows8 ... />?
228               <cdhs:windowsServer ... />?
229               <cdhs:windowsVista ... />?
230               <cdhs:windows98 ... />?
231             </wsp:Policy>
232           </cdhs:windows>
233           <cdhs:linux ... >
234             <wsp:Poicy>
235               <cdhs:ubuntu ... />?
236               <cdhs:fedora ... />?
237               <cdhs:achLinux ... />?
238               <cdhs:gentoo ... />?
239               <cdhs:openSUSE ... />?
240               <cdhs:slackware ... />?
241             </wsp:Poicy>
```

```
242           </cdhs:linux>
243           <cdhs:macOS ... >
244             <wsp:Policy>
245               <cdhs:macOSX ... />?
246               <cdhs:osX ... />?
247             </wsp:Policy>
248           </cdhs:macOS>
249         </wsp:Policy>
250       </cdhs:OS>
251       ...
252     </wsp:Policy>
253     ...
254   </cdhs:interoperability>
255
256   <!-- Compatibility -->
257   <cdhs:compatibility (cdhs:Priority="xs:integer")? ... >
258     <wsp:Policy>
259       <cdhs:productIncludingVersion (cdhs:Priority="xs:integer")? ... >
260         <wsp:Policy>
261           <cdhs:mySQL (cdhs:version="xs:string")?  ... />?
262             <cdhs:postgreSQL (cdhs:version="xs:string")?  ... />?
263           <cdhs:msSQL (cdhs:version="xs:string")? />?
264           <cdhs:db2 (cdhs:version="xs:string")? />?
265           <cdhs:oracle (cdhs:version="xs:string")? />?
266         </wsp:Policy>
267       </cdhs:productIncludingVersion>
268       ...
269     </wsp:Policy>
270   </cdhs:compatibility>
271
272   <!-- Storage -->
273   <cdhs:storage (cdhs:Priority="xs:integer")? ... >
274     <wsp:Policy>
275       <cdhs:accessability (cdhs:Priority="xs:integer")? ... >
276         <wsp:Policy>
277           <cdhs:vpn ... />? <!-- VPN (virtual private network) -->
278           ...
279         </wsp:Policy>
280       </cdhs:accessability>
281       <cdhs:storageType (cdhs:Priority="xs:integer")? ... >
282         <wsp:Policy>
283           (<cdhs:rdbms ... /> | <cdhs:noSQL ... /> | <cdhs:blobStore ... /> | <cdhs:cdn ...
                /> | ...)
284         </wsp:Policy>
285       </cdhs:storageType>
286       <cdhs:transactionSupport (cdhs:Priority="xs:integer")? ... >
287         <wsp:Policy>
288           <cdhs:acid ... />?
289           ...
290         </wsp:Policy>
291       </cdhs:transactionSupport>
292       ...
293     </wsp:Policy>
294   </cdhs:storage>
295
```

```
296
297   <!-- Performance -->
298   <cdhs:performance (cdhs:Priority="xs:integer")? ... >
299     <wsp:Policy>
300       <cdhs:predictable (cdhs:Priority="xs:integer")? ... >
301         <wsp:Policy>
302           <cdhs:responseTime cdhs:Milliseconds="xs:integer" ... />
303           ...
304         </wsp:Policy>
305       </cdhs:predictable>?
306       <cdhs:latency cdhs:Milliseconds="xs:integer" (cdhs:Priority="xs:integer")? ... />
307       <cdhs:throughput cdhs:value="xs:integer" (cdhs:Priority="xs:integer")? ... />
308       ...
309     </wsp:Policy>
310   </cdhs:performance>
311
312   <!-- CAP (Consistency, Availability, and Partitioning) property -->
313   <cdhs:cap (cdhs:Priority="xs:integer")? ... >
314     <wsp:Policy>
315       <cdhs:consistencyModel  (cdhs:Priority="xs:integer")? ... >
316         <wsp:Policy>
317           <cdhs:strongConsistency ... />?
318           <cdhs:weakConsistency ... />?
319           <cdhs:eventuallConsistency ... />?
320           ...
321         </wsp:Policy>
322         ...
323       </cdhs:consistencyModel>
324       <cdhs:availableInCaseOfPartitioning (cdhs:Priority="xs:integer")? ... />
325       ...
326     </wsp:Policy>
327   </cdhs:cap>
328
329   <!-- Flexibility -->
330   <cdhs:flexibility (cdhs:Priority="xs:integer")? ... >
331     <wsp:Policy>
332       <cdhs:schema (cdhs:Priority="xs:integer")? ... />?
333       <cdhs:schemaCustomizable (cdhs:Priority="xs:integer")? ... />?
334       ...
335     </wsp:Policy>
336   </cdhs:flexibility>
337
338   <!-- Cloud Computing -->
339   <cdhs:cloudComputing (cdhs:Priority="xs:integer")? ... >
340     <wsp:Policy>
341       <cdhs:serviceModel (cdhs:Priority="xs:integer")? ... >
342         <wsp:Policy>
343           (<cdhs:iaas ... /> | <cdhs:paas ... /> | <cdhs:saas ... />)
344         </wsp:Policy>
345       </cdhs:serviceModel>
346       <cdhs:deploymentModel (cdhs:Priority="xs:integer")? ... >
347         <wsp:Policy>
348           <cdhs:private ... />?
349           <cdhs:public ... />?
350           <cdhs:community ... />?
```

```
351           <cdhs:hybrid ... />?
352         </wsp:Policy>
353       </cdhs:deploymentModel>
354       ...
355     </wsp:Policy>
356   </cdhs:cloudComputing>
357
358   <!-- Management and Maintenance Effort -->
359   <cdhs:managementAndMaintenanceEffort (cdhs:Priority="xs:integer")? ...>
360     <wsp:Policy>
361       <cdhs:selfService ... />?
362       <cdhs:managedOrAutomated ... />?
363       ...
364     </wsp:Policy>
365   </cdhs:managementAndMaintenanctEffort>
366
367   <!-- Monitoring -->
368   <cdhs:monitoring (cdhs:Priority="xs:integer")? ... >
369     <wsp:Policy>
370       <cdhs:supportedKPIs (cdhs:Priority="xs:integer")? ... >
371         <wsp:Policy>
372           <cdhs:processingLoad ... />? <!-- CPU -->
373           <cdhs:networkDataTransfer ... />?  <!-- Network IO -->
374           <cdhs:diskDataTransfer ... />?  <!-- Disk IO -->
375           <cdhs:memoryLoad ... />?  <!-- RAM -->
376           ...
377         </wsp:Policy>
378       </cdhs:supportedKPIs>
379       ...
380     </wsp:Policy>
381   </cdhs:monitoring>
382
383   <!-- Backup -->
384   <cdhs:backup (cdhs:Priority="xs:integer")? ... >
385     <wsp:Policy>
386       <cdhs:backupInterval (cdhs:Priority="xs:integer")? ... >
387         <wsp:Policy>
388           <cdhs:none/>?
389           <cdhs:periodic cdhs:Minutes="xs:integer" ... />?
390           <cdhs:onDemand ... />?
391           ...
392         </wsp:Policy>
393       </cdhs:backupInterval>?
394       <cdhs:accessInterrupt (cdhs:Priority="xs:integer")? ... />?
395       <cdhs:numberOfBackupsKept (cdhs:Priority="xs:integer")? ... >
396         <wsp:Policy>
397           <cdhs:none/>?
398           <cdhs:backupNumber cdhs:number="xs:integer" ... />?
399           <cdhs:backupDays cdhs:days="xs:integer" ... />?
400           ...
401         </wsp:Policy>
402       </cdhs:numberOfBackupsKept>?
403       <cdhs:backupMethod  (cdhs:Priority="xs:integer")? ... >
404         <wsp:Policy>
405           <cdhs:snapshot ... />?
```

```
406           <cdhs:fileBackup ... />?
407           <cdhs:incrementalBackup ... />?
408           ...
409         </wsp:Policy>
410       </cdhs:backupMethod>?
411       ...
412     </wsp:Policy>
413   </cdhs:backup>
414
415     <!-- Multi-Tenancy -->
416   <cdhs:multiTenancy  (cdhs:Priority="xs:integer")? ... >
417     <wsp:Policy>
418       <cdhs:multiTenancyType (cdhs:Priority="xs:integer")? ... >
419         <wsp:Policy>
420           <cdhs:multipleInstances  ... />?
421           <cdhs:nativeMultiTenancy  ... />?
422           ...
423         </wsp:Policy>
424       </cdhs:multiTenancyType>
425       ...
426     </wsp:Policy>
427   </cdhs:multiTenancy>?
428
429 </wsp:Policy>
```

Listing A.1: Syntax of WS-Policy Assertion Language Schema.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:tns="http://iaas.uni-stuttgart/cdhs" targetNamespace="http://iaas.uni-
       stuttgart/cdhs" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3
4      <!-- Scalability -->
5      <xs:element name="scalability" type="tns:NestedPolicyType"/>
6
7          <xs:element name="automationDegree" type="tns:NestedPolicyType"/>
8
9              <xs:element name="manual" type="tns:SimplePolicyType"/>
10
11             <xs:element name="automatic" type="tns:SimplePolicyType"/>
12
13         <!-- Type of Scalability -->
14         <xs:element name="scalabilityType" type="tns:NestedPolicyType"/>
15
16             <xs:element name="vertical" type="tns:SimplePolicyType"/>
17
18             <xs:element name="horizontal" type="tns:SimplePolicyType"/>
19
20         <!-- Degree of Scalability -->
21         <xs:element name="degree" type="tns:NestedPolicyType"/>
22
```

```
23              <xs:element name="virtuallyLimited" type="tns:SimplePolicyType"/>
24
25              <xs:element name="limited">
26                  <xs:complexType>
27                      <xs:attribute name="entityQuantity" type="xs:integer"/>
28                      <xs:attribute name="entityType" type="xs:string"/>
29                      <xs:anyAttribute namespace="##any" processContents="lax"/>
30                  </xs:complexType>
31              </xs:element>
32
33          <!-- Time To Launch New Instance for Scalability -->
34          <xs:element name="timeToLaunchNewInstance">
35              <xs:complexType>
36                  <xs:attribute name="Milliseconds" type="xs:integer"/>
37                  <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/>
38                  <xs:anyAttribute namespace="##any" processContents="lax"/>
39              </xs:complexType>
40          </xs:element>
41
42          <!-- Automatic Criterion for Scalability -->
43          <xs:element name="automaticScalabilityCriterion" type="tns:NestedPolicyType"/>
44
45              <xs:element name="systemLoadCriteria" type="tns:SimplePolicyType"/>
46
47              <xs:element name="latencyCriteria" type="tns:SimplePolicyType"/>
48
49          <!-- Transfer Limit for Scalability -->
50          <xs:element name="transferLimit">
51              <xs:complexType>
52                  <xs:attribute name="Gigabyte" type="xs:float"/>
53                  <xs:anyAttribute namespace="##any" processContents="lax"/>
54              </xs:complexType>
55          </xs:element>
56
57      <!-- Availability -->
58      <xs:element name="availability" type="tns:NestedPolicyType"/>
59
60          <!-- Replication for Availability -->
61          <xs:element name="replication" type="tns:NestedPolicyType"/>
62
63             <!-- Type of Replication -->
64             <xs:element name="replicationType" type="tns:NestedPolicyType"/>
65
66                <xs:element name="masterMaster" type="tns:SimplePolicyType"/>
67
68                <xs:element name="masterSlave" type="tns:SimplePolicyType"/>
69
70              <!-- Type of Replication -->
71              <xs:element name="replicationMethod" type="tns:NestedPolicyType"/>
72
73                  <xs:element name="synchronous" type="tns:SimplePolicyType"/>
74
75                  <xs:element name="asynchronous" type="tns:SimplePolicyType"/>
76
77             <!-- Location of Replication -->
```

```
78            <xs:element name="replicationLocation" type="tns:NestedPolicyType"/>
79
80               <xs:element name="sameDC" type="tns:SimplePolicyType"/>
81
82               <xs:element name="diffDCinSameRegion" type="tns:SimplePolicyType"/>
83
84               <xs:element name="diffDCinDiffRegion" type="tns:SimplePolicyType"/>
85
86         <xs:element name="automaticFailover" type="tns:QNameAssertionType"/>
87
88         <xs:element name="availabilityDegree">
89             <xs:complexType>
90                 <xs:attribute name="Percentage" type="tns:percentageType"/>
91                 <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/>
92                 <xs:anyAttribute namespace="##any" processContents="lax"/>
93             </xs:complexType>
94         </xs:element>
95
96      <!-- Recovery -->
97      <xs:element name="recovery" type="tns:NestedPolicyType"/>
98
99         <xs:element name="disasterRecovery" type="tns:QNameAssertionType"/>
100
101     <!-- Security -->
102     <xs:element name="security" type="tns:NestedPolicyType"/>
103
104        <xs:element name="storageEncryption" type="tns:QNameAssertionType"/>
105
106        <xs:element name="transferEncryption" type="tns:QNameAssertionType"/>
107
108        <xs:element name="firewall" type="tns:QNameAssertionType"/>
109
110        <xs:element name="authentication" type="tns:QNameAssertionType"/>
111
112        <xs:element name="confidentiality" type="tns:QNameAssertionType"/>
113
114        <xs:element name="integrity" type="tns:QNameAssertionType"/>
115
116        <xs:element name="authorization" type="tns:QNameAssertionType"/>
117
118     <!-- Privacy -->
119     <xs:element name="privacy" type="tns:NestedPolicyType"/>
120
121        <xs:element name="secureDestruction" type="tns:QNameAssertionType"/>
122
123        <xs:element name="completeDestruction" type="tns:QNameAssertionType"/>
124
125        <xs:element name="supportedPrivacyPolicyLanguages" type="tns:NestedPolicyType"/>
126
127            <xs:element name="p3p" type="tns:SimplePolicyType"/>
128
129            <xs:element name="xamcl" type="tns:SimplePolicyType"/>
130
131            <xs:element name="epal" type="tns:SimplePolicyType"/>
132
```

```
133             <xs:element name="wsPrivacy" type="tns:SimplePolicyType"/>
134
135     <!-- Location -->
136     <xs:element name="location" type="tns:NestedPolicyType"/>
137
138         <xs:element name="choice" type="tns:NestedPolicyType"/>
139
140     <xs:element name="yes" type="tns:NestedPolicyType"/>
141
142         <xs:element name="no" type="tns:NestedPolicyType"/>
143
144             <xs:element name="singleLocation" type="tns:SimplePolicyType"/>
145
146             <xs:element name="automaticallySelected" type="tns:SimplePolicyType"/>
147
148             <xs:element name="geographicLocation">
149                 <xs:complexType >
150                     <xs:attribute name="name" type="xs:string"/>
151                     <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/
                            >
152                     <xs:anyAttribute namespace="##any" processContents="lax"/>
153                 </xs:complexType>
154             </xs:element>
155
156         <xs:element name="cloudLocation" type="tns:NestedPolicyType"/>
157
158             <xs:element name="onPremise" type="tns:SimplePolicyType"/>
159
160             <xs:element name="offPremise" type="tns:SimplePolicyType"/>
161
162     <!-- Data Constraints -->
163     <xs:element name="dataConstraints" type="tns:NestedPolicyType"/>
164
165        <xs:element name="maxItemSize">
166            <xs:complexType>
167                <xs:attribute name="Bytes" type="xs:integer"/>
168                <xs:anyAttribute namespace="##any" processContents="lax"/>
169            </xs:complexType>
170        </xs:element>
171
172        <xs:element name="rowSize">
173            <xs:complexType>
174                <xs:attribute name="Bytes" type="xs:integer"/>
175                <xs:anyAttribute namespace="##any" processContents="lax"/>
176            </xs:complexType>
177        </xs:element>
178
179        <xs:element name="fileSize">
180            <xs:complexType>
181                <xs:attribute name="Bytes" type="xs:integer"/>
182                <xs:anyAttribute namespace="##any" processContents="lax"/>
183            </xs:complexType>
184        </xs:element>
185
186        <xs:element name="maxDomainSize">
```

```
187             <xs:complexType>
188                 <xs:attribute name="Gigabyte" type="xs:float"/>
189                 <xs:anyAttribute namespace="##any" processContents="lax"/>
190             </xs:complexType>
191         </xs:element>
192 
193         <xs:element name="tableSize">
194             <xs:complexType>
195                 <xs:attribute name="Gigabyte" type="xs:float"/>
196                 <xs:anyAttribute namespace="##any" processContents="lax"/>
197             </xs:complexType>
198         </xs:element>
199 
200         <xs:element name="bucketSize">
201             <xs:complexType>
202                 <xs:attribute name="Gigabyte" type="xs:float"/>
203                 <xs:anyAttribute namespace="##any" processContents="lax"/>
204             </xs:complexType>
205         </xs:element>
206 
207         <xs:element name="maxItemNumberPerInstance">
208             <xs:complexType>
209                 <xs:attribute name="Number" type="xs:integer"/>
210                 <xs:anyAttribute namespace="##any" processContents="lax"/>
211             </xs:complexType>
212         </xs:element>
213 
214         <xs:element name="maxRowNumberPerInstance">
215             <xs:complexType>
216                 <xs:attribute name="Number" type="xs:integer"/>
217                 <xs:anyAttribute namespace="##any" processContents="lax"/>
218             </xs:complexType>
219         </xs:element>
220 
221         <xs:element name="maxFileNumberPerInstance">
222             <xs:complexType>
223                 <xs:attribute name="Number" type="xs:integer"/>
224                 <xs:anyAttribute namespace="##any" processContents="lax"/>
225             </xs:complexType>
226         </xs:element>
227 
228         <xs:element name="maxSizePerInstance">
229             <xs:complexType>
230                 <xs:attribute name="Gigabyte" type="xs:float"/>
231                 <xs:anyAttribute namespace="##any" processContents="lax"/>
232             </xs:complexType>
233         </xs:element>
234 
235         <xs:element name="predefinedInstanceSize" type="tns:QNameAssertionType"/>
236 
237     <!-- Interoperability -->
238     <xs:element name="interoperability" type="tns:NestedPolicyType"/>
239 
240          <xs:element name="dataPortability" type="tns:NestedPolicyType"/>
241 
```

```
242             <xs:element name="import" type="tns:SimplePolicyType"/>
243
244             <xs:element name="export" type="tns:SimplePolicyType"/>
245
246             <xs:element name="importAndExport" type="tns:SimplePolicyType"/>
247
248             <xs:element name="oneWaySynchronization" type="tns:SimplePolicyType"/>
249
250             <xs:element name="twoWaySynchronization" type="tns:SimplePolicyType"/>
251
252         <xs:element name="dataExchangeFormat" type="tns:NestedPolicyType"/>
253
254             <xs:element name="xml" type="tns:SimplePolicyType"/>
255
256             <xs:element name="json" type="tns:SimplePolicyType"/>
257
258             <xs:element name="proprietory" type="tns:SimplePolicyType"/>
259
260         <xs:element name="storageAccess" type="tns:NestedPolicyType"/>
261
262             <xs:element name="soa" type="tns:SimplePolicyType"/>
263
264             <xs:element name="rest" type="tns:SimplePolicyType"/>
265
266             <xs:element name="sql" type="tns:SimplePolicyType"/>
267
268         <xs:element name="orm">
269             <xs:complexType>
270                 <xs:attribute name="name" type="xs:integer"/>
271                 <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/>
272                 <xs:anyAttribute namespace="##any" processContents="lax"/>
273             </xs:complexType>
274         </xs:element>
275
276             <xs:element name="jpa" type="tns:SimplePolicyType"/>
277
278             <xs:element name="jdo" type="tns:SimplePolicyType"/>
279
280             <xs:element name="linq" type="tns:SimplePolicyType"/>
281
282         <xs:element name="migrationAndDeploymentSupport" type="tns:QNameAssertionType"/>
283
284         <xs:element name="supportedIDEs" type="tns:NestedPolicyType"/>
285
286             <xs:element name="eclipse" type="tns:SimplePolicyType"/>
287
288             <xs:element name="netBeans" type="tns:SimplePolicyType"/>
289
290             <xs:element name="visualStudio" type="tns:SimplePolicyType"/>
291
292             <xs:element name="intelliJIdea" type="tns:SimplePolicyType"/>
293
294         <xs:element name="developerSDKs" type="tns:NestedPolicyType"/>
295
296             <xs:element name="java" type="tns:SimplePolicyType"/>
```

```
297
298             <xs:element name="dotNet" type="tns:SimplePolicyType"/>
299
300             <xs:element name="php" type="tns:SimplePolicyType"/>
301
302             <xs:element name="ruby" type="tns:SimplePolicyType"/>
303
304             <xs:element name="python" type="tns:SimplePolicyType"/>
305
306             <xs:element name="standartSupport" type="tns:SimplePolicyType"/>
307
308         <xs:element name="OS" type="tns:NestedPolicyType"/>
309
310             <xs:element name="windows" type="tns:NestedPolicyType"/>
311
312                 <xs:element name="windowsXP" type="tns:SimplePolicyType"/>
313
314                 <xs:element name="windows7" type="tns:SimplePolicyType"/>
315
316                 <xs:element name="windows8" type="tns:SimplePolicyType"/>
317
318                 <xs:element name="windowsServer" type="tns:SimplePolicyType"/>
319
320                 <xs:element name="windowsVista" type="tns:SimplePolicyType"/>
321
322                 <xs:element name="windows98" type="tns:SimplePolicyType"/>
323
324             <xs:element name="linux" type="tns:NestedPolicyType"/>
325
326                 <xs:element name="ubuntu" type="tns:SimplePolicyType"/>
327
328                 <xs:element name="fedora" type="tns:SimplePolicyType"/>
329
330                 <xs:element name="archLinux" type="tns:SimplePolicyType"/>
331
332                 <xs:element name="gentoo" type="tns:SimplePolicyType"/>
333
334                 <xs:element name="openSUSE" type="tns:SimplePolicyType"/>
335
336                 <xs:element name="slackware" type="tns:SimplePolicyType"/>
337
338             <xs:element name="macOS" type="tns:NestedPolicyType"/>
339
340                 <xs:element name="macOSX" type="tns:SimplePolicyType"/>
341
342                 <xs:element name="osX" type="tns:SimplePolicyType"/>
343
344     <!-- Compatibility -->
345     <xs:element name="compatibility" type="tns:NestedPolicyType"/>
346
347         <xs:element name="productIncludingVersion" type="tns:NestedPolicyType"/>
348
349             <xs:element name="mySQL" type="tns:VersionType"/>
350
351             <xs:element name="postgreSQL" type="tns:VersionType"/>
```

```
352
353             <xs:element name="msSQL" type="tns:VersionType"/>
354
355             <xs:element name="db2" type="tns:VersionType"/>
356
357             <xs:element name="oracle" type="tns:VersionType"/>
358
359     <!-- Storage -->
360     <xs:element name="storage" type="tns:NestedPolicyType"/>
361
362         <xs:element name="accessability" type="tns:NestedPolicyType"/>
363
364             <xs:element name="vpn" type="tns:SimplePolicyType"/>
365
366         <xs:element name="storageType" type="tns:NestedPolicyType"/>
367
368             <xs:element name="rdbms" type="tns:SimplePolicyType"/>
369
370             <xs:element name="noSQL" type="tns:SimplePolicyType"/>
371
372             <xs:element name="blobStore" type="tns:SimplePolicyType"/>
373
374             <xs:element name="cdn" type="tns:SimplePolicyType"/>
375
376         <xs:element name="transactionSupport" type="tns:NestedPolicyType"/>
377
378             <xs:element name="acid" type="tns:SimplePolicyType"/>
379
380     <!-- Performance -->
381     <xs:element name="performance" type="tns:NestedPolicyType"/>
382
383         <xs:element name="predictable" type="tns:NestedPolicyType"/>
384
385             <xs:element name="responseTime">
386                 <xs:complexType>
387                     <xs:attribute name="Milliseconds" type="xs:integer"/>
388                     <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/
                            >
389                     <xs:anyAttribute namespace="##any" processContents="lax"/>
390                 </xs:complexType>
391             </xs:element>
392
393         <xs:element name="latency">
394             <xs:complexType>
395                 <xs:attribute name="Milliseconds" type="xs:integer"/>
396                 <xs:anyAttribute namespace="##any" processContents="lax"/>
397             </xs:complexType>
398         </xs:element>
399
400         <xs:element name="throughput">
401             <xs:complexType>
402                 <xs:attribute name="value" type="xs:float"/>
403                 <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/>
404                 <xs:anyAttribute namespace="##any" processContents="lax"/>
405             </xs:complexType>
```

```
406         </xs:element>
407
408     <!-- CAP -->
409     <xs:element name="cap" type="tns:NestedPolicyType"/>
410
411         <xs:element name="consistencyModel" type="tns:NestedPolicyType"/>
412
413              <xs:element name="strongConsistency" type="tns:SimplePolicyType"/>
414
415              <xs:element name="weakConsistency" type="tns:SimplePolicyType"/>
416
417              <xs:element name="eventuallConsistency" type="tns:SimplePolicyType"/>
418
419         <xs:element name="availableInCaseOfPartitioning" type="tns:QNameAssertionType"/>
420
421     <!-- Flexibility -->
422     <xs:element name="flexibility" type="tns:NestedPolicyType"/>
423
424       <xs:element name="schema" type="tns:QNameAssertionType"/>
425
426       <xs:element name="schemaCustomizable" type="tns:QNameAssertionType"/>
427
428     <!-- Cloud Computing -->
429     <xs:element name="cloudComputing" type="tns:NestedPolicyType"/>
430
431         <xs:element name="serviceModel" type="tns:NestedPolicyType"/>
432
433             <xs:element name="iaas" type="tns:SimplePolicyType"/>
434
435             <xs:element name="paas" type="tns:SimplePolicyType"/>
436
437             <xs:element name="saas" type="tns:SimplePolicyType"/>
438
439      <xs:element name="deploymentModel" type="tns:NestedPolicyType"/>
440
441             <xs:element name="private" type="tns:SimplePolicyType"/>
442
443             <xs:element name="public" type="tns:SimplePolicyType"/>
444
445             <xs:element name="community" type="tns:SimplePolicyType"/>
446
447             <xs:element name="hybrid" type="tns:SimplePolicyType"/>
448
449     <!-- Management and Maintenance Effort -->
450     <xs:element name="managementAndMaintenanceEffort" type="tns:NestedPolicyType"/>
451
452         <xs:element name="selfService" type="tns:SimplePolicyType"/>
453
454         <xs:element name="managedOrAutomated" type="tns:SimplePolicyType"/>
455
456     <!-- Monitoring -->
457     <xs:element name="monitoring" type="tns:NestedPolicyType"/>
458
459         <xs:element name="supportedKPIs" type="tns:NestedPolicyType"/>
460
```

```
461             <xs:element name="processingLoad" type="tns:QNameAssertionType"/>
462
463             <xs:element name="networkDataTransfer" type="tns:QNameAssertionType"/>
464
465             <xs:element name="diskDataTransfer" type="tns:QNameAssertionType"/>
466
467             <xs:element name="memoryLoad" type="tns:QNameAssertionType"/>
468
469     <!-- Backup -->
470     <xs:element name="backup" type="tns:NestedPolicyType"/>
471
472         <xs:element name="backupInterval" type="tns:NestedPolicyType"/>
473
474             <xs:element name="periodic">
475                 <xs:complexType>
476                     <xs:attribute name="Minutes" type="xs:integer"/>
477                     <xs:attribute name="Priority" type="tns:priorityInteger"/>
478                     <xs:anyAttribute namespace="##any" processContents="lax"/>
479                 </xs:complexType>
480             </xs:element>
481
482             <xs:element name="onDemand" type="tns:SimplePolicyType"/>
483
484         <xs:element name="accessInterrupt" type="tns:QNameAssertionType"/>
485
486         <xs:element name="numberOfBackupsKept" type="tns:NestedPolicyType"/>
487
488             <xs:element name="backupNumber">
489                 <xs:complexType>
490                     <xs:attribute name="number" type="xs:string"/>
491                     <xs:anyAttribute namespace="##any" processContents="lax"/>
492                 </xs:complexType>
493             </xs:element>
494
495             <xs:element name="backupDays">
496                 <xs:complexType>
497                     <xs:attribute name="number" type="xs:string"/>
498                     <xs:anyAttribute namespace="##any" processContents="lax"/>
499                 </xs:complexType>
500             </xs:element>
501
502         <xs:element name="backupMethod" type="tns:NestedPolicyType"/>
503
504             <xs:element name="snapshot" type="tns:SimplePolicyType"/>
505
506             <xs:element name="fileBackup" type="tns:SimplePolicyType"/>
507
508             <xs:element name="incrementalBackup" type="tns:SimplePolicyType"/>
509
510     <!-- Multi-Tenancy -->
511     <xs:element name="multiTenancy" type="tns:NestedPolicyType"/>
512
513         <xs:element name="multiTenancyType" type="tns:NestedPolicyType"/>
514
515              <xs:element name="multipleInstances" type="tns:SimplePolicyType"/>
```

```
516
517             <xs:element name="nativeMultiTenancy" type="tns:SimplePolicyType"/>
518
519     <!-- Global Elements -->
520
521     <xs:element name="none" type="tns:NoneType"/>
522
523     <xs:element name="type" type="tns:SimplePolicyType"/>
524
525     <!-- Schema Types -->
526
527     <xs:complexType name="NestedPolicyType">
528         <xs:sequence>
529             <!-- Actual content model is non-deterministic, hence wildcard. The following
                    shows intended content model: <xs:element ref="wsp:Policy" minOccurs="0" />
                    -->
530             <xs:any minOccurs="0" maxOccurs="unbounded" namespace="##other" processContents=
                    "lax"/>
531         </xs:sequence>
532         <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/>
533         <xs:anyAttribute namespace="##any" processContents="lax"/>
534     </xs:complexType>
535
536     <xs:complexType name="SimplePolicyType">
537         <xs:sequence/>
538         <xs:anyAttribute namespace="##any" processContents="lax"/>
539     </xs:complexType>
540
541     <xs:complexType name="QNameAssertionType">
542         <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/>
543         <xs:anyAttribute namespace="##any" processContents="lax"/>
544     </xs:complexType>
545
546     <xs:simpleType name="priorityInteger">
547         <xs:restriction base="xs:integer">
548             <xs:minInclusive value="0"/>
549             <xs:maxInclusive value="4"/>
550         </xs:restriction>
551     </xs:simpleType>
552
553     <xs:simpleType name="percentageType">
554         <xs:restriction base="xs:float">
555              <xs:minInclusive value="0"/>
556              <xs:maxInclusive value="100"/>
557         </xs:restriction>
558     </xs:simpleType>
559
560     <xs:complexType name="ValueUnitType">
561         <xs:attribute name="value" type="xs:integer"/>
562         <xs:attribute name="unit" type="xs:string"/>
563         <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/>
564         <xs:anyAttribute namespace="##any" processContents="lax"/>
565     </xs:complexType>
566
567     <xs:complexType name="PercentageRangeType">
```

```
568         <xs:attribute name="Min" type="tns:percentageType"/>
569         <xs:attribute name="Max" type="tns:percentageType"/>
570         <xs:attribute name="Priority" type="tns:priorityInteger" use="optional"/>
571         <xs:anyAttribute namespace="##any" processContents="lax"/>
572     </xs:complexType>
573 
574     <xs:complexType name="VersionType">
575         <xs:attribute name="version" type="xs:string"/>
576         <xs:anyAttribute namespace="##any" processContents="lax"/>
577     </xs:complexType>
578 
579     <xs:complexType name="NoneType">
580         <xs:attribute name="none"/>
581     </xs:complexType>
582 
583 </xs:schema>
```

Listing A.2: CDHS WS-Policy Assertion Language Schema.

A.2. Rules Interface

Rules created in a XML file must conform to the following XML schema:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <schema targetNamespace="http://iaas.uni-stuttgart/dsds" elementFormDefault="qualified"
3  xmlns="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://iaas.uni-stuttgart/dsds">
4  
5      <complexType name="tXPathRules">
6        <sequence>
7          <element name="Rule" minOccurs="1" maxOccurs="unbounded">
8            <complexType mixed="true">
9              <attribute name="assertionQName" type="QName" use="required"/>
10           </complexType>
11         </element>
12       </sequence>
13     </complexType>
14 
15     <element name="XPathRules" type="tns:tXPathRules"></element>
16 
17 </schema>
```

Listing A.3: Post-Processing Rules XML Schema.

A.3. Validation Policy Documents Interface

This section contains some policy documents that are used to validate the implemented prototype.

```
<wsp:Policy Name="http://iaas.uni-stuttgart/cdhs/GoogleCloudSQL" wsu:Id="GoogleCloudSQL"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/ns/ws-policy http://www.w3.org/2007/02/ws-policy.
        xsd"
    xmlns:wsp="http://www.w3.org/ns/ws-policy"
    xmlns:cdhs="http://iaas.uni-stuttgart/cdhs"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility
        -1.0.xsd">

    <!-- Security -->
    <cdhs:security wsp:Ignorable="true">
        <wsp:Policy>
            <cdhs:authentication wsp:Ignorable="true"/>
        </wsp:Policy>
    </cdhs:security>

    <!-- Data Constraints -->
    <cdhs:dataConstraints wsp:Ignorable="true">
        <wsp:Policy>
            <cdhs:maxSizePerInstance cdhs:Gigabyte="100" wsp:Ignorable="true"/>
        </wsp:Policy>
    </cdhs:dataConstraints>

    <!-- Compatibility -->
    <cdhs:compatibility wsp:Ignorable="true">
        <wsp:Policy>
            <cdhs:productIncludingVersion wsp:Ignorable="true">
                <wsp:Policy>
                    <cdhs:mySQL version="5.5.x"/>
                </wsp:Policy>
            </cdhs:productIncludingVersion>
        </wsp:Policy>
    </cdhs:compatibility>

    <!-- Storage -->
    <cdhs:storage wsp:Ignorable="true">
        <wsp:Policy>
            <cdhs:storageType wsp:Ignorable="true">
                <wsp:Policy>
                    <cdhs:rdbms wsp:Ignorable="true"/>
                </wsp:Policy>
            </cdhs:storageType>

            <cdhs:transactionSupport wsp:Ignorable="true">
                <wsp:Policy>
                    <cdhs:acid wsp:Ignorable="true"/>
```

```
                </wsp:Policy>
            </cdhs:transactionSupport>
        </wsp:Policy>
    </cdhs:storage>

    <!-- Monitoring -->
    <cdhs:monitoring wsp:Ignorable="true">
        <wsp:Policy>
            <cdhs:supportedKPIs wsp:Ignorable="true">
                <wsp:Policy>
                    <cdhs:networkDataTransfer wsp:Ignorable="true"/>
                </wsp:Policy>
            </cdhs:supportedKPIs>
        </wsp:Policy>
    </cdhs:monitoring>

</wsp:Policy>
```

Listing A.4: Google Cloud SQL Service Provider Policy.

```
<wsp:Policy Name="http://iaas.uni-stuttgart/cdhs/SQLDatabase" wsu:Id="SQLDatabase"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/ns/ws-policy http://www.w3.org/2007/02/ws-policy.
        xsd"
    xmlns:wsp="http://www.w3.org/ns/ws-policy"
    xmlns:cdhs="http://iaas.uni-stuttgart/cdhs"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility
        -1.0.xsd">

        <!-- Scalability -->
        <cdhs:scalability wsp:Ignorable="true">
            <wsp:Policy>
                <cdhs:scalabilityType wsp:Ignorable="true">
                    <wsp:Policy>
                        <cdhs:vertical/>
                        <cdhs:horizontal/>
                    </wsp:Policy>
                </cdhs:scalabilityType>
            </wsp:Policy>
        </cdhs:scalability>

        <!-- Availability -->
        <cdhs:availability wsp:Ignorable="true">
            <wsp:Policy>
                <cdhs:automaticFailover wsp:Ignorable="true"/>
            </wsp:Policy>
        </cdhs:availability>

        <!-- Security -->
```

```
29          <cdhs:security wsp:Ignorable="true">
30              <wsp:Policy>
31                  <cdhs:storageEncryption wsp:Ignorable="true"/>
32                  <cdhs:transferEncryption wsp:Ignorable="true"/>
33                  <cdhs:authentication wsp:Ignorable="true"/>
34                  <cdhs:confidentiality wsp:Ignorable="true"/>
35                  <cdhs:integrity wsp:Ignorable="true"/>
36              </wsp:Policy>
37          </cdhs:security>
38
39          <!-- Location -->
40          <cdhs:location wsp:Ignorable="true">
41              <wsp:Policy>
42                  <cdhs:cloudLocation wsp:Ignorable="true">
43                       <wsp:Policy>
44                           <cdhs:offPremise/>
45                       </wsp:Policy>
46                  </cdhs:cloudLocation>
47              </wsp:Policy>
48          </cdhs:location>
49
50          <!-- CAP -->
51          <cdhs:cap wsp:Ignorable="true">
52              <wsp:Policy>
53                  <cdhs:consistencyModel wsp:Ignorable="true">
54                      <wsp:Policy>
55                          <cdhs:strongConsistency/>
56                      </wsp:Policy>
57                  </cdhs:consistencyModel>
58
59                  <cdhs:availableInCaseOfPartitioning wsp:Ignorable="true"/>
60              </wsp:Policy>
61          </cdhs:cap>
62
63          <!-- Flexibility -->
64          <cdhs:flexibility wsp:Ignorable="true">
65              <wsp:Policy>
66                  <cdhs:schema wsp:Ignorable="true"/>
67                  <cdhs:schemaCustomizable wsp:Ignorable="true"/>
68              </wsp:Policy>
69          </cdhs:flexibility>
70
71          <!-- Backup -->
72          <cdhs:backup wsp:Ignorable="true">
73              <wsp:Policy>
74                  <cdhs:numberOfBackupsKept wsp:Ignorable="true">
75                      <wsp:Policy>
76                          <cdhs:backupDays days="35" wsp:Ignorable="true"/>
77                      </wsp:Policy>
78                  </cdhs:numberOfBackupsKept>
79
80                  <!-- no support for backup method -->
81
82              </wsp:Policy>
83          </cdhs:backup>
```

```
84
85 </wsp:Policy>
```

Listing A.5: SQL Database Service Provider Policy.

Bibliography

[4Ca] 4CaaSt – EU Project. http://www.4caast.eu/.

[ADy] Amazon DynamoDB. http://aws.amazon.com/dynamodb/.

[AGJ+08] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger. Multi-Tenant Databases for Software as a Service: Schema-Mapping Techniques. In Proc. SIGMOD International Conf. on Management of Data SIGMOD'08, pages 1195–1206. ACM, 2008.

[AJ10] S. T. AIMRUDEE JONGTAVEESATAPORN. Dynamic Service Selection Capability for Load Balancing in Enterprise Service Bus. In Proceedings of the 4th WSEAS, 2010, 2010.

[AMQ] The Apache Software Foundation. Apache ActiveMQ. http://activemq.apache.org/.

[AMV] The Apache Software Foundation. Apache Maven. http://maven.apache.org/.

[ANe12] Apache Neethi, 2012. http://ws.apache.org/neethi/.

[AOD] The Apache Software Foundation. Apache ODE (Orchestration Director Engine). http://ode.apache.org/.

[APA11a] The Apache Software Foundation. Apache Camel User Guide 2.10.0, 2011. http://camel.apache.org/manual/camel-manual-2.10.0.pdf.

[APA11b] The Apache Software Foundation. Apache Karaf Users' Guide 2.2.5, 2011. http://repo1.maven.org/maven2/org/apache/karaf/manual/2.2.5/manual-2.2.5.pdf.

[ARD] Amazon Relational Database Service (Amazon RDS). http://aws.amazon.com/rds/.

[ASD] Amazon Simple DB. http://aws.amazon.com/rds/.

[ASM] The Apache Software Foundation. Apache ServiceMix. http://servicemix.apache.org/.

[Bre12] E. Brewer. CAP twelve years later: How the "rules" have changed. In Proc. 28th IEEE Int Conf. on Data Engineering, 2012, pages 23–29, 2012.

[Cha04] D. A. Chappel. Enterprise Service Bus: Theory in Practice. O'Reilly Media, 2004.

[CHLP09] A. M. Colyer, H. Hildebrand, C. Leau, and A. Piper. Spring Dynamic Modules Reference Guide 1.2.1, 2009. http://static.springsource.org/osgi/docs/1.2.1/reference/pdf/spring-dm-reference.pdf.

[CLC12] CLOUDCYCLE, 2012. http://www.cloudcycle.org/.

[Ehc] Ehcache. http://ehcache.org/.

[EPA] Enterprise Privacy Authorization Language (EPAL 1.2). W3C Member Submission, http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/.

[Ess11] S. Essl. Extending an Open Source Enterprise Service Bus for Multi-Tenancy Support. Master's thesis, Institute of Architecture of Application Systems, University of Stuttgart, 2011.

[FB96] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, 1996. RFC 2045, http://www.ietf.org/rfc/rfc2045.txt.

[Goo] Google Cloud SQL. https://developers.google.com/cloud-sql/.

[HW03] G. Hohpe and B. Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, 2003.

[JBI05] Java Business Integration (JBI) 1.0, Final Release, 2005. JSR-208, http://jcp.org/aboutJava/communityprocess/final/jsr208/.

[JDB] PostgreSQL JDBC Driver. http://jdbc.postgresql.org/.

[Mar02] F. Marinescu. EJB Design Patterns: Advanced Patterns, Processes, and Idioms. John Wiley & Sons, Inc., 2002.

[MF06] F. R. Mariagrazia Fugini, Pierluigi Plebani. A User Driven Policy Selection Model. In Proc. 4th Int Conf. 2006, pages 427–433, 2006.

[MIG12] MIGRATE, 2012. http://www.migrate-it2green.de/.

[Muh11] D. Muhler. Extending an Open Source Enterprise Service Bus for Multi-Tenancy Support Focusing on Administration and Management, 2011. Institute of Architecture of Application Systems, University of Stuttgart, Diploma Thesis.

[Mul] Mule ESB. http://www.mulesoft.org/.

[NIS11] National Institute of Standards and Technology. The NIST Definition of Cloud Computing, 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[OAS04] OASIS. Web Services Resource Framework, 2004. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.

[OAS06] OASIS. Web Services Security (WS-Security) 1.1, 2006. http://www.oasis-open.org/standards/.

[OAS09] OASIS. WS-SecurityPolicy 1.3, 2009. http://www.oasis-open.org/standards/.

[OAS12] OASIS. Topology and Orchestration Specification for Cloud Applications (TOSCA), 2012. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca.

[OPG11] The Open Group. IBM Cloud Computing Reference Architecture 2.0, 2011. https://www.opengroup.org/cloudcomputing/uploads/40/23840/CCRA.IBMSubmission.02282011.doc.

[OSG11] OSGi Alliance. OSGi Service Platform: Core Specification Version 4.3, 2011. http://www.osgi.org/Download/Release4V43/.

[OWJ] OW2 Consortium. JOnAS: Java Open Application Server. http://wiki.jonas.ow2.org/.

[P3P] The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. W3C Working Group Note, http://www.w3.org/TR/P3P11/.

[PSQ] PostgreSQL. http://www.postgresql.org/.

[Rei] Rei Ontology Specifications, Ver 2.0. http://www.csee.umbc.edu/~lkagal1/rei/.

[SKLU11] S. Strauch, O. Kopp, F. Leymann, and T. Unger. A Taxonomy for Cloud Data Hosting Solutions. In Proc. 9th IEEE Int on Cloud and Green Computing, CGC 2011, pages 577–584, 2011.

[SOAa] SmartBear Software. soapUI. http://www.soapui.org.

[SOAb] SOA Work Group. www.opengroup.org/projects/soa/.

[SOA07] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), 2007. W3C Recommendation, http://www.w3.org/TR/2007/REC-soap12-part1-20070427/.

[SR12] A. G. u. H. T. Stefan Renner. Vergleich von Policy Sprachen zur Anwendung im Bereich des Cloud Computings, 2012. Student Project, Institute of Architecture of Application Systems, University of Stuttgart.

[WAS] Windows Azure SQL Database. http://msdn.microsoft.com/en-us/library/windowsazure/ee336279.aspx.

[WCL+05] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson. Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall, 2005.

[Wie07] A. Wiese. Konzeption und Implementierung von WS-Policy- und WSRF-Erweiterungen für einen Open Source Enterprise Service Bus, 2007. Institute of Architecture of Application Systems, University of Stuttgart, Diploma Thesis.

[WSD01] Web Services Description Language (WSDL) 1.1, 2001. W3C Note, http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[WSD06] WSDL 1.1 Binding Extension for SOAP 1.2, 2006. W3C Member Submission, http://www.w3.org/Submission/2006/SUBM-wsdl11soap12-20060405/.

[WSD07] Web Services Description Language (WSDL) Version 2.0, 2007. W3C Recommendation, http://www.w3.org/TR/wsdl20/.

[WSP07a] Web Services Policy 1.5 - Attachment, 2007. W3C Recommendation, http://www.w3.org/TR/ws-policy-attach/.

[WSP07b] Web Services Policy 1.5 - Framework, 2007. W3C Recommendation, http://www.w3.org/TR/2007/REC-ws-policy-20070904/.

[WSP07c] Web Services Policy 1.5 - Primer, 2007. W3C Note, http://www.w3.org/TR/ws-policy-primer.

[XAC] eXtensible Access Control Markup Language (XACML). OASIS, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[Xer] Xeround. http://xeround.com/.

[XPA99] XML Path Language (XPath), 1999. W3C Recommendation, http://www.w3.org/TR/xpath/.

[XPF03] XPointer Framework, 2003. W3C Recommendation, http://www.w3.org/TR/xptr-framework/.

[XSD04] XML Schema Part 1: Structures Second Edition, 2004. W3C Recommendation, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.

[YL04] L. Z. Z. Yutu Liu, Anne H. Ngu. QoS computation and policing in dynamic web service selection. In Proc. 13th Int WWW Conf. WWW Alt. '04, pages 66–73, 2004.

All links were last followed on November 29, 2012.

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

Stuttgart, November 30, 2012 _____

(Mansur Uralov)