CARL VON OSSIETZKY universität OLDENBURG

Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

# Lightweight Semantic-enabled Enterprise Service-Oriented Architecture

Dissertation

Submitted in fulfillment of the requirements for the degree of

**Doktors der Ingenieurwissenschaften (Dr.-Ing.)**

Submitted by

**Dipl.-Inform. Tariq Mahmoud**

April, 2013
Oldenburg, Germany

| | |
|---|---|
| **First Supervisor:** | Prof. Dr.-Ing. Jorge Marx Gómez |
| **Second Supervisor:** | Prof. Dr. Klaus Turowski |
| **Place of the Disputation:** | Carl von Ossietzky University of Oldenburg |
| **Disputation Date:** | 05.07.2013 |

## Acknowledgements

The years I have spent doing my PhD have come to an end. It is now the time for me to deliver a few kind words to the people who have strongly encouraged and supported me during that time.

At the beginning I would like to give my deepest heartfelt thanks to my parents, who shaped and formed my life, they bestowed in me all the motivation I needed to pursue my doctorate. Words can’t covey the credit they deserve. All I can do is to say thanks from the deepest chambers of my heart to my dear Souhel, Nada, Housam and Ayham.

I would also like to take this time to thank my first supervisor Prof. Dr.-Ing. Jorge Marx Gómez for giving me the freedom needed to conduct my research while also being a remarkable supervisor. Thanks also go to my second supervisor Prof. Dr. Klaus Turowski and the PhD examination committee members Prof. Dr.-Ing. Oliver Theel and Dr. Ute Vogel. Last but not least, I’d like to thank all the members of our Very Large Business Application (VLBA) group. I believed we formed a very good team, which always made the work more enjoyable. Thanks to all of you for your help, I achieved so much in my time spent at the VLBA group.

It has been also a pleasure for me to work with the students at the University of Oldenburg while supervising their research in bachelors’ and masters’ works.

I’ve spent so much time working hard in delivering this dissertation. This has heavily influenced my private life. I highly appreciate the support of my friends and family during that time. Thanks to all of you again.

At the end, it is hard to use any language to represent my gratitude to the people who have helped me. You all know how much I love and respect you. Thanks again for your patience, feedback and support.

Tariq Mahmoud

Oldenburg, July 2013

# Zusammenfassung

Heutzutage wird es für Anbieter am Markt immer wichtiger Ihre Produkte und ihre Software auf die Bedürfnisse Klein- und mittlerer Unternehmen (KMU) zuzuschneiden, da deren Marktanteile enorm gestiegen sind. Dies gilt ebenso für die Erfüllung der Anforderungen aus dem Business-to-Business (B2B) Bereich, die zur wichtigen Herausforderung wird.

Betrachtet man das Thema Systemintegration neben anderen wichtigen Marktfaktoren, scheinen Webservices eine der wichtigen Technologien für die Lösung von Integrationsproblemen zu sein. Service-orientierte Architekturen und SOA-fähige Systeme bieten leistungsstarke Applikationen für den KMU-Markt. Jedoch haben bestehende Architekturen im Unternehmensweb viele Nachteile wie enorm große Datenmengen, eine wachsende Zahl unverbundener Systeme sowie einen Mangel an Interoperabilität. Weiterhin fehlen den SOA-basierten Systemen semantische Dokumentationen für die Webservice-Schnittstellen.

Semantische Webservices bieten Methoden, die (teil-) automatische Suche, Komposition und Ausführung von Webservices zu vereinfachen. Jedoch erscheinen diese neu entstehenden semantischen Technologien aufgrund der Komplexität für nicht-technische Nutzer ungeeignet zu sein, die Kundenanfragen und Webserviceleistungen semantisch zusammenzuführen.

Diese Dissertation liefert eine semantische Webservice-basierte Referenzarchitektur, die im Wesentlichen auf der Idee basiert, semantische Ausdrücke mit dem leichtgewichtigen Resource Description Framework (RDF) auf Webservices anzuwenden, um effiziente Unternehmenssystemlösungen zu erhalten. In der Dissertation wird diese Referenzarchitektur “Lightweight Semantic-enabled Enterprise Service-Oriented Architecture (SESOA)” genannt. Sie vereint Geschäftsprozesse und SOA-Konzepte, um eine agile und flexible Unternehmenslösung zu schaffen, deren Geschäftsfunktionalitäten im Wesentlichen auf Webservices basieren. Darüber hinaus ist das wesentliche Ziel dieser Arbeit, das gesamte Unternehmensweb zu einem Medium zu gestalten, in dem die Bedeutung der verwalteten Informationen automatisch verstanden und verarbeitet werden.

## Abstract

Nowadays, it becomes more and more essential for the vendors in the markets to tailor their products and software to suit the Small and Medium Enterprises (SME) section since their market share has been enormously raised. The issues related to Business-to-Business (B2B) environment are becoming important challenges to be considered in such area as well.

Talking about system integration among the major market business factors, Web Services seem to be one of the powerful technologies to solve the integration problems. Service-Oriented Architecture and SOA-enabled systems provide powerful applications to be utilized in the SME market. However, the existing architecture of the enterprise's Web has many drawbacks like enormous volumes of unstructured data, growing number of disconnected systems besides the lack of interoperability. Moreover, SOA-based systems lack the semantic documentation of the Web Service interfaces.

Semantic Web Services provide methods to ease the (semi-) automatic discovery, composition, and execution of Web Services. However, these new emerging semantic technologies seem to be inaccurate to be used in terms of semanticizing the consumer requests and the capabilities of the Web Services besides its complexity when non-technical skilled staff is involved.

This dissertation presents a semantic Web Service-based reference architecture that relies mainly on the idea of applying lightweight Resource Description Framework (RDF) semantic statements to Web Services to have an efficient enterprise system solution. In this dissertation, the reference architecture is called "Lightweight Semantic-enabled Enterprise Service-Oriented Architecture (SESOA)". It merges both business processes and SOA concepts to provide an agile and flexible enterprise solution in which business functionalities are mainly implemented using Web Services. Moreover, the ultimate goal behind this work is to upgrade the entire enterprise Web into a medium where the meaning of its associated information can be automatically understood and processed.

## Table of Contents

## List of Abbreviations and Acronyms

| | |
|---|---|
| AES | Advanced Encryption Standard |
| ARP | Another RDF Parser |
| ASCII | American Standard Code for Information Interchange |
| B2B | Business-to-Business |
| BI | Business Intelligence |
| BPM | Business Process Management |
| BPMN | Business Process Management Notation |
| BPMS | Business Process Management Systems |
| BPR | Business Process Reengineering |
| CEMIS | Corporate Environmental Management Information Systems |
| CMS | Conceptual Models for Services |
| CORBA | Common Object Request Broker Architecture |
| CRM | Customer Relationship Management |
| DAO | Data Access Object |
| DBMS | Database Management System |
| DCE | Distributed Computing Environment |
| DCOM | Distributed Component Object Model |
| DEC | Digital Equipment Corporation |
| DES | Data Encryption Standard |
| DSRM | Design Science Research Methodology |
| EA | Enterprise Architecture |
| EAI | Enterprise Application Integration |
| EJB | Enterprise Java Beans |
| ERP | Enterprise Resource Planning |
| EPA | Evaluation Processing Authority |
| EPI | Environmental Performance Indicators |
| ESB | Enterprise Service Bus |
| FERP | Federated Enterprise Resource Planning |
| FTP | File Transfer Protocol |
| GUID | Globally Unique Identifier |
| GUI | Graphical User Interface |
| HTML | Hypertext Markup Language |

| | |
|---|---|
| HTTP | Hypertext Transfer Protocol |
| IDL | Interface Definition Language |
| IIS | Internet Information Service |
| IS | Information System |
| IRI | Internationalized Resource Identifier |
| ISO | International Standards Organization |
| LINQ | Language Integrated Query |
| MAUT | Multi-Attribute Utility Theory |
| OASIS | Organization for the Advancement of Structured Information Standards |
| OEPI | Organizations' Environmental Performance Indicators |
| OLAP | Online Analytical Processing |
| OLTP | Online Transaction Processing |
| OMG | Object Management Group |
| ORB | Object Request Broker |
| OWL | Web Ontology Language |
| OWL-S | Web Ontology Language for Web Services |
| PDP | Policy Decision Point |
| PEP | Policy Enforcement Point |
| REST | Representational State Transfer |
| RDF | Resource Description Framework |
| RDFS | Resource Description Framework Schema |
| RIF | Rule Interchange Format |
| RPC | Remote Procedure Call |
| RSA | Rivest Shamir Adelman |
| SAWSDL | Semantic Annotations for WSDL and XML Schema |
| SESOA | Semantic-enabled Enterprise Service-Oriented Architecture |
| SCA | Service Composition Architecture |
| SCM | Supply Chain Management |
| SCXML | State Chart XML |
| SDS | Security Decision Service |
| SES | Security Enforcement Service |
| SHA | Secure Hash Algorithm |
| SLA | Service Level Agreement |

| | |
|---|---|
| SME | Small and Medium Enterprises |
| SMTP | Simple Mail Transfer Protocol |
| SOA | Service-oriented architecture |
| SOAP | Simple Object Access Protocol |
| SPARQL | SPARQL Protocol and RDF Query Language |
| SSL | Secure Socket Layer |
| SQL | Structured Query Language |
| SWSF | Semantic Web Services Framework |
| SWSL | Semantic Web Services Language |
| SWSO | Semantic Web Service Ontology |
| TLS | Transport Layer Security |
| TPL | Task Parallel Library |
| UDDI | Universal Description, Discovery and Integration |
| UML | Unified Modeling Language |
| URI | Universal Resource Identifier |
| URL | Uniform Resource Locator |
| W3C | World Wide Web Consortium |
| WCF | Windows Communication Foundation |
| WF | Workflow Foundation |
| WfMC | Workflow Management Coalition |
| WfMS | Workflow Management System |
| WS-BPEL | Web Services Business Process Execution Language |
| WS-CDL | Web Service Choreography Description Language |
| WSCI | Web Service Choreography Interface |
| WSDL | Web Services Description Language |
| WSMF | Web Service Modeling Framework |
| WSML | Web Service Modeling Language |
| WSMO | Web Service Modeling Ontology |
| WWW | WorldWideWeb |
| XML | Extensible Markup Language |

## List of Figures

## List of Tables

# 1 Introduction

It is very rare in the last few years to find an enterprise that is based only on one information system, rather it has to have bundle of systems that are specialized for different organizational sections. This is besides the fact that the utilization of information technology is playing an essential role in the competitiveness between enterprises. Nowadays, the duration of the production of goods and service provisioning[1] has been notably decreasing through the increment of automation procedure in almost all the business work processes. This automation procedure reflects the mapping of the enterprise's business processes to be realized as services supplied over a distributed network.

Furthermore, since the digital market is growing drastically and the vendors are controlling this market, there is a need for new approaches to deal with that (Brehm et al., 2008, p. 26) side by side with one of the main troubles that the information systems are facing that is integration. In other words, software complexity and integration are important factors to be considered while developing software systems. To handle complexity and integration factors, many frameworks and approaches have been designed and implemented for enterprise information systems like Enterprise Application Integration (EAI) (Linthicum, 2000) and Service-Oriented Architecture (SOA) (Erl, 2005).

The main business driver behind EAI is to achieve process integration among third-party applications[2] and legacy systems to make the number of adapters needed to connect systems as less as possible (Haller, Gomez, & Bussler, 2005). SOA and more precisely enterprise SOA outlines a set of services to align business with IT. These services are made available for all key players inside an enterprise or even outside it. These services conjointly handle all enterprise goals and business processes (Rosen et al., 2008, p. 78). Enterprise SOA is in turn providing integration between multiple lines of business[3] by offering standardized services.

This work is devoted to the problems related to information systems complexity and integration. It examines the extent of the existing integration approaches to provide an alternative solution that balances between the business technical functionalities and structural complexity that might accompany its information system(s).

[1] In economics, service provisioning means that the end user does not exclusively own or benefit of a purchased product unless he/she pays for it or the service is provided for public use.

[2] Third-party applications refer to those applications that are delivered by companies other than the company that is using them.

[3] "Line of business" refers to an enterprise's internal organization unit. It is in contrast with the term "industry" that denotes to an external enterprise that is competing in a similar market.

## 1.1 Motivation

Recently, the market share of Small and Medium Enterprises (SME)[4] section has enormously raised (Armario, Ruiz, & Armario, 2008). Therefore, it becomes more essential for the vendors to tailor their products and software to suit SMEs in such market. SOA is classified as a crucial architectural style that aligns business with information technology at different business levels - especially SMEs. This encouraged several companies to transfer their entire information infrastructures towards the Web platform, offering unified and standardized access for their customers, suppliers and employees to the information and services they offer (Brehm et al., 2008).

Enriching data with semantics made the major shift towards Semantic Web[5] (Berners-Lee, 1998). As mentioned by (Cardoso, Hepp, & Lytras, 2007, p. 4), Semantic Web can ease the integration and interoperability of intra- and inter-business processes and systems. It also establishes global infrastructures to share data and documents making the process of information searching and reusing easier. Web Service-enabled SOA systems represent powerful applications to be utilized in the SME market (Bieberstein et al., 2005). By annotating the Web Services supplied to these systems with semantics, the meaning of information is defined to provide more powerful and efficient discovery, selection and invocation of the used services.

The connection between enterprise information systems, Semantic Web and Web Service-enabled SOA systems are the main concepts related to this work. Moreover, the integration issues between heterogeneous and monolithic business applications and the complexity accompanied the semantic conceptual frameworks for Web Services and the lightweight service description extensions form the motivation of this work to deliver a lightweight semantic SOA reference architecture.

## 1.2 Problem Definition

Almost every enterprise has one or more applied information systems serving for its different organizational sections. This indicates that one of the most problematic issues that the businesses have to overcome is the software integration. The integration approaches include EAI software, Enterprise Service Bus (ESB) implementations, service-oriented implementation, message-oriented middleware, message brokers, etc. In other words, the main question that can be asked here is how to handle the data exchange among heterogeneous dissimilar information systems?

[4] It is also known as: SMEs, Small and Medium Businesses or SMBs.

[5] Semantic Web is merely an expansion of the conventional WorldWideWeb (a.k.a. WWW or W3) in which the semantics - meaning - of Web information and services are defined, processed and understood by both people and machines to use the Web content (Berners-Lee, Hendler, & Lassila, 2001).

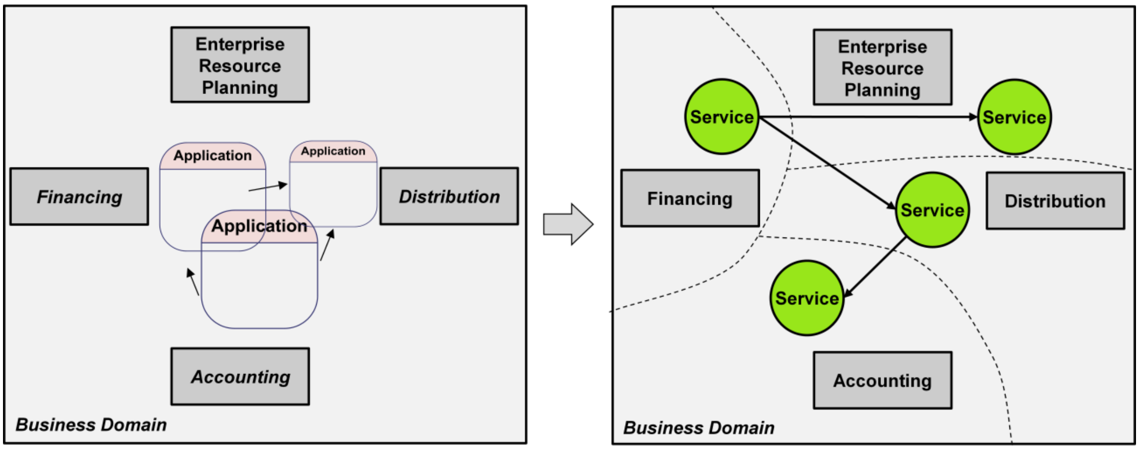


**Fig. 1.1:** The Problem of Existing Enterprise Systems

This question is reflected in Figure 1.1. The main problems that can be observed in this figure are:

- The problem of redundancy: Nearly in each business domain, many enterprise systems are employed to achieve enterprise goals. The main problem in such contexts is that business functionality is duplicated in each enterprise application that requires it (see the top left side of the figure above). This raises the problem of having monolithic enterprise system with many application silos. EAI leverages these application silos with the risk of data and function redundancy besides the overlapping of resources and providers. Moreover, the integrated architecture is always bounded to EAI vendors. This results in having a limited set of business processes and as a consequence less consumers.

- The problem of lacking semantic descriptions: Any business domain can be split into different organizational units. Based on SOA concept, different providers supply services to these units (see the top right side of the Figure 1.1). This indicates that SOA structures any business domain and its systems as a set of capabilities offered as services. Services in this context virtualize how such capabilities are performed, where and by whom the resources are provided. This enables multiple providers and consumers to participate together in shared business activities. The semantic descriptions of services in most of the SOA-based systems are not taken into consideration. Many approaches concentrate on this issue either by providing heavyweight frameworks or lightweight semantic descriptions. However, both types focus on providing semantics first with less interest on the business perspectives.

To addresses these problems, a reference architecture have been developed. It offers a service architecture in which services are grouped based on the organizational unit to which they belong. The lightweight semantic annotation is made between these services and groups. The resulted artifact from this architecture is applied to an accompanying business case to prove its applicability in different business environments.

### 1.2.1 General Problem Definition

Traditional Enterprise Architecture (EA) concepts in the EAI domain focus on the ex-post integration of application interfaces by pipelining different middleware technologies like message queuing or remote method invocations (Bussler, 2003a; Hohpe & Woolf, 2003; Mahmoud & Marx Gómez, 2008a). However, EAI systems lack in common the difficulty of applying semantic integration to the systems since no formal interface data definition exists (Bussler, 2003b). This requires that the engineers who are going to integrate the enterprise application systems need to have an in depth knowledge regarding the meaning of the low-level data in order to apply a semantically correct integration (Haller, Gomez, et al., 2005).

On the other hand, the emerging trends in the EAI market as for example SOA and ESB[6] promote the Web Service technology to advertise services based on standards claiming to solve the integration problems. Before referring to the problems of Web Service-enabled SOA systems, it is necessary to introduce the lifecycle of Web Services. The major phases in the service lifecycle as described in the Web Services conceptual architecture are: build, deploy, run and manage (Kreger, 2001). These phases deal with service discovery, selection, invocation, composition, and monitoring. The build phase deals with the development and testing of a Web Service. The publication of the service interface takes place in the deploy phase. The service is then made available to be invoked in the run phase. Finally, managing services is covered in the manage phase.

Service discovery and publication usually come together. Providers implement Web Services and publish their descriptions (interfaces) in one or more service registries. Consumers can then discover these published service descriptions to select a proper service to invoke. Service selection is based on a set of criteria like binding information, performance, availability, load balancing, quality of service, etc. As soon as a Web Service is selected, the service consumer can invoke it and call its functions.

Since the implementation of a Web Service is considered as a software module, new Web Services can always be composed from the existing ones. Some composition approaches are mentioned in Section 6.1.3.4. Finally, monitoring and management of enterprise business services are highly needed to ensure that the communications with these services do not introduce delay and denial failures. If these failures might occur, proper handling is always recommended.

Based on this short introduction of the Web Service lifecycle, the accompanied problems to the Web Service-enabled SOA systems can be defined. Relying mainly on Web Service standards, these systems lack the formal semantic documentation of their under-

[6] Enterprise Service Bus or ESB is a standard-based integration platform that merges several techniques like Web Services, data transformation and intelligent routing to integrate several applications via event-driven SOA across a multiprotocol service bus (Chappell, 2004, pp. 1, 2).

lying Web Service interfaces and data structures. This indicates that the already existing syntactical problem in their antecedent EAI technologies still exists. This comes with some other accompanying drawbacks like enormous volumes of unstructured data, growing number of disconnected systems, and the lack of interoperability as well (Hu et al., 2008, p. 589).

Furthermore, existing SOA-based systems lack the (semi)-automatic service discovery, (semi)-automatic service composition, data and process interoperability besides the information sharing, finding, extraction, interpreting, maintaining, and representation (Mahmoud & Marx Gómez, 2010, p. 2). Annotating Web Services in SOA-based systems with semantics results in the so-called Semantic Web Services (Studer, Grimm, & Abecker, 2007; McIlraith, Son, & Zeng, 2001). The main purpose behind Semantic Web Services research paradigm is to ease the automation process of Web Services discovery, selection, invocation, composition, and monitoring in a distributed and open environment[7] (Payne & Lassila, 2004, p. 14).

One of the promises of the Semantic Web Services is to overcome the heterogeneity problems of the enterprise Web resources at both data and process levels by providing methods to ease the (semi-) automatic discovery, composition, and execution of Web Services (Mahmoud & Marx Gómez, 2008b, p. 791; Mahmoud, 2009, p. 475). On the one hand, it is complex to use these new emerging semantic techniques in terms of applying semantics to Web Service capabilities and their consumer requests (they provide semantics first) (cf. Sabou, 2005). On the other hand, the complexity of these semantic techniques is an important issue to be considered when non-technical skilled staff is going to be involved.

From process orientation perspective, Web Service-enabled business processes ease the interaction of enterprises with markets, competitors, suppliers and customers. They enable enterprise-level and core cross-sectional business functionalities and support both intra- and inter-organizational workflows. They are mainly composed of Web Services as underlying activities. As described in (Cardoso et al., 2007; van der Aalst, 2009), The main challenges of Web Service-enabled business processes are autonomy and heterogeneity that come from the complex interactions rules in B2B and e-commerce in general. Business demands like efficient discovery, composition, etc. require dynamic nature of business interactions. Last but not least scalability needs to be taken into account while dealing with Web Service-enabled business processes.

From service discovery perspective and based on (Sivashanmugam et al., 2003; Srinivasan, Paolucci, & Sycara, 2004) the limitations are firstly in the search mechanism where coarse results with high precision and recall errors might results and secondly in the XML usage to describe service repositories' data models that causes syntactic interoperability and fails to provide semantic description of its content.

[7] E.g., the Web that presents a disordered and unregulated environment.

More details about the problems of existing approaches are going to be presented in the following section.

### 1.2.2 Problems of Existing Approaches

In the last few decades, an increasing number of approaches and researches aimed at the integration of existing software systems. In this section, some of these related efforts in the research areas of Web Service-enabled and Semantic Web Services-enabled SOA are listed and discussed.

#### 1.2.2.1 Web Service-enabled SOA

SOA represents a concept and Web Services are considered merely as a technology to realize it[8]. Despite the fact that most building blocks of SOA concept were set up before Web Services brought into life[9], the majority of modern service-oriented architectures utilize them as underlying technology (Keen et al., 2004). Business applications based on SOA concept generally wrap their functionalities using standardized Web Service interfaces[10]. One example is the Federated Enterprise Resource Planning (FERP) system that is totally based on SOA standards (Brehm, 2009). It follows the already mentioned wrapping process. The problem that encounters FERP system is the heterogeneity of its data models, interfaces and architectural components (Brehm, Lübke, & Marx Gómez, 2007, p. 303). As a contribution to this issue, (Brehm & Marx Gómez, 2005, p. 105) proposed that these disparate components have to be standardized in order to make FERP services interoperable with each other. What is still not solved in FERP system is the lack of semantic definition of its Web Service interfaces besides the traditional drawbacks inherited from SOA systems (see Section 1.2.1).

#### 1.2.2.2 Conceptual Frameworks

There are many efforts done in the field of Semantic Web Services. The most notable conceptual frameworks in this domain (Mahmoud & Marx Gómez, 2010, p. 6; cf. Mahmoud, 2009, p. 476) are listed in Table 1.1: SWSF (Battle et al., 2005a), WSMO (De Bruijn et al., 2005) and OWL-S (Martin et al., 2005).

[8] Most researchers in distributed computing world agree upon the point that the main roles, operations and principles of SOA and Web Services are similar.

[9] Any service within SOA concept is totally independent and might have nothing to do with the Web Service concept.

[10] There are many ways to develop SOA solutions using standards, but all of these standards have to use Internet protocols.

**Tab. 1.1:** Conceptual Frameworks

| Conceptual Framework | Description |
|---|---|
| **Semantic Web Services Framework (SWSF)** | SWSF includes the Semantic Web Services Language (SWSL) (Battle et al., 2005b), that is used to define the Semantic Web Services Ontology (SWSO) (Battle et al., 2005c), besides individual Web Services. |
| **Web Service Modeling Ontology (WSMO)** | WSMO initiated the standards related to Semantic Web Services. It represents a meta-model for the related aspects of Semantic Web Services (De Bruijn et al., 2005). |
| **Ontology Language for Web Services (OWL-S)** | OWL-S is an ontology to describe the related aspects of Web Services. It is part of the OWL-based framework of the Semantic Web (Martin et al., 2004). |

As mentioned in (Roman et al., 2005), WSMO represents a formal model for describing diverse aspects related to Semantic Web Services. It is based on the Web Service Modeling Framework (WSMF) (Fensel & Bussler, 2002). Based on the researches of (De Bruijn et al., 2005; Fensel et al., 2007, 2011), the main objective of WSMO (see http://www.wsmo.org) is to define a consistent technology for Semantic Web Services by providing the means for semi-automated discovery, composition, and execution of Web Services based on logical inference-mechanisms (Cimpian & Mocan, 2005). WSMO applies Web Service Modeling Language (WSML) (De Bruijn et al., 2006) as an underlying language based on different logical formalisms.

Found in (Martin et al., 2005), OWL-S is an ontology[11] for describing Semantic Web Services represented in OWL[12] (Patel-Schneider, Hayes, & Horrocks, 2004). It compounds the expressivity of description logics together with the pragmatism related to the emerging Web Service standards, so that services can be expressed semantically and yet grounded within a well-defined data typing formalism (Cabral et al., 2004, p. 234).

The abovementioned Semantic Web Services conceptual frameworks represent heavyweight semantic systems that are dealing with complex ontology specifications and rules. They introduce new languages based on expressive formalisms and follow the

[11] In philosophy, an ontology is a systematic account of existence. (Gruber, 1993, p. 1) defined an ontology as a formal explicit specification of a shared conceptualization and this definition is adopted whenever the term “ontology” is found in this thesis.

[12] OWL stands for Web Ontology Language and it is used to define the meaning of information to machines as well as to humans (McGuinness & Van Harmelen, 2004).

top-down modeling approach[13]. This leads to the conclusion that it is difficult to use such semantic frameworks in running business environments because of their complexity and the less focus on business values.

#### 1.2.2.3 Lightweight Semantic Annotation Mechanisms

In the field of Web Services and Semantic Web Services, there are many tools, ontologies and mechanisms to apply lightweight semantic annotations to services. SAWSDL, WSMO-Lite and MicroWSMO are listed in Table 1.2 with a brief discussion for each of them. This section then defines some of their problems.

**Tab. 1.2:** Lightweight Annotation Mechanisms

| Annotation Mechanism | Description |
|---|---|
| **Semantic Annotations for WSDL[14] and XML[15] Schema (SAWSDL)** | SAWSDL represents a standard to relate WSDL documents to semantic descriptions like OWL-S and WSMO (Farrell & Lausen, 2007). |
| **WSMO-Lite** | Using SAWSDL annotation mechanism, WSMO-Lite represents a lightweight set of semantic service descriptions (Fensel et al., 2010). |
| **MicroWSMO** | MicroWSMO represents a microformat-like mechanism to annotate RESTful Web Services (Maleshkova, Kopeckỳ, & Pedrinaci, 2009). |

The World Wide Web Consortium (W3C) (see http://www.w3.org) defined and initiated SAWSDL working group at April 2006. SAWSDL (Kopeckỳ et al., 2007) is based on the WSDL-S[16] W3C member submission (Akkiraju et al., 2005). It provides standardization methods by which traditional WSDL documents can be extended by a set of lightweight semantic descriptions, such as those provided by the Semantic Web Services frameworks (Martin, Paolucci, & Wagner, 2007).

[13] By following the top-down modeling approach, these frameworks give the full intention just to provide semantics first.

[14] Web Services Description Language (WSDL) is a W3C recommendation. It is an XML-based language used for describing Web Services (Christensen et al., 2001). The WSDL acronym has changed from version 1.1 where the D letter stood for Definition.

[15] XML stands for eXtensible Markup Language.

[16] WSDL-S or Web Service Semantics is based on and extends WSDL by having new elements and extensions for the existing WSDL elements. In other words, WSDL-S connects WSDL and OWL together (Herrmann, Dalferth, & Aslam, 2007).

WSMO-Lite (Kopeckỳ & Vitvar, 2008) is a service ontology created on the basis of SAWSDL as a community effort resulted in the CMS working group[17]. (Vitvar et al., 2008, p. 675) stated that WSMO-Lite represents a lightweight service ontology that annotates WSDL descriptions using SAWSDL and applies RDFS[18] as a main description language.

(Kopeckỳ, Vitvar, & Fensel, 2008) stated that MicroWSMO adopts the WSMO-Lite service ontology as a reference ontology to describe services semantically (Pedrinaci et al., 2010, p. 246). It represents a semantic annotation mechanism for Representational State Transfer (REST) Web Services[19] using the hRESTS microformat[20] (Kopeckỳ, Gomadam, & Vitvar, 2008) to provide machine-readable service descriptions (De Giorgio, Ripa, & Zuccalà, 2010, p. 342).

To sum up, SAWSDL annotation is used to describe Simple Object Access Protocol (SOAP)[21] services semantically and MicroWSMO is used to describe the RESTful Web Services using hRESTS, and together they constitute the WSMO-Lite service ontology (Fensel et al., 2011, p. 280).

All of these lightweight semantic annotation efforts share the idea of augmenting existing service specifications with semantic descriptions following the bottom-up modeling approach. They also cover the other grounding approaches namely: SOAP-based and RESTful services using their lightweight service ontologies. As discussed in (Lytras & García, 2008), these efforts extend the existing service descriptions with semantics focusing just on providing semantics with less effort to address the business values.

#### 1.2.2.4 Other Related Work Problems

To complete the analysis of the state-of-the-art related work problems, there is the research area of component-based software engineering (Heineman & Councill, 2001). By enhancing this domain with the so-called “business components” (Turowski, 2001), the main outcomes behind this research domain are shaped and refined in the context of business application systems (Fellner & Turowski, 2000). As stated in (Conrad & Turowski, 2001, p. 153), a business component is an application-oriented component

[17] CMS here stands for Conceptual Models for Services (Lambert & Benn, 2010).

[18] Resource Description Framework Schema (RDFS) is a vocabulary description language and W3C recommendation. While RDF is accorded as a general-purpose language to represent the information in the Web, RDF Schema is used to describe RDF vocabularies (Brickley & Guha, 2004).

[19] REST as proposed by (Fielding, 2000) in his PhD dissertation is an architectural style of networked and distributed hypermedia systems. REST-based services or RESTful Web Services (Richardson & Ruby, 2007) are Web Services implemented using REST principles.

[20] HyperText Markup Language (HTML) for RESTful Services or hRESTS is an HTML microformat used to describe the main aspects (inputs, outputs, operations) of the RESTful services.

[21] SOAP (Lafon & Mitra, 2007) is a protocol specification designed to exchange information while implementing Web Services in distributed computing environment. REST differs from SOAP in the point that it is designed to be employed in communications over Hypertext Transfer Protocol (HTTP).

that provides a specific set of services using well-defined interfaces out of a given business domain. Detailed efforts and discussions on how to standardize business components had been already provided in (Turowski, 2000). XML offers the opportunity to standardize the component's data formats so that the overall software systems interoperability can be improved using standard communication protocols and data formats (Brehm, 2009, p. 5). According to the comparison made by (Schmietendorf et al., 2003, pp. 31–33), a Web Service can be technically described, provided and registered in central directories as a set of business components. However and since they are supplied in development, Web Services reside within the Internet and can remain within their own applications as well. This means that an application can be implemented based on a network of distributed business components called Web Services. Therefore, SOA problems can still apply to these business components.

## 1.3 Thesis Statement

The main aim behind this work is the derivation and systematic presentation of a new approach for the implementation and delivery of a lightweight semantic enterprise system by relying on Web Services. This system can be considered as a special form of business software that enables the proper use of components and application functions supplied by competing providers. To develop and supply such components, a technical information infrastructure approach is developed together with a business case that implements and applies its main outputs.

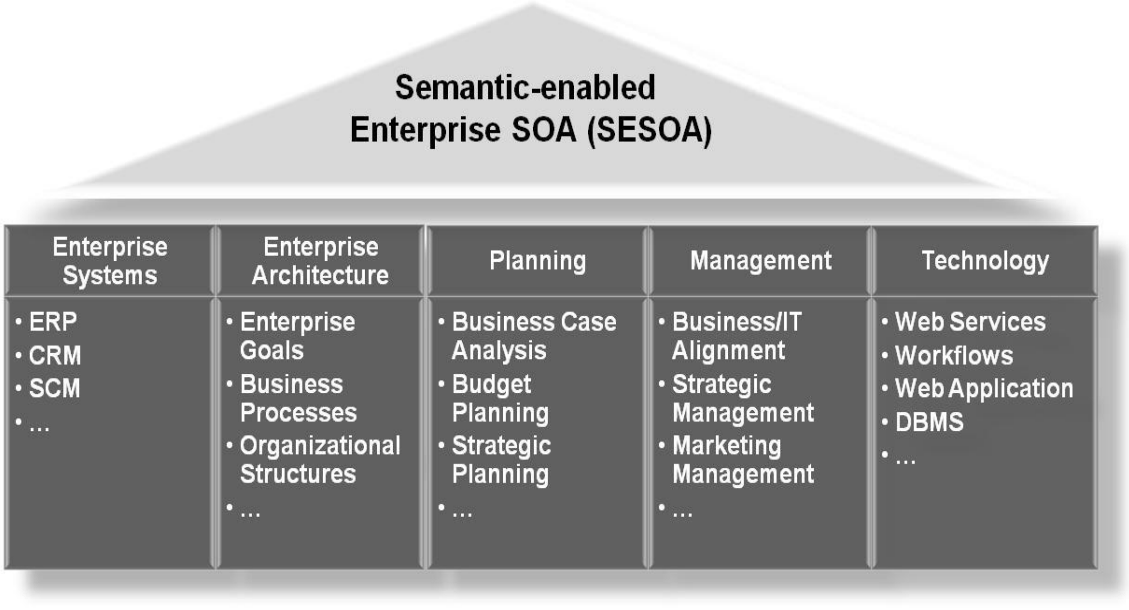


**Fig. 1.2:** The Related Research Topics

Figure 1.2 represents the main research topics that form the basis of the proposed approach. It is realized based on the idea of applying lightweight semantic annotations on

Web Services. The output of this work can be applied to different application domains like Enterprise Resource Planning (ERP) (Monk & Wagner, 2008; Robert Jacobs & "Ted" Weston, 2007), Customer Relationship Management (CRM) (Grönroos, 2000), Supply Chain Management (SCM) (Handfield, Nichols, & Ernest, 1999)...

Currently, one of the concepts involved in enterprise integration is the SOA concept. With the advent of the Semantic Web (Berners-Lee & Hendler, 2001) and its semantic annotation techniques, the proposed approach is involved in forming the future version of integration among enterprises. It is considered as a component-based software system that enables the integration of heterogeneous business application using Web Services.

Moreover, it derives its main aspects from multiple disciplines in enterprise engineering[22]: Enterprise Architecture, planning, management and technology. As defined in (Giachetti, 2010, p. 102), an Enterprise Architecture represents the construction of an enterprise and its business entities (subsystems), their properties and relationships, the relationships to the external systems together with the main principles to design and evolve an enterprise. The main concepts derived from Enterprise Architecture include business architecture, technical architecture, IT-governance besides modeling, organization, and realization of enterprise business processes, visions, and goals. Planning and management play also an important role in shaping and enhancing the suggested approach. The main terms derived from planning include strategic and budget planning[23], business case analysis, migration strategies, sourcing together with program and project scoping. Management science is quite important and plays also an important role in designing and developing the proposed approach starting from how to align business with IT and ending with marketing and information management.

The proposed solution seems to mediate between the supply and demand of software components represented as Web Services in an open market. Moreover, several technologies have been used to realize the desired solution. Web Service technology has been used to realize the SOA concept. Workflows executed and managed by workflow management systems have been used to implement business processes. Furthermore, Database Management System (DBMS) has been used to create, maintain and use the system's databases. RDF statements have been used to annotate the relations among services. Furthermore, the resulted artifacts of this work are implemented as Web appli-

[22] Enterprise engineering in this context can be defined following (Dietz, 2006, p. 71) as an entire knowledge consistence that represents all the aspects of development, implementation and operational use of enterprises together with its practical application derived from the engineering projects.

[23] While strategic planning determines to where an enterprise is going to be by time and how it will get there, budget planning is setting the financial budgets for the enterprise and each of its organizational units (cf. Steiner, 1997).

cations in which all the involved users can log in and benefit from the offered Web Services based on their rights and privileges.

Due to the very high number of functionalities and data types used in enterprise systems (Brehm, 2009), this study does not describe the actual content of such standards and specifications, rather it is dedicated to answer the following research questions:

- What is the difference between existing SOA-based systems and the proposed approach in terms of applying semantic annotations to services?
- How can the existing technologies and concepts of current enterprise systems as well as the general integration aspects of software systems be utilized and reused in the suggested approach?
- How the relations among services are annotated and later on made available in semantic service repository?
- How the validation and automatic evaluation of services can be achieved?

Answering these questions involves the development of a distributed enterprise system architecture enabled by multiple service providers. The aim of this architecture is to unify the technical information infrastructure among enterprises to facilitate the common enterprise system operations, in which each entity can ask the community for the available functionalities encapsulated and offered as services and/or provide them. Semantic annotations are applied to the service relations within this architecture. Service validation (cf. Maximilien & Singh, 2004; cf. von der Dovenmühle, 2009) and evaluation (cf. Hasan, 2010) constitute two major outcomes. The overall system in this dissertation is called "Lightweight Semantic-enabled Enterprise Service-Oriented Architecture (SESOA)".

Figure 1.3 shows the big picture of the developed approach. As it can be seen in this figure, in any enterprise there are bunch of systems that communicate with each other like SCM, CRM, ERP, accounting, legacy systems, etc. in a daily manner. These systems have their own way in representing data. This makes enterprise looking like a medium[24] in which disparate and heterogeneous systems are collaborating to achieve their business goals (cf. Hu et al., 2008). The first step in this work is the inspection of the current use of such conventional enterprise systems (more precisely the SOA-based ones) and the exploration of their internal structures. In the second step, SESOA is proposed as a service architecture that enables the provision and use of Web Services within a distributed network via its semantic service repository. This approach is developed to be responsible of splitting the semantic annotation from the core services description

[24] Medium in this context means a production channel that stores and/or supplies information or data to all its organizational sections.

where both Web Services and Semantic Web Services can be used, evaluated and validated.

The following points summarize the main objectives that can be derived from this dissertation where some of them are inherited from other related works:

a) to improve the time-to-market via simplified marketing of the system's subcomponents

b) to reduce system maintenance complexity

c) to open the market to include small, medium and large businesses

d) to validate and evaluate Web Services from both functional and non-functional perspectives

e) to consolidate business processes and service orientation concepts to effectively utilize the market's best practices.

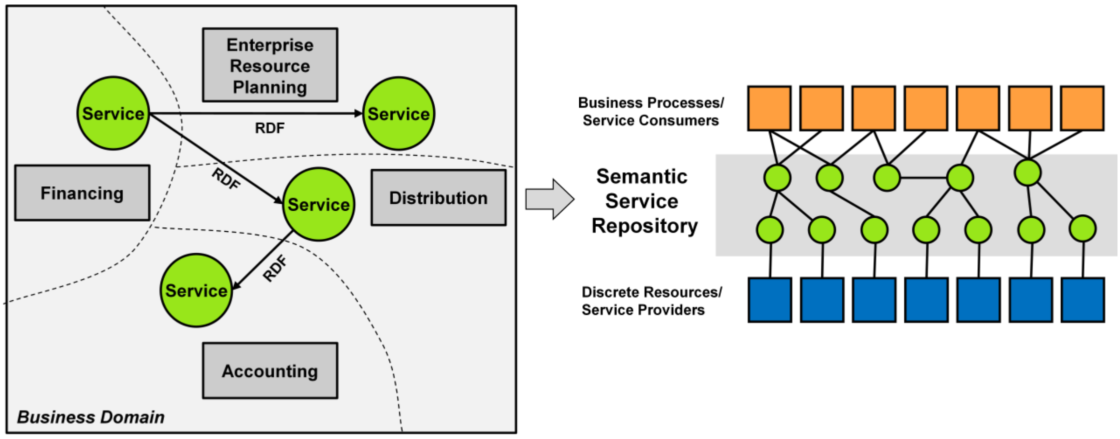

Enterprise Resource Planning
Service
Service
RDF
Financing
RDF
Service
Distribution
Service
RDF
Accounting
Business Domain
Business Processes/ Service Consumers
Semantic Service Repository
Discrete Resources/ Service Providers

**Fig. 1.3:** The Big Picture

As a last step in this work, the subsequent prototypical implementations are presented in details to show the practicability of the proposed approach.

## 1.4 Thesis Structure

The structure of this thesis is realized in eight chapters as depicted in Figure 1.4. Chapter one is the introductory part of the thesis. It includes the motivation behind this work, the related work, the problem definition that this work addresses, thesis statement, and a reader's guide. The main related concepts and technologies are placed in Chapter two including a short overview of distributed computing, Web Services, Business Process Management, workflows, and the Semantic Web pyramid.

Enterprise SOA and other architectures like Enterprise Architecture and Software Architecture are explained and compared in Chapter three. Chapter four illustrates the research methods that have been followed in this work. Chapter five illustrates the general requirements and the semantic support of Web Services together with defining the main requirements of the accompanying business case.

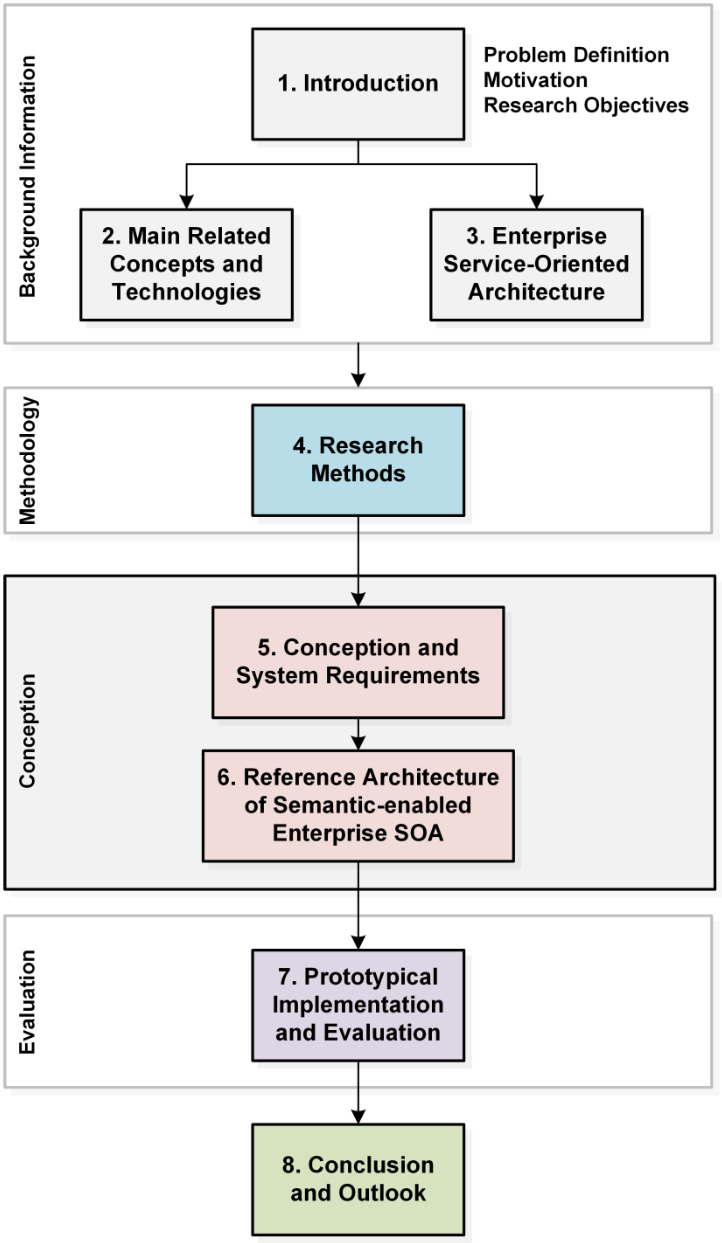

Background Information
1. Introduction
Problem Definition
Motivation
Research Objectives
2. Main Related Concepts and Technologies
3. Enterprise Service-Oriented Architecture
Methodology
4. Research Methods
Conception
5. Conception and System Requirements
6. Reference Architecture of Semantic-enabled Enterprise SOA
Evaluation
7. Prototypical Implementation and Evaluation
8. Conclusion and Outlook

**Fig. 1.4:** The Structure of the Thesis

The definition of SESOA and its reference architecture are explained in Chapter six where the core building components are formally described using various Unified Modeling Language (UML) diagrams. Web Service validation and evaluation and the main system interactions are explained in this chapter as well.

Chapter seven demonstrates the prototypical implementations of the resulted artifacts. It gives highlights to the SESOA main Web application together with the Web application of the accompanying business case. It further shows how SESOA concept has been evaluated in four different application fields to prove the applicability of this approach in an industrial context.

Finally, the work's main contributions together with future work directions are summarized in Chapter eight.

# 2 Main Related Concepts and Technologies

This chapter provides a holistic background information and overview about the main traditional and Semantic Web disciplines. From the traditional Web[25] perspective, the main related concepts to this work include distributed computing, Web Services, Business Processes Management and workflows. As from the Semantic Web perspective, the main related concepts to this work include the Semantic Web pyramid and its layers.

## 2.1 Distributed Computing

One of the important challenges to achieve distributed computing is remoteness and its abstractions. Services serve as one of the distributed computing's software systems. This means that a consumer machine remotely invokes a computer program (service) that doesn't reside in the same physical machine. This remoteness is hidden from the consumer and it is the provider task to select the proper invocation scenario with a suitable degree of granularity. Distributed computing infrastructure has been around for almost three decades. It started with the traditional costly large computers and mainframes that perform tasks on their own (Brodie, 1992). Later on, the development had been shifted towards more interactive multi-user systems that enabled the distribution using terminal devices like DEC VT100[26] or the IBM 3270. This distribution was only from data capturing and display perspectives by sharing just data and output devices (cf. Krafzig, Banke, & Slama, 2005, p. 19).

In the beginning of the 1970s, the computers became more small and cheap and mainframes were substituted with these smaller computers. The price/performance ratio accompanied with the evolution of small computers paved the way towards distributed operating systems like UNIX where the use of network was essential part of it. This enabled the ideas of controlling programs and computers remotely besides providing services to other computers across the network. Many tools to enable these ideas were emerged like telnet and the Berkeley r-tools. Moreover, remote printing and storage space administration were facilitated by the accompanying file systems like for example the Sun Microsystem's NFS[27] file system. NFS was the main origin of the SUN-RPC standard that is in turn considered as one of the foremost Remote Procedure Call (RPC) standards (cf. Krafzig et al., 2005, p. 19).

Lately in the 1990s, the spoke-hub distribution paradigm in which a storage and printing central system could be accessed by many desktop computer systems had been developed. The client/server model was another distribution paradigm emerged as the relational databases became more and more popular. In this model, the server provides a

[25] What is meant by the traditional Web is all of the Web classifications in the pre Semantic Web era.

[26] VT100 is a terminal device created by the Digital Equipment Corporation (DEC) company.

[27] NFS or Network File System developed by Sun Microsystems was one of the first distributed file systems that enabled users to access files over the network.

service or a resource to a client that requests such service. Big portion of the application logic was laid at the client side that remotely accessed a database server. The execution logic was split among the client and the server (Orfali, Harkey, & Edwards, 2007). The next step in the client/server application development was to make a clear distinction between the client and the server by developing distribution platforms like Distributed Computing Environment (DCE) and Common Object Request Broker Architecture (CORBA) promoted by the OMG[28]. To save the number of servers needed to serve the clients, CORBA introduced the approach of breaking down the functionality into uniquely remote access objects that can manage its own states (Bolton, 2002). These objects communicate with each other through Object Request Broker (ORB) middleware that supports the interoperability among the objects developed and supplied via different vendors. Such interoperability had been achieved through the introduction of abstraction mechanisms like naming services that handled the objects runtime discovery. Likewise to object-oriented programming, CORBA adopted the programming by interface principle where the objects were developed in several programming language and their interfaces were described using Interface Definition Language (IDL) (cf. Krafzig et al., 2005, p. 20).

Another example of distribution frameworks is the DCOM[29]. It is promoted by Microsoft Corporation as a technology developed for distributed components. It is an extension of its COM ancestor. It added the distribution of software components over networks. COM and DCOM had been merged then into one runtime that supported and enabled both local and remote access functionalities (Grimes & Grimes, 1997).

In the mid-1990s and as a result of the complexities accompanied by CORBA and its IDL language mappings, the need of more stable platforms for Internet applications was inevitable. Sun Microsystem’s Enterprise Java Beans (EJB) based on distributed object model is an example of such platforms. EJB supports many types of objects like data-centric entity beans and session-oriented objects. Moreover, it introduces the container concept to manage resources making such management as transparent as possible to the developers. However, while EJB and the numerous number of similar distributed computing platforms tried to solve the application heterogeneity problem, the middleware heterogeneity problem emerged and new enterprise-wide standards to solve such heterogeneity are needed (cf. Krafzig et al., 2005, p. 21).

XML was the logical successor of this development as a middleware-independent format to exchange data among heterogeneous applications. It is one of the few standards upon which the IT industry could agree. In contrast to the aforementioned distributed computing concepts, XML is independent from technologies and middleware standards. It represents an ad-hoc format for handling data among heterogeneous middleware plat-

[28] OMG stands for Object Management Group. It is one of the most popular international industry consortiums in business information systems field.

[29] DCOM stands for Distributed Component Object Model.

forms. Moreover, XML is seen as a dominator that solves both the application and the middleware problems. However, even if it is considered one of the major advantages of XML, flexibility had been seen as one of its major problems since the call for higher-levels of data structures and messaging formats (from semantic point of view) becomes inevitable (Haller, Gomez, et al., 2005; cf. Krafzig et al., 2005, pp. 21–22).

XML-based Web Services have been developed then by Microsoft engineers as a result of the need for higher-level XML messaging standards. Relying on XML, Web Services also tried to leverage the omnipresence of Internet. SOAP is then created and used with Web Services as a lightweight protocol to enable server-to-server besides the browser-to-server communications. Similar to IDL that describes object interfaces in CORBA, Microsoft designed the WSDL as an interface definition language. SOAP and WSDL are developed to solve the middleware heterogeneity problem by the definition of several bindings to various lower-level communication protocols. XML-based Web Services are considered as the main technology to implement SOA but this doesn't mean that it is the only way. Rather, there are many technologies to realize SOA and Web Service technology is just the most popular and stable one. Summing up, as object orientation has been widely considered as the endpoint of the programming development, service orientation is anticipated to be the most mature and stable outcome from the long evolutionary distributed computing (Bieberstein et al., 2005; cf. Krafzig et al., 2005, p. 22).

## 2.2 Web Services

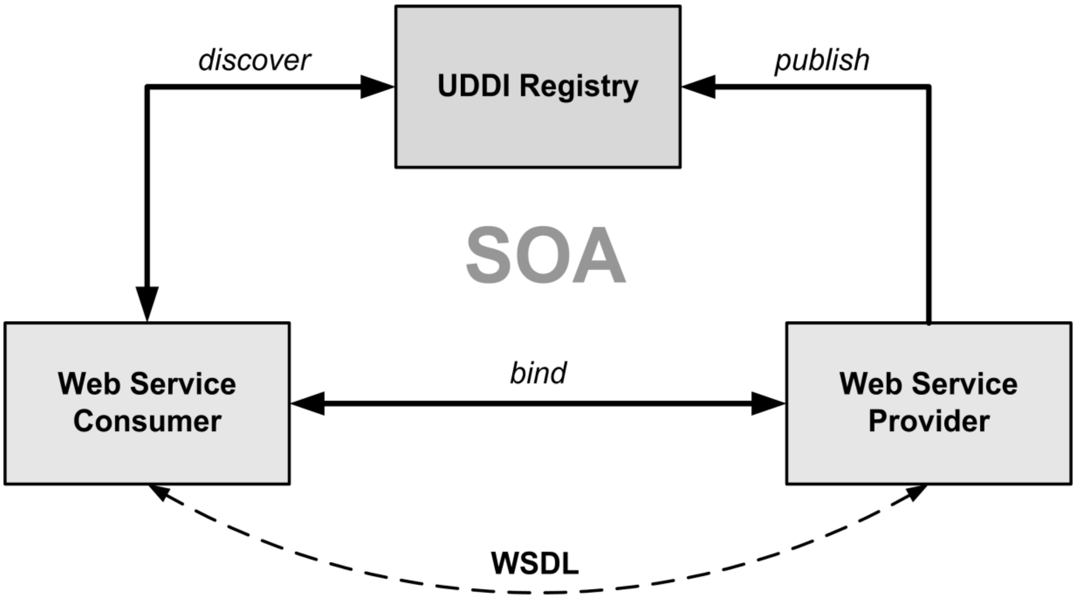
Before defining Web Services, it is important to comprehend the concept behind services in general. A service as defined by (Katzan, 2008, p. 11) is "a provider/client interaction that creates and captures value. A unique characteristic of services, unlike agriculture and manufacturing, is that both parties participate in the transaction, and in the process, both capture value. In a sense, the provider and the client co-produce the service event, because one can't do without the other". As for a Web Service, a variety of definitions can be found in literature. One of these definitions defines the Web Service as: "a business function made available via the Internet by a service provider and accessible by clients that could be human users or software applications" (Casati & Shan, 2001). A Web Service as defined by the W3C consortium is "a software system designed to support interoperable machine to machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" (Booth et al., 2004).

These different definitions complement each other and each of them reflects one part of the Web Service characteristics. As from a technical perspective, a Web Service has two parts: a contract that specifies its business functionality (purpose, functionality, price, constraints, usage…), and an implementation of this functionality. Each Web Service is supplied by a service provider to be discovered and invoked by a service consumer.

From business perspective, Web Services must be programmatically accessible in a sense that their design enables them to be called and invoked by other Web Service applications (cf. Medjahed, 2004, p. 49). Moreover, Web Services are designed to follow the loose coupling principle in a way that the dependencies between the service contract, the service implementation, and the service consumers are as much reduced or loosened as possible (cf. Erl, 2009, p. 753).

### 2.2.1 Web Service-enabled SOA

This section illustrates the concept of service orientation enabled by Web Service technology. The OASIS[30] SOA Reference Model group defines SOA as a: “paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations” (MacKenzie et al., 2006, p. 29). Several technologies are used to realize SOA but Web Services are considered the most common one[31].



Source: Figure 2 in **(Mahmoud & Marx Gómez, 2008a)**

**Fig. 2.1:** Web Service-enabled SOA

As explained in (Mahmoud & Marx Gómez, 2008a) and illustrated in Figure 2.1, there are three main components that constitute the Web Service-enabled SOA:

[30] OASIS stands for the Organization for the Advancement of Structured Information Standards. It is together with W3C, the main source of Web Service standards and recommendations.

[31] Software agents for example are other technologies to realize SOA.

- Service Provider: It creates Web Services and possibly publishes their interfaces and access information at the service registry.

- Service Registry[32]: It is responsible for making the access information of both Web Service interface and implementation available to any potential service requester by categorizing the results in taxonomies. An example of service registry is the Universal Description Discovery and Integration (UDDI) by which the information about Web Services can be published and discovered.

- Service Consumer: It is the Web Service client that locates entries in the service registry and then binds directly to the service provider to invoke one of its Web Services.

SOAP, WSDL, and UDDI standardization initiatives were adopted by the W3C consortium to facilitate and enable the interactions between Web Services.

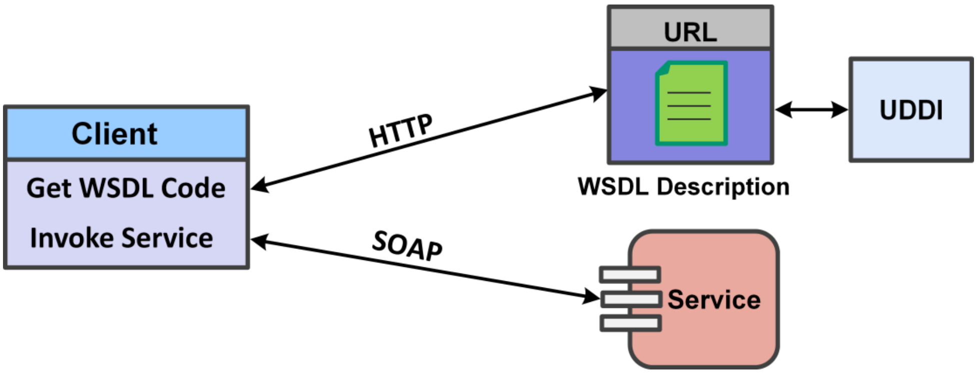


**Fig. 2.2:** Interoperability Using WSDL

As can be seen in Figure 2.2, the service client searches the UDDI for a Web Service that might match his needs. It can browse its WSDL description over HTTP connection to see whether the service fulfills its request or not. Upon selection, the service client invokes the Web Services and the communications between the client and the service are achieved in form of request and response SOAP messages. What can be concluded is that the WSDL represents a contract between the service consumer and the service provider. It is platform and language-independent to describe the SOAP services. Moreover, WSDL is always seen as an automatic tool to generate client and server code. Since 2003, WSDL became a W3C standard.

### 2.2.2 Service Discovery

Service discovery can be classified into centralized and decentralized service discovery. Last part of the previous section briefed the centralized type of service discovery para-

[32] It is also known as service broker or repository.

digm. In such centralized environment, developers require automated systems for service discovery, to enable more Web Service interactions and less human effort. The traditional UDDI exists precisely for such need. However, it is not always the case that the service consumer can comprehend in advance the exact form and meaning of the WSDL description of a specific Web Service. Therefore, the combination of WSDL and UDDI accompanied by a coarse-grained of business description are not enough to facilitate the full- or semi-automated service discovery.

That shows the pitfalls of the centralized service discovery and opens the research for alternatives like the decentralized service discovery. If Web Services are considered as resources of peer-to-peer networks, search strategies can be applied to decentralized enterprise networks in which each peer (enterprise) can benefit of both the usage and the provision of access points (interfaces). Such provision is made then available to business application systems or business components (Brehm, Marx Gómez, & Rautenstrauch, 2006) without being dependent on a central control instance. Generally, two different scenarios are conceivable to make use out of the provision of business applications based on Web Services in a peer-to-peer network. First possible scenario enables the Web Services to provide access to enterprise services as part of a supply chain where companies act either as suppliers or as customers. Based on that, they can either search for product offerings (e.g. materials) or publish them. The second scenario enables the enterprises to share professional business functionalities in form of software components encapsulated and implemented as Web Services (such functionalities can be for example information resources about the interests of customers) (Brehm & Marx Gómez, 2007). In both scenarios, finding a suitable service is still problematic. This can be transferred into the field of peer-to-peer networks in which possible existing solutions can aid to bring the characteristics of open markets to the world of Web Services. In this sense, the focus behind decentralized Web Service discovery approach is mainly to increase the marketing value of Web Services and this is why it could be seen better than the centralized service discovery approach (cf. Brehm et al., 2008, pp. 30–31).

## 2.3 Business Process Management

This section gives an overview of the Business Process Management (BPM). It starts with explaining the main ideas behind BPM besides making a distinction between BPM and Business Process Management Systems (BPMS).

Many concepts were considered as the ancestors of the BPM concept. At the late 1980s and the beginning of 1990s the term Business Process Reengineering (BPR) was one of the predecessors of BPM. Starting from the early 1990s and as described in (Hammer & Champy, 2003), excess number of reengineering and process orientation topics came into view. These topics were focusing on the delivery of striking business performance improvements in a relatively short time by doing reinvention of the existing business processes starting from scratch to create totally new business processes. This had been seen ineffective because such approaches threw off the old business processes. After investing a lot of money on BPR, the development became idle because of many reasons like reluctance to change, lack of understanding business models and its underlying processes among many other reasons (cf. Krafzig et al., 2005, p. 104). Ten years after BPR, the whole development process had been reactivated under the BPM term umbrella (Frankel, 2003). The main differences between BPR and BPM are shown in Table 2.1.

**Tab. 2.1:** BPR and BPM Comparison

| Business Process Reengineering | Business Process Management |
|---|---|
| Business processes are often created from scratch | Business processes are transformed from the existing ones |
| BPR revealed insufficient change and process optimization | BPM recommends incremental change and evolutionary optimization |

As defined by (Krafzig et al., 2005, p. 104) BPM is: “a general management topic that focuses on the strategic and the operational aspects of process orientation in a given business area”. Speaking about BPM, it is important to distinct between business and IT. While terms like ISO 9000 or Six Sigma are often related to the business side of BPM, the more technical terms like process modeling and workflow management are more related to the IT side of BPM. Business Process Management System (BPMS) represents the technical platforms that realize the management initiatives of the BPM and provide business process integration solutions. These technical platforms include many tools such as BPM engine, business process monitoring, and design tools (cf. Krafzig et al., 2005, p. 105).

It is important to analyze business processes before using them in production systems. This detects the early problems that might appear at the design phase like unsatisfied customers, damage claims, and the need to exceed capacities (van der Aalst, 2009). Business process analysis represents a major part of BPM. It aims at identifying the states of current information systems besides pointing out their problems and bottlenecks. Analyzing a business process can be performed through validation, verification, and performance analysis. While validation’s main role is to test the business process whether it behaves as expected or not, verification intends to establish the correctness of a process definition (cf. Celino et al., 2007; Mahmoud, Petersen, & Rummel, 2012).

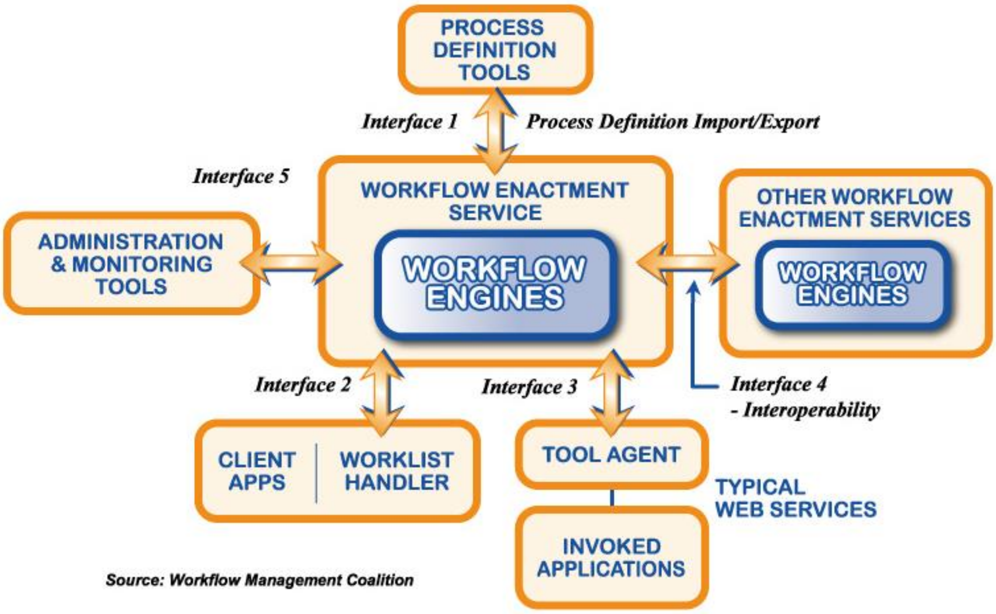
To sum up, BPM supports the entire lifecycle of modeling, executing, and monitoring of business processes. Moreover, BPM introduces the process processing concept and comprises the aspects of discovering, designing, and deploying business processes besides applying executive, administrative, and supervisory control over these processes to make sure that they stay compatible with the business objectives (cf. Frankel, 2003; cf. Krafzig et al., 2005).

## 2.4 Workflows

This section defines workflows as technical foundations to realize business processes. It then follows with illustration of the workflow reference model as proposed by the Workflow Management Coalition (WfMC) (see http://www.wfmc.org/).

In the mid-1990s, WfMC had published a glossary of all terms related to workflows. As defined by (Hollingsworth, 1995, p. 6), a workflow is: "the computerised facilitation or automation of a business process, in whole or part" or "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" where the participant represents a resource (either human or machine). Workflow Management System (WfMS) represents the technical platforms that manage the whole lifecycle of workflows.

To make the workflow-related aspects unified, WfMC developed the workflow reference model. This reference model is depicted in Figure 2.3.



Source: Workflow Management Coalition

**Fig. 2.3:** The Workflow Reference Model

As can be seen in the figure above, the main and central part of the workflow reference model is the workflow enactment service. It represents the core component of any known workflow system. It is comprised of one or more workflow engines in which each engine deals with bunch of cases (processes). These engines are designed to enhance scalability of the systems to which they are applied and they are potentially not noticed by the user. Based on (Hollingsworth, 1995), the following explains the main interfaces of the workflow reference model:

The first interface is with the process definition tools. These tools include: process definition, resource classification, and analysis tools. These tools are responsible of:

- Creating/deleting cases
- Routing access among cases
- Managing case's attributes
- Managing/handling triggers
- Recording historical data
- Providing a workflow's summary
- Monitoring the consistency of a workflow
- Starting up application software during activity execution
- Submitting work items to the right resources based on specific resource classification.

The second interface with the client applications is used by employees mainly to execute processes. These applications include standard or integrated workflow handlers that present the work items and their attributes. These handlers provide relevant properties of a specific work item, sort and select a work item, and report the completion of an activity.

The third interface with the invoked applications starts when the tasks are executed. These applications normally don't form a part of the workflow management system. Rather, they belong to the workflow system itself. They are normally classified into interactive and fully-automatic applications. Interactive applications can be initiated by the selection of a work item and can be any kind of text processors, spreadsheet, or electronic form filling tools.

The fourth interface is with the other enactment services. The main responsibility of this interface is to link several kinds of workflow systems. Via this interface, the cases or parts of these cases can be transferred from one workflow system to another. This interface is mainly designed and developed to enable workflow interoperability in general.

The last interface in the workflow reference model is with the administration and monitoring tools. These tools can be split into operational management tools and recording and reporting tools. Operational management tools can be case-related or non-case-related to administer cases and inspect their logistical state[33] respectively. The recording and reporting tools are responsible of collecting historical data for performance analysis.

The workflow reference model motivates the design and development of the "Processing System" within the SESOA approach. It is considered as a workflow management system. It provides a runtime environment for executing the system's workflow instances. The main characteristics of this system are derived from its counterpart of the WfMC reference model (Hollingsworth, 1995, p. 20). It is explained in details later in Chapter six (Section 6.1.3.1).

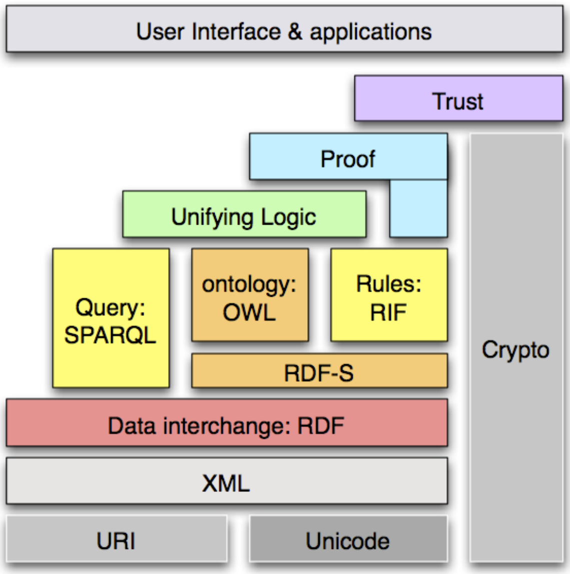
## 2.5 Semantic Web Pyramid

This section illustrates the main interactions and markup language pyramid in the world of Semantic Web. As it is widely known, most Web pages nowadays contain themselves hyperlinks to other related-pages, downloads, source documents, and other Web resources. Such a collection of pages that are interconnected via hypertext links is what was called a "Web" of information (Cooley, Mobasher, & Srivastava, 1997). Making it available on the Internet initiated what Tim Berners-Lee first called the WorldWideWeb (Berners-Lee & Cailliau, 1990). Caused by the excessive growth of information in the Web, a need for a new approach becomes inevitable. Web 2.0 is defined as an extension to the WorldWideWeb to increase creativity, information sharing, and collaboration between users. Therefore, the mentality behind Web 2.0 was about aiding users to find ways to solve a problem among all other possible ways besides the continuous optimization of that way. However, Berners-Lee stated that "Web 2.0" is "a piece of jargon, nobody really knows what it means" (Berners-Lee, 2006). He stated also that the Web was mainly created to connect people. Summing this up in few words leads to the conclusion that having people connected to each other enables some of them to benefit from the others' experiences, which means that humans are the analyzing power that is forging best solutions. Machines have much more processing power than humans when the knowledge could be represented in a correct way to them. This last point can be considered as a pitfall in the Web 2.0. As a conclusion, the revolution towards new ways of utilizing machines in the information daily consumption leads to the idea of the Semantic Web (cf. Brehm et al., 2008, p. 27).

The main ingredients of the Semantic Web pyramid are illustrated in Figure 2.4. The main goal behind Semantic Web is to enable machine-readable intelligence based on the existing hyperlinked vocabularies that the Web pages use to express their words and concepts (Cardoso et al., 2007). The existing Web pages are written in Hypertext Markup Language (HTML) that describes the way the information is displayed and presented for the humans to read in form of Web pages. Computers are able to parse Web

[33] The logistical state of a case gives the operational management tools information about any problem and exceptional circumstances that might arise.

pages for layout and routine processing but they are unable to comprehend and process the meaning of their contents. XML supports the enablement of exchanging the data across the Web but it is still unable to provide the meaning of such data. Semantic Web languages are based on XML to structure the meaningful content of the Web pages. This enables the software agents roving across Web pages to perform automated tasks (cf. Alesso & Smith, 2005, p. 166).



Source: Figure 5-2 in **(Heuser, Alsdorf, & Woods, 2008)** & **(Bratt, 2006, p. 19)**

**Fig. 2.4:** Semantic Web Pyramid

The figure above shows the evolutionary development of the markup languages from the traditional XML to RDF to OWL. The first layer in this figure includes the URI[34] and Unicode. URIs can be used to identify Web resources like books, locations, etc. As for the Unicode[35], it is defined in (Xu & Peng, 2005) as “a character encoding system, designed to help developers who want to create software applications that work in any language in the world”. Unicode provides unique numbers up to one million characters that are independent from platforms, programs, and languages.

The second layer in the pyramid is the XML layer. XML represents the meta-language that enables Web users to create and define markups. It provides a clear separation between content and structure from one side, and formatting from the other side. XML represents the de facto standard that is used to represent and exchange structured information on the Web and it is nowadays supported by a variety of query languages. How-

[34] Sometimes it is referred to as URI/IRI or Uniform Resource Identifier/Internationalized Resource Identifier (Berners-Lee, Fielding, & Masinter, 2005). Uniform Resource Locator or URL is also part of URI/IRI where URLs are subsets of URIs and URIs are subsets of IRIs.

[35] American Standard Code for Information Interchange (ASCII) is an example of Unicode character encoding systems.

ever, tag nesting doesn't provide any standard meaning and the semantics of XML documents can be processed just by humans but not by machines (cf. Antoniou & Van Harmelen, 2004, p. 55).

This calls for new markup languages that try to provide machine-readable way to present and process data. RDF had been developed for this reason and became a W3C domain-independent standard (Manola, Miller, & McBride, 2004). It has sufficient expressive power by being based on statements that relate subjects to objects using predicates. It is more extensible than XML in a way that more layers can be built on top of it. RDF Schema (RDFS) extends RDF vocabularies to provide a primitive ontology language. It allows expressing facts and defines domain's semantics including its classes and properties, class hierarchies and inheritance besides property hierarchies as well. Finally, both RDF and RDFS allow the inference of explicit knowledge (cf. Antoniou & Van Harmelen, 2004, p. 104).

The next layer in the Semantic Web pyramid represents the ontology layer. Each ontology can be seen as a set of vocabularies with the explicit specifications of their meaning, object classifications, constraints and relations (Gruber, 1993). The main objectives behind developing ontologies are firstly capturing a kind of shared comprehension of a specific domain and secondly providing a formal and machine-readable model of such domain. OWL is one of the languages created to achieve such objectives and became a W3C standard (McGuinness & Van Harmelen, 2004). Similar to DBMS data dictionary or schema, OWL is considered as a global and standard syntax to describe ontology based on RDF/RDFS. However, the main difference between RDF and OWL is that the latter is capable of defining more complex and graph relationships. Moreover, through the mapping of OWL on logics like predicate logic and description logic (cf. Hitzler, Krötzsch, & Rudolph, 2009, p. 159), more formal semantics and reasoning capabilities can be provided.

The Semantic Web development was accompanied with queries, rules, transformations, and deployment. As can be seen in Figure 2.4, one of the query languages in the Semantic Web pyramid is the SPARQL Protocol and RDF Query Language (SPARQL). As defined by (Kjernsmo & Passant, 2009), SPARQL is a "concise query language to retrieve and join information from multiple RDF graphs via a single query". It provides a standard format to write queries that target RDF data and a set of standard rules to process such queries to return the results. It is considered as the de facto query language for RDF and it is one of the W3C recommendations. SPARQL is not XML-based rather it is based on a roughly SQL-like syntax in which RDF graphs are represented as triples. Moreover, it provides a simple communication protocol to be used by clients to apply SPARQL queries against specific endpoints (cf. Domingue, Fensel, & Hendler, 2011, p. 301).

As for the Rule Interchange Format (RIF), it is a set of W3C standards developed to ease exchanging rules across different and dissimilar rule engines, especially across Web-enabled engines. Similar to RDF and OWL, the main goal behind RIF is to make a revolution in the Web application development field by creating infrastructure dedicated for truly intelligent Web applications (cf. Domingue et al., 2011, p. 400).

More to be mentioned regarding the Semantic Web pyramid is that semantic query languages are developed as alternatives to the logic stack and via their functionalities they are overlapped with the unifying logic layer. The proof layer doesn’t constitute a proper layer anymore because it overlaps partially with the unifying logic layer as well. The proof and trust layers have mainly two major functions: firstly to prevent an upper layer in the Semantic Web pyramid from re-implementing functionality offered by a layer below and secondly to allow an application that only understands a lower layer to interpret at least parts of the definitions from a higher layer (cf. Domingue et al., 2011, pp. 19–20).

As a conclusion and from a business perspective, both academia and industry have realized that the Semantic Web can ease the integration and interoperability of intra- and inter-business processes and systems. Moreover, it can also enable the creation of global infrastructures to share documents and data. Such infrastructure makes searching and reusing of information far easier and tries to build better business applications than the existing ones (Cardoso et al., 2007; Brehm et al., 2008).

## 2.6 Summary

This chapter provided the main information of the most related concepts and technologies to this work. Since the main output of this work is Web Service-enabled solution, the importance in this chapter was given in the beginning to the distributed computing. Different historical developments of distributed technologies were explained to conclude that service orientation is commonly considered as the top of distributed computing development. The Web Service technology then was presented as a mighty solution to offer machine-to-machine interactions over network. This work goes with the opinion that Web Services are used as one of the common enabling technologies to realize service orientation. The Web Service-enabled SOA concept was then presented in this chapter and different service discovery scenarios were presented. From business computing perspective, BPM has also influenced this work. BPM aspects were illustrated in this chapter to show how proper BPM can ease the road to achieve the enterprise’s strategic and operational goals. Workflows have been explained as technical implementations of business processes and the WfMC workflow reference model had been explained in details.

Finally, the semantic aspects that influence this work were demonstrated as located in the Semantic Web pyramid. All layers of this pyramid were explained starting from Unicode and URIs to XML up to RDF and OWL markup languages. Moreover, semantic query languages like SPARQL and rules frameworks like RIF had been introduced.

# 3 Service-Oriented Architecture

This chapter gives a detailed overview of Service-Oriented Architecture (SOA) and lists its architectural considerations and basic principles.

## 3.1 Service-Oriented Architecture Concept

Service-oriented architecture is a software architecture model that offers services to end-user applications, executable business processes, or other services by means of published and discoverable service interfaces. Section 3.1.23.1.1 demonstrates the main motivation behind SOA and lists its basic principles. Section 3.1.2 points out SOA's architectural considerations. Web Service technology as enabler for SOA and the Web Service stack are explicated in Section 3.1.3. Adding business value to SOA results in the term business-driven or enterprise SOA is then introduced in Section 3.2. Enterprise Architecture and Software Architecture are then explained and compared with SOA in sections 3.2.1 and 3.2.2 respectively. The last section then concludes with a summary of the ideas presented in this chapter.

### 3.1.1 Motivation behind SOA

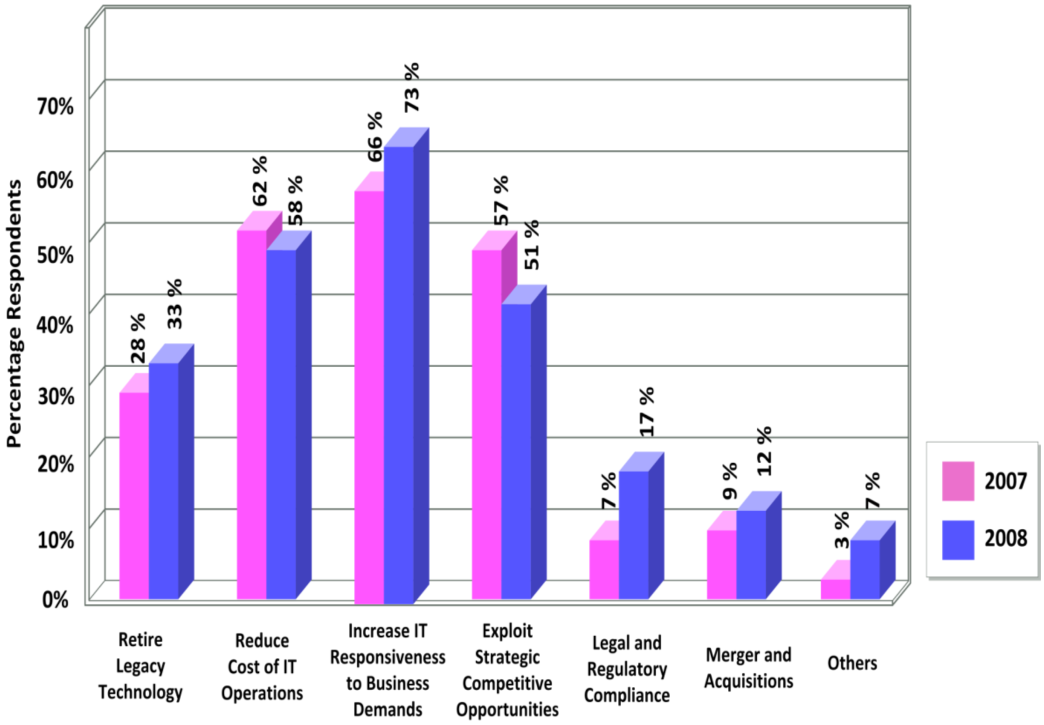
In SOA, business functionalities are realized and encapsulated in form of self-contained, distinguishable and reusable building blocks called services that:

- Represent high level business concepts
- Can be published and discovered in a network
- Can be reused to build new business applications

Service can be defined as "the means by which the needs of a consumer are brought together with the capabilities of a provider" where the service provider represents "an entity (person or organization) that offers the use of capabilities by means of a service" (MacKenzie et al., 2006, p. 29).

Figure 3.1 illustrates the results of a survey made by the Cutter consortium (Cutter SOA Survey, 2008) in which the primary drivers that could be seen behind SOA initiatives have been questioned.

As can be seen in the figure below, the most important factors for the companies to adopt SOA is to increase IT responsiveness to business demands, reducing the cost of IT operations besides exploiting strategic competitive opportunities. Less important factors were: retire legacy technology, legal and regulatory compliance, merger and acquisitions among some other factors.



Based on **(Cutter SOA Survey, 2008)**

**Fig. 3.1:** Reasons to adopt SOA

Based on the aforementioned survey results and as described in (Rosen et al., 2008, pp. 10–11), the main business and technical motivations behind SOA concept can be summarized as follows:

- Agility, flexibility, alignment: Agility and flexibility comes to mind when business processes within an enterprise can be effectively realized and implemented by benefiting from its available set of services (Erl, 2004; cf. Rosen et al., 2008, p. 17). Alignment is more complex. It requires the provision of a reference architecture that provides definitions for both business and information aspects of SOA. A common semantic model for designing service interfaces is considered as a main requirement to achieve the alignment (Shishkov, van Sinderen, & Quartel, 2006). Moreover, model-based techniques to follow up the status of implementing business model besides having processes to activate and validate conformance are required as well.

- Reusability: SOA achieves effective reusability by the enablement of publishing and discovery of services besides offering variety of functions that meet wide range of consumer's needs. This is in addition to provide capabilities to manage and maintain service lifecycle across all organizational sectors (Dan, Johnson, & Carrato, 2008; cf. Rosen et al., 2008, p. 13). Doing that eases the speed and simplicity of project deployment to a great extent. Moreover, the availability and lifetime of all service's versions have to be guaranteed as well. Finally, SOA must provide the proper mechanisms to decouple the life cycles of service consumers and service providers.

- Data rationalization and master data management: This is applied where data coming from different existing applications and enterprise systems are rationalized to obtain precise information to the services (Dreibelbis et al., 2008; Inaganti & Behara, 2007).

- Integration of applications and data: One of the most promising ideas behind SOA is to achieve integration of applications and data (Roshen, 2009; Hohpe & Woolf, 2003). The integration is needed to solve the communication problems among many heterogeneous systems by offering XML-based standardized services.

- Costs reduction: Promoting service reuse makes from SOA an effective concept to have considerable costs reduction (Channabasavaiah, Tuggle, & Holley, 2003, p. 18; Rosen et al., 2008, p. 25). This is done by using services across organizational boundaries that are having variety of business functions that suit all of the organizational units. This activates service reusability and reduces the development costs not just in a short scale, rather in the total cost of ownership behind the involved IT systems.

- Speed and simplicity of project deployment: When a service is deployed, it can be circulated over the network to maximize the performance or to get rid of data redundancy in order to provide optimum availability (Channabasavaiah et al., 2003, p. 10; Josuttis, 2007).

- External collaborators support: Service providers are considered the main players in the SOA paradigm. Service provision is not done just internally within the enterprise boundaries rather external service providers can also offer their services. This makes from SOA a sounding concept that initiates external collaborators support besides joining different business lines from various geographical regions to offer their services to speed up the time needed to enter the market (Bell, 2008; cf. Rosen et al., 2008, p. 108).

The following section gives an overview about a set of architectural considerations that have to be taken into account while developing SOA solutions.

### 3.1.2 Architectural Considerations

To complement the principles explained in the previous section, this section gives an overview of the architectural considerations that have to be taken into account when designing SOA. Following (Rosen et al., 2008, p. 30), the most common architectural principles and practices are as follows:

- Separation of concerns: As it can be comprehended from the name, the separation of concerns is a straightforward principle. It is often considered as one of the most significant architectural principles. The main goal behind separating concerns is to sustain the independency status of the independent elements (Erl, 2009; Tran, Zdun, & Dustdar, 2007). This means that changing part of the system does not influence on the system's other parts. Good practice accompanied with this principle is the separation of interface and implementation.

- Architectural views: It contributes also into the separation of concerns. This is achieved by including or excluding the presentation of information or specific data to variety of stakeholders. Such views are normally designed to refer to specific aspects of software development or alliance of enterprise groups and organizations that play a role in the full life cycle of enterprise solutions. Architectural views (concerns) can be classified as software and enterprise views. Software views can be sorted as logical, deployment, process, or network views. Enterprise views on the other hand can be applied to business, information, application, technology, or implementation levels. Fortunately, the implementation and design of SOA conform well to the before-mentioned set of architectural views (Newcomer & Lomow, 2004).

- Accommodation of change: It indicates that the architecture should support flexibility in a way that application requirements in the long term perspective can be fulfilled with less difficulty (Cheesman & Ntinolazos, 2004). This requires the identification of such long-term requirements and domains that are possibly subject to change. Reviewing preceding industry drifts can greatly assist in the identification of domains that are likely to change. Such identification has to be clearly reflected in the architecture. If such identification is ignored or not mapped into the architecture, there is a big chance to have difficulties when change to the ignored requirements and domains become inevitable.

- Abstraction: To accommodate with change, separate concerns and achieve decoupling, abstraction is considered as a fundamental architectural principle. While designing SOA, abstraction eases the functional decomposition. Moreover, the significant increase of abstraction and encapsulation provided by software components supports the concept of service orientation by abstracting functionalities as service capabilities (cf. Krafzig et al., 2005, pp. 16–17).

- Consistency: This principle has always been seen as main goal of any architecture. It is often that the two terms: consistency and reusability come always together. In SOA, consistency is achieved by providing services that can be utilized in every part of an enterprise or even across enterprises. The variety of functions the services provide enables them to be utilized by several families of enterprise solutions. Therefore, designing SOA has to take into account developing business capabilities in a way they can be easily invoked by a wide range of business processes (cf. Rosen et al., 2008, p. 31).

- Business derivation: The most significant architectural principle that aligns business and IT is the business derivation principle. It assures that the only reason behind the designed SOA is to achieve enterprise goals and support its business strategies (Shishkov et al., 2006). Therefore, enterprises ought to be the only resource where the requirements for designing SOA are extracted and the resulted artifacts applied to the enterprises have to meet these requirements.

- Patterns: This architectural principle is always considered as a powerful tool to describe architectures. A pattern represents merely a template for a solution that has a specific requirement set. Christopher Alexander[36] is considered the spiritual father behind the evolution of patterns. The most notable definition of a pattern is: “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use the same solution a million times over, without ever doing it the same way twice” (Alexander, Ishikawa, & Silverstein, 1977, p. X).

- Facilitation: Generally spoken, an architecture must facilitate building solutions that are related or conformed to that architecture. This is usually referred to as facilitation or conformance. In other words, architecture is not only seen as a descriptor of what the system behind does rather it has also to facilitate all the possible methods and means to build the system and all of its components (cf. Rosen et al., 2008, p. 32).

- Communications: The last architectural principle that clears the ambiguity behind architectures is the communication principle. This means that the architectures have to offer the involved staff proper mechanisms to communicate and get common understandings of IT systems (Colombo et al., 2005).

If the aforementioned architectural considerations are employed correctly while designing SOA, the ambiguities that might appear at the different degrees of abstractions are then cleared. Moreover, these principles place SOA in proper contexts to be comprehendible by each involved stakeholder.

### 3.1.3 Web Service Technology

Web Services represent relatively new means that promote integration of computer systems across the boundaries of organizations and applications. To realize service-oriented architectures in inter-organizational scales, the conventional distributed compu-

[36] Just for clarification, Christopher Alexander is a professor of architecture at UC Berkeley. His research is concerned with buildings and urban planning not software architecture.

ting technologies are not sufficient because they are centralized and managed often by a single company and considered more as intra-enterprise technologies. Instead, they need agreement on means or platforms that can promote kind of global workflow for the entire business processes. In the context of inter-organizational integration, the Web technologies and standards become more beneficial. They are considered as the essential elements to achieve integration in B2B environments. One of the early Web enablement means were the application servers (cf. Oberle, 2005). Web Services extends the Web enablement and considered as a powerful technology to achieve integration by promoting the loose coupling of software functionalities and providing well-defined programmatic interfaces (cf. Lamparter, 2007, pp. 16–17).

Based on the definitions of (Lamparter, 2007, p. 17; Papazoglou, 2008, p. 5), this work defines a Web Service as follows:

**A Web Service** is a distributed self-described and self-contained software module identified by a URI to accomplish a task, solve a problem, or handles transaction over a network (like Internet) as a representative of user or computer program. Transactions with Web Services are performed using XML-based messages carried out by Internet protocols. Each Web Service has public interface described in XML and can be discovered over the network by users and applications.

Table 3.1 depicts the layers of the Web Service technology stack. It includes the main standards and specifications that enable and describe all the aspects of Web Services.

Source: **(Papazoglou, 2008, p. 33)**

**Tab. 3.1:** Web Service Technology Stack

| Layer | Standard |
|---|---|
| **Process** | - WS-BPEL<br>- WSCI<br>- WS-CDL |
| **Discovery** | - UDDI |
| **Description** | - WSDL |
| **XML-Messaging** | - XML-RPC<br>- SOAP<br>- XML |
| **Transport** | - HTTP<br>- SMTP<br>- FTP |

The table above might contain further specifications. However, this section introduces the most popular ones that are related in a way or another to this work[37].

In the transport layer, the main notable protocols are HTTP, SMTP, and FTP. HTTP is used to facilitate communications among Web Services. Other protocols like Simple Mail Transfer Protocol (SMTP) or File Transfer Protocol (FTP) can still be used for the same purpose.

The second layer in the Web Service technology stack is the XML-Messaging layer that contains specifications like: XML, SOAP, and XMP-RPC[38]. These specifications are used to describe and validate the messages exchanged between Web Services. The description layer includes WSDL as the main standard for describing the functionalities of Web Services. It is an XML-based standard that provides machine-readable descriptions of how to call a Web Service, what input parameters it requires, and what output data structures it returns. The discovery layer includes the UDDI as a registry standard to facilitate the processes of Web Services publishing and discovery. The top layer in the Web Service technology stack is the process layer in which many standards can be used to provide mechanisms for Web Service orchestration and choreography. One of the main standards used for orchestrating Web Services is the Web Services Business Process Execution Language (WS-BPEL) (Jordan et al., 2007). It is an OASIS standard executable language to compose Web Services and serves more to achieve better degree of data mediation. To achieve process mediation on the behavioral level of Web Services, standards like: Web Service Choreography Description Language (WS-CDL) (Kavantzas et al., 2005) and Web Service Choreography Interface (WSCI) (Arkin et al., 2002) can be used among other standards to facilitate such task.

The following explains the main SOA principles. These principles had been defined and quoted[39] from the encyclopedia of service design in (Erl, 2007) and later updated in (Erl, 2009). These principles are: standardized service contracts, loose coupling, abstraction, reusability, autonomy, statelessness, discoverability, and composability.

As mentioned in Section 2.2, each Web Service is composed of a service contract and a service implementation. **Standardized service contracts** means that the "Services within the same service inventory are in compliance with the same contract design standards". In other words, service contracts are used by a service to express firstly the purpose behind the service and secondly to express its main capabilities (Turner, Budgen, & Brereton, 2003).

[37] A detailed introduction of all of the Web Service specifications can be found online at: http://www.ibm.com/developerworks/views/webservices/libraryview.jsp?type_by=standards

[38] RPC stands for Remote Procedure Call.

[39] All the definitions in this section are taken from these two listed references of Thomas Erl: (Erl, 2007) and (Erl, 2009).

**Loose coupling** is widely considered as one of the most fundamental SOA principles. It means that "service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment". Therefore, services are designed in a way that the dependencies between service contract, implementation, and consumer are reduced or loosened as much as possible (Josuttis, 2007).

**Abstraction** of services means that "service contracts only contain essential information and information about services is limited to what is published in service contracts". Therefore, it is always a good practice to design services by hiding as much underlying details as possible. This assures and complements the previous loose coupling SOA principle. Therefore, the avoidance of providing any unneeded service information and meta-data is important in this principle (Erl, 2005).

**Reusability** means that the services can "contain and express agnostic logic and can be positioned as reusable enterprise resources". This means that the logic behind the services is designed to be reusable and highly generic. Moreover, the service contracts in this sense have to be extensible and generic as well and finally the services behind are designed to be reused and accessed simultaneously sectors (Dan et al., 2008).

**Autonomy** or service independence means that the "services exercise a high level of control over their underlying runtime execution environment" (Erl, 2009). This means that services have to contain their logic apart of any external influence. In other words, services are designed to be more isolated. This increases service reliability and trigger its behavioral predictability (Daigneau, 2011, p. 15).

**Statelessness** means that the services "minimize resource consumption by deferring the management of state information when necessary". Hence, the deferral extensions management must be incorporated within service design to reach three goals: increasing service scalability, supporting agnostic logic design, and last but not least improving service reusability (Papazoglou, 2008, pp. 15–16).

**Discoverability** is another important SOA principle. It indicates that "Services are supplemented with communicative meta-data by which they can be effectively discovered and interpreted". Appropriate meta-data about the service's purpose and capabilities are provided in service contracts. These contracts are designed to be discoverable to both humans and other service clients. To be discovered, the service contracts are normally published in one or more service registries (Arsanjani, 2004, p. 3; Bean, 2009, p. 37).

**Composability** is the last SOA principle. It means that "Services are effective composition participants, regardless of the size and complexity of the composition". As a design consideration, there is always high possibility that a service can join in various composition scenarios to produce new services that are able to solve bigger problems and this in turn boosts the reusability principle (Katzan Jr, 2008, p. 131).

The aforementioned principles make from Web Services good technology to overcome application and middleware heterogeneity problems.

Next section explains the business value of SOA by introducing the concept of business-driven or enterprise SOA.

## 3.2 Enterprise SOA and Other Architectures

Business-driven or enterprise SOA is merely representing the concept where bunch of independent services can be utilized efficiently in many styles to deliver multiple, high-level business services and workflow-based business processes. To achieve this, a combination of technology and business is always required. Business services have to be composed out of other services published in an enterprise's service repository. These business services have to be fit into the enterprise's business model and to be effective part of it. The main goal of business models in enterprise SOAs is to create sets of services that can produce higher business values. If enterprise SOA is implemented without having a powerful business model, it will end up with bunch of incompatible services that have short lifetime. Therefore, business models have to have higher priorities at the design time to provide higher-level values to the enterprise SOA solutions (cf. Rosen et al., 2008, p. 43).

The following two subsections compare enterprise SOA with Enterprise Architecture and Software Architecture respectively. This comparison is made mainly to show that SOA can be integrated with both architectures without having serious integration incompatibilities.

### 3.2.1 Enterprise Architecture

Similar architecture to be compared with enterprise SOA is the Enterprise Architecture (EA). It is composed of several enterprise components that represent different business entities, their properties and the relationships among these components. EA divides enterprises into manageable pieces. These subdivisions can be architectural perspectives, views, domains, sub-architectures, or perspectives. Generally, EA can be broken down into business architecture, information architecture, application architecture, and technology architecture. Based on (Alwadain et al., 2011; Rosen et al., 2008, p. 44; Kistasamy, Van Der Merwe, & De La Harpe, 2010), there are a lot of similarities between EA and enterprise SOA. The similarities between these two architectures can be summarized as follows:

- The business architecture of EA can be mapped to the business model of enterprise SOA.

- The information architecture in EA can be mapped to the common semantics and data model of enterprise SOA or more precisely to its semantic information model.
- The EA application architecture can be mapped to most of the enterprise SOA-based solutions including integration services and enterprise business processes.
- The EA technology architecture can be mapped to the service bus of enterprise SOA and defines how the service infrastructure meets all SOA-related issues like distribution, binding, security, performance and so forth.

As a conclusion and if an organization has applied an EA and wants to introduce enterprise SOA, there will be no big incompatibility in integrating EA and SOA based on the previous comparison as a supporting evidence.

### 3.2.2 Software Architecture

Another architecture that can be compared with enterprise SOA is the Software Architecture. As defined by (Bass, Clements, & Kazman, 2003, p. 21), the Software Architecture is “the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them”. One of the most common Software Architecture approaches is the “4+1Views” approach that was primarily developed by Philippe Kruchten (Kruchten, 1995). This Software Architecture approach is composed of five views namely: logical, component, process, physical, and use cases views.

The logical view includes the system’s object model supported by class, collaboration, and sequence diagrams. The component view includes the system’s files and dependencies supported by component diagrams. The process view includes the system’s processes and threads supported by deployment diagrams. The physical view includes the system’s applied network topologies and its diagrams. Finally and as its name indicates, the use case view includes the system’s main business scenarios supported by bunch of use case diagrams.

The comparison between the Software Architecture and enterprise SOA is based on comparing SOA to the “4+1 Views” approach (Ionita, Florea, & Jelea, 2009; Rosen et al., 2008, pp. 49–50; Zimmermann, Koehler, & Leymann, 2006). This comparison can be summarized as follows:

- The logical view map to the service design.
- The component view maps to the main interactions between services.
- The process and physical view map to the implementation of services using specific technologies and their deployment in specific infrastructures.

- The use case view maps to the main scenarios of enterprise SOA that involve either single service or set of services.

## 3.3 Summary

This chapter explains the main aspects of service orientation. It started with showing the main motivations behind SOA. It then detailed the most common architectural considerations that have to be taken in account to design efficient service-oriented architectures. The chapter then explained Web Service technology as one of the most common SOA-enabling technologies. The Web Service technology stack had been presented as well together with the common Web Service principles.

The last part of this chapter was dedicated to introduce enterprise SOA and compare it with similar architectures like Enterprise Architecture and Software Architecture. These comparisons were to prove that enterprise SOA can be integrated with other architectures without having too much difficulty.

After introducing the main concepts and theoretical foundations related to this work, Chapter four is dedicated to explain the research methods that are followed in this work to deliver the SESOA approach.

# 4 Research Methods

This chapter explains the research methodologies that have been used as orientation to deliver the research conducted in this work. The first part of this chapter includes the selected design science research processes in the research domain of business information systems. The second part of this chapter explains the service design process.

## 4.1 Design Science

This section explains the two research processes adopted in this work. These research processes are the design science research methodology by (Peffers et al., 2007) and the design science in information systems research by (Hevner et al., 2004). However, before diving in the details of these two research processes, some words have to be given to the literature review process.

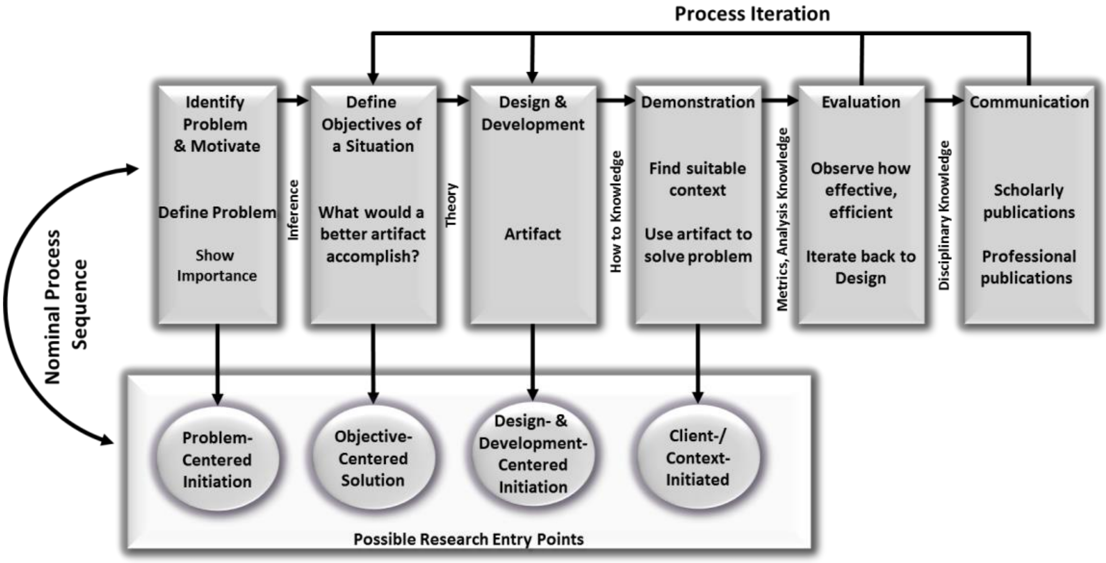
One important part of the research method followed in this work is the literature review process. This process includes reviewing the state of the art in the scientific publications in the SOA, Web Services, Semantic Web, Semantic Web Services, service discovery, semantic conceptual frameworks, and lightweight annotation frameworks. The literature review process is mainly based on the systematic literature review approach proposed by (Webster & Watson, 2002). Based on this approach, an excessive amount of research papers were reviewed on the aforementioned research fields to determine the main problems in the SOA and semantic-enabled SOA solutions. Based on the recommendations proposed by (vom Brocke et al., 2009), this amount has been eliminated to the top 20% of the ranked publications in the domain of information systems. The problem definition had been then initiated based on these ranked publications and that leaded to the process of requirement definition and the other key activities to produce the main aspects of this research.

The following parts of this section are dedicated to explain the two selected research processes in the field of information systems. Each research process is explained and followed by a brief discussion of how it has been followed in this work.

### 4.1.1 DSRM Process Model

(Peffers et al., 2007) proposed a generic design science research methodology (DSRM) for information systems research. The DSRM process model is composed of five main activities as depicted in Figure 4.1.

The first activity in this process model is *problem identification and motivation.* In this activity, the main research problems are identified and well defined to be able to defend the contributions of the resulted solution. On the one hand, this activity is important because it motivates the researcher and encourages the community in the targeted research domain to follow the outputs resulted from the work. On the other hand, this activity supports the argumentation the researcher provides in defining the research problems. The resources needed in this activity are mainly the in-depth knowledge of the main drawbacks in the targeted research domain besides the potential significance of the resulted solution (cf. Peffers et al., 2007, pp. 52–55).


Process Iteration
Identify Problem & Motivate
Define Problem
Show Importance
Inference
Define Objectives of a Situation
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artifact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to Design
Disciplinary Knowledge
Communication
Scholarly publications
Professional publications
Nominal Process Sequence
Problem-Centered Initiation
Objective-Centered Solution
Design- & Development-Centered Initiation
Client-/ Context-Initiated
Possible Research Entry Points

Source: Figure 1 in **(Peffers et al., 2007, p. 54)**

**Fig. 4.1:** DSRM Process Model

The second activity in the DSRM process model is the *object definition*. The objectives that are defined here are derived from the previous activity. The resulted objectives resulted are classified either quantitative if the solution behind could be better than the existing ones or qualitative if the solution behind would solve problems that are not yet addressed by other solutions. The resources needed in this activity are the problems' state knowledge and being familiar with the similar solutions in the targeted research domain (cf. Peffers et al., 2007, p. 55).

The third activity in the DSRM process model is the *design and development*. The theory produced from the object definition activity shapes the main aspects of this activity. The main goal of this activity is to deliver artifacts including models, architectures, constructs, or instantiations (Hevner et al., 2004). The outputs can also include "new properties of technical, social, and/or informational resources" (Järvinen, 2007, p. 49). The novelty has to be embedded in the resulted output (even if it is merely a set of objects). Therefore, the main tasks carried out in this activity are determining the expected functionalities and its architecture from the resulted solution and eventually creating the objects and artifacts that realize them. The resources needed in this activity are the theory knowledge of transforming objectives into artifacts in the resulted solution (cf. Peffers et al., 2007, p. 55).

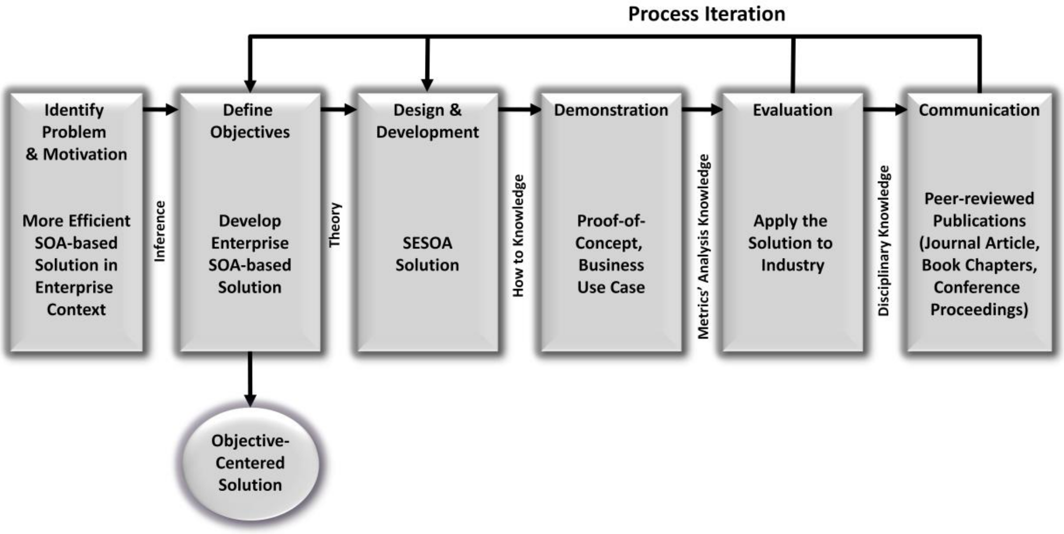
The fourth activity in the DSRM process model is the *demonstration*. This activity includes the steps needed to apply the resulted artifact from the design and development activity to solve one or more of the addressed problems. Potential demonstration can vary from experiments to case studies, proofs, simulations or any other proper demonstration. The resources needed in this activity are the "how to knowledge" to show how the resulted artifact can address the problems in the targeted research domain (cf. Peffers et al., 2007, p. 55). Moreover and as proposed by (Peffers et al., 2007), the applicability of the ideas behind the resulted solution can be proved either in a single act demonstration (Walls, Widmeyer, & El Sawy, 1992) or in a more formal evaluation of the designed artifact (Eekels & Roozenburg, 1991; Hevner et al., 2004; J. F. Nunamaker, Chen, & Purdin, 1990; Rossi & Sein, 2003; Vaishnavi & Kuechler, 2004).

The fifth activity in the DSRM process model is the *evaluation*. It includes both the observation and the measurement whether the artifact from the resulted solution supports well the researched problems or not. Moreover, it is necessary in this activity to compare the solution's objectives to the recent monitored results of applying the artifact in the demonstration phase. Based on the nature of both the problem origin and the resulted artifact, the evaluation can take different paths like comparing the functionalities of the artifact with the objectives defined in the second activity, feedback of the solution users, or simulation. Evaluation might also contain quantifiable measurement of the system performance (e.g. its availability and response time), or any other empirical proof. As can be seen in Figure 4.1 and upon the completion of this activity, the researcher decides whether it is necessary to get back to the second and third activities to enhance the functionalities of the artifact or to continue to the sixth activity in the DSRM model. The resources needed in this activity are the knowledge and awareness of the metrics and analysis techniques (cf. Peffers et al., 2007, p. 56).

The last activity in the DSRM process model is the *communication*. The main target of this activity is to communicate all the research phases to the scientific community in the targeted research domain. This communication is normally done in form of scholarly research publications. (Hevner et al., 2004) stated that this activity is necessary to disseminate the resulting knowledge behind the conducted research. The main resources needed in this activity are the disciplinary knowledge of how to write and publish coherent publications of the different research parts (cf. Peffers et al., 2007, p. 56).

As depicted in Figure 4.1, the DSRM process model can lead to four possible research entry points namely: problem-centered initiation, objective-centered solution, design- and development-centered solution, and client-/context-initiated solution. If the researcher goes through the aforementioned six activities sequentially, the resulted research entry point is a problem-centered approach. This means that the model has to start exactly from the first activity until the sixth activity. This approach is normally resulted from observing a research problem or from getting ideas from the suggestions of future directions from previous researches. If a researcher decides to start the DSRM process model from the second activity, the research entry point is an objective-oriented solution (this is the case in this work). Such solution is normally activated by industrial or research needs that can be collected and identified to develop the resulted artifact. If the DSRM process model started from the third activity, the resulted research entry point is a design- and development-centered approach. This approach normally originates from an existing artifact that had not been formally placed in a solution to solve the problems in the domain to which it can be applied. This artifact could also be resulted from another research domains or it might be even applied to solve the problems in those domains. Eventually the last research entry point results when the researcher starts the DSRM process model from the fourth activity. A client-/context-initiated solution might originate from monitoring a practical solution that already runs. The researcher gets back to produce more rigor solution after taking into consideration the drawbacks of the existing ones and making the best use of consulting experience (cf. Peffers et al., 2007, p. 56).

Based on the abovementioned explanation of the DSRM research process, the rest of this section explains how this process is mapped to this work. Figure 4.2 shows the main activities of this process.


Process Iteration
Identify Problem & Motivation
More Efficient SOA-based Solution in Enterprise Context
Inference
Define Objectives
Develop Enterprise SOA-based Solution
Theory
Design & Development
SESOA Solution
How to Knowledge
Demonstration
Proof-of-Concept, Business Use Case
Metrics' Analysis Knowledge
Evaluation
Apply the Solution to Industry
Disciplinary Knowledge
Communication
Peer-reviewed Publications (Journal Article, Book Chapters, Conference Proceedings)
Objective-Centered Solution

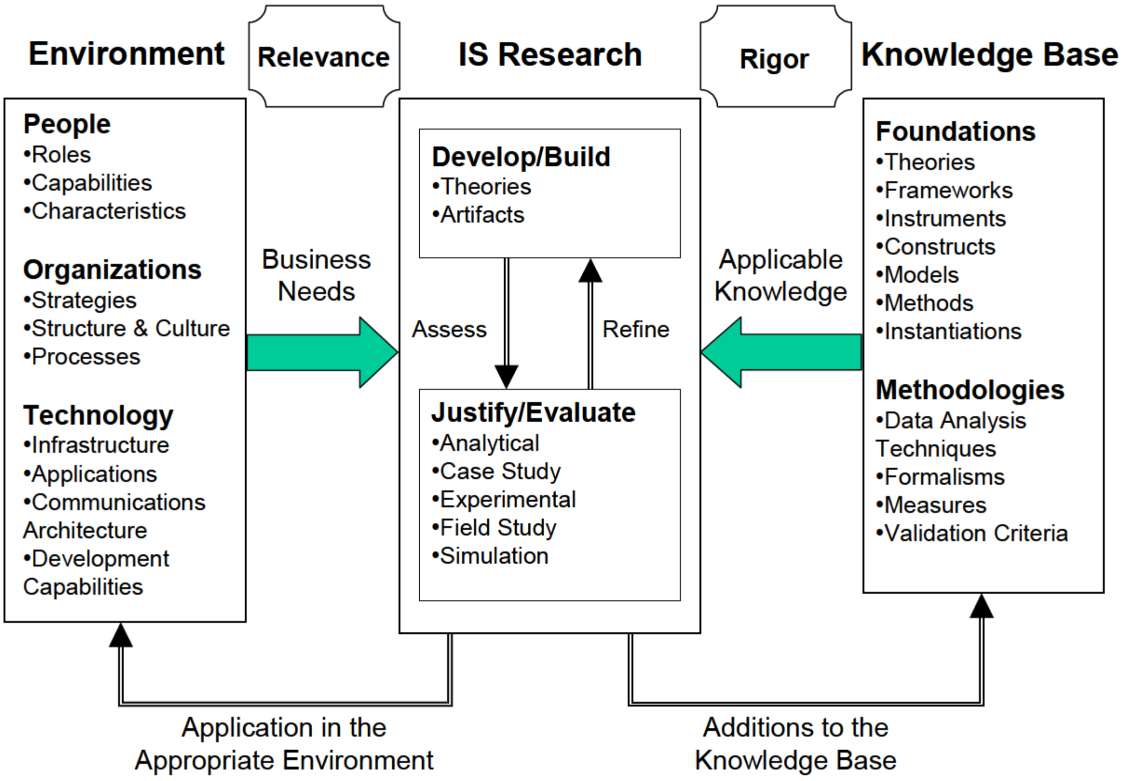
Based on **(Peffers et al., 2007, p. 54)**

**Fig. 4.2:** DSRM Process for SESOA

As can be seen the figure above, the first activity is to identify the main problems in the considered field of study. This includes the problem of service discovery and the main drawbacks of the Web Service-enabled business processes and service repositories. This forms the motivation behind this work in providing more efficient SOA-based solution in enterprise context supported by semantic annotation of Web Services relations in a semantic service repository. This infers the second DSRM process model activity that defines the research objectives and system requirements. The work's objectives have been listed in Chapter one together with the wider research goal to develop a semantic-enabled enterprise SOA-based solution. This facilitates the design and development activity to create the SESOA solution. The demonstration activity represents the proof-of-concept in which the SESOA artifact has been prototypically implemented together with an accompanying business case. This prototypical implementation is necessary to show how the research objectives and the system requirements are met. The resulted prototypical implementations from this work are then evaluated in the evaluation activity by applying it to different business domains and applications. Finally, the communication activity introduces the main concepts in this work to the scientific community. This activity is actually achieved by writing number of peer-reviewed publications in conference proceedings, books, and journals (these publications are listed among others in the Publications Section at the end of this dissertation).

### 4.1.2 Information Systems Research Framework

Besides the research process explained in the previous section, the research behind this work employs the design science approach in information systems discipline proposed by (Hevner et al., 2004). This approach is widely considered as one of the main reference sources in the whole information systems research because of its high quality, guidelines and criteria it presents in the design science. The conceptual framework of the proposed approach is depicted in Figure 4.3.



Source: Figure 2 in **(Hevner et al., 2004, p. 80)**

**Fig. 4.3:** Information Systems Research Framework

The *environment* represents the targeted problem domain (Simon, 1996) where the research interest resides. As can be seen in the figure above, the environment includes *people, organizations*, and *technology* (Silver, Markus, & Beath, 1995). The business needs are derived from the environment's problems, goals, tasks, and motivations that are comprehended by the organization's people. Such comprehension is normally enriched by the people's *roles, capabilities*, and *characteristics*. The organization's running business *processes* together with its *strategies, structure*, and *culture* assess the business needs. Based on that, it has to be determined which technology *infrastructure, applications, communication architectures*, or *development capabilities* will be required to define the *business needs*. Altogether the three ingredients of the environment section from Figure 4.3 define the problem (the business need) that the researcher comprehended (cf. Hevner et al., 2004, p. 79).

Based on the business needs, (Hevner et al., 2004, pp. 79–80) argued that the *Information Systems (IS) research* can be done following the two complementary behavioral and design sciences. While the behavioral science's goal is truth, the goal of the design science is utility. On the one hand, behavioral science develops and justifies theories that explicate the aspects related to the comprehended business needs. On the other hand, design science builds and evaluates the artifacts that are developed to meet the comprehended business need. As illustrated in Figure 4.3, *justify/evaluate* activity drives the research assessment to identify the main drawbacks of the *theory* or the *artifact* in order to be refined and reassessed. Therefore, they are seen in the research method inseparable (cf. Hevner et al., 2004, p. 80).

The *knowledge base* in the right side of Figure 4.3 provides the *foundations* and *methodologies* to complete the IS research. On the one hand, foundations are useful in the *develop/build* phase of the IS research to take benefit from former *theories, frameworks, instruments, constructs, models, methods*, and *instantiations*. On the other hand, methodologies that include *data analysis techniques, formalisms, measures*, and *validation criteria* are considered as the main guidelines resources for IS research. As a conclusion, IS research is considered rigor when the foundations and methodologies that compose the knowledge base are applied appropriately (cf. Hevner et al., 2004, p. 80).

Based on the abovementioned explanation of the information systems research framework, the rest of this section is going to explain how this research process is followed in this work. The start point is identifying the business needs to ensure that the goal of relevance in the conducted research is achieved. Therefore, the organizational strategies, existing business processes, structures and culture elements needed to develop SESOA system are examined. It is also important to define the target organizations to which this system is relevant. The business processes used to deliver the artifacts of this research are more concerned with ERP processes and more precisely, the selling business process has been selected to demonstrate the relevance of this work to the target businesses. The infrastructures and applications that shape SESOA business needs include storing customer information in the system's database besides having proper connection to the other external available resources. Moreover, the resulted artifact is composed of services belonging to different business domains and developed as standardized XML-based Web Services. The applicable knowledge to this work include reviewing most of the relevant work published in high ranked scholarly publications about SOA-based solutions, service discovery and repositories, lightweight semantic annotation mechanisms, Web Service semantic conceptual frameworks, and their execution environments. This step is necessary to assure that the conducted research is rigor.

In order to make this research following the design science, it has to produce and evaluate a novel artifact (Hevner et al., 2004). This means that the IS research has to be as cumulative as possible in a way that it is built on the existing similar researches (Kuechler & Vaishnavi, 2008). The *develop/build* activity in this research is obtained by reviewing and considering service discovery and adding semantics to service repositories (Sivashanmugam et al., 2003; Srinivasan et al., 2004), most of the existing Web Service conceptual frameworks like (De Bruijn et al., 2005; Martin et al., 2005; Battle et al., 2005a), their semantic execution environments (Haller, Cimpian, et al., 2005; Elenius et al., 2005), lightweight semantic annotation mechanisms for Web Services (Kopeckỳ et al., 2007; Vitvar et al., 2008; Kopeckỳ, Vitvar, et al., 2008), and other related IS researches (Medjahed, 2004; Lamparter, 2007; Brehm, 2009).

All of the abovementioned researches are considered either as heavyweight semantic solutions focusing on providing semantics first or as lightweight solutions focusing on extending the service description languages. Moreover, the drawbacks of service discovery and service repositories problems have been investigated. The main distinction between the research done in this work and the other existing researches is that SESOA gives the possibility to add the semantic annotation to the services' relations not to the services themselves to enhance performance and to trigger interoperability. Moreover, SESOA tries to overcome the traditional service repositories problems by providing lightweight semantic service repository in which service relations are semantically annotated using RDF statements. This enables the creation of a lightweight semantic-enabled enterprise Web Service-based artifact making this research cumulative and novel at the same time.

To make the resulted artifact from SESOA relevant for the practitioners side by side with making it contributing to the IS knowledge base to be rigorous, it has to pass tight evaluation and justification. This *justify/evaluate* activity has been achieved by implementing an ERP case study on top of the SESOA system. This case study represents how the selling business process can take benefit from the SESOA functions to realize a semantic-enabled Web Service-based online shop. Based on the results that had been collected during the development phase of the resulted artifact, different cycles of justifications have been made based on the publication's feedback coming from the scientific community side by side with the feedback coming from the targeted practitioners. Finally, more information about how this IS research has been evaluated from industrial and scientific perspective can be found in Chapter 7.

Moreover, all design-science research guidelines proposed by (Hevner et al., 2004, p. 83) have been followed in this research from designing an artifact to problem relevance to design evaluation to research contributions to research rigor to design as a search process ending up with communication of the conducted research.

To sum up, the business environment and more precisely the enterprise systems environment creates the business need behind this IS research. Based on the existing IS researches in this field, this work applies the knowledge harvested from these researches to perceive the problem scope for the development of the SESOA artifact. This research can add to the knowledge base and at the same time proves its applicability in an appropriate business environment (cf. Hevner et al., 2004).

## 4.2 Service Design Process

The previous section gave insights to the main selected research deign processes in information systems. This section provides more low-level information concerning the core elements behind the artifact of this research: services and more precisely Web Services. As defined in the W3C Web Service Glossary: “a service is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities. To be used, a service must be realized by a concrete provider agent” (Haas & Brown, 2004). Based on this definition, it is quite important from a methodological perspective to have a proper service design process compatible with the aforementioned research processes.

(Rosen et al., 2008, pp. 106–109) discussed in their book the three main service design process approaches namely: top-down, bottom-up and middle-out approaches. The following subsections give some highlights to each of these approaches. This section then concludes with a recommendation of which service design approach is selected.

### 4.2.1 Top-Down Approach

As it can be comprehended from its name, the top-down approach starts from the top of the enterprise that constitutes the wide scope to end up down with specific enterprise issues. This service design approach deals usually with the whole set of enterprise needs over the time. It is concerned of how to adapt specific solution in the enterprise by making proper adjustment on it to suit its strategy and roadmap. Moreover, it can also be considered as a design approach to manage any project in the enterprise together with its set of tactical requirements. Enterprise system analysis and business process models are considered as the most common top-down approaches (cf. Bean, 2009, p. 143).

On the one hand, enterprise system analysis deals with enterprise requirements and business needs to build service repository and initiate a plan to identify which services are required, the main entities to which these services can be applied, and the service groups that support the enterprise's business goals over time (cf. Daigneau, 2011, p. 15). On the other hand, another way to handle the enterprise requirement is to develop business process models that are usually composed of multiple tasks. With a proper design of the business process models, its activities are realized as business services that can be reused across enterprise borders. From this perspective, business process models are often project-based and they can be seen as an extension to the enterprise system analysis or a more detailed level of the overall business architecture. As a conclusion, the top-down approach represents proper comprehension of the enterprise context and plans. It integrates the enterprise roadmap into project-based service design. Therefore, the top-down service design approach can be applied either on the enterprise level or on one of its specific projects.

### 4.2.2 Bottom-Up Approach

In contrast to the aforementioned top-down approach that starts from the enterprise perspective, the bottom-up service design approach starts normally from narrower perspectives like the already running systems, technologies, or shared services. It is usually more concerned with specific projects that have instant set of requirements and must be realized on a project not on the enterprise level. Utility services and service enablement are considered as the most common bottom-up approach (cf. Bean, 2009, p. 149).

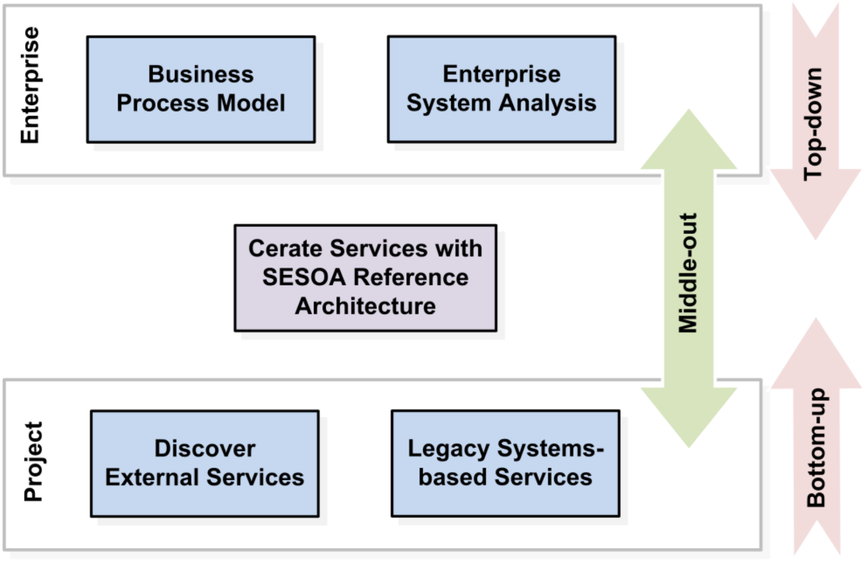
It is often the case that each enterprise has set of common utilities that are used in all of its organizational units. Such set can help the service developers trying to realize these utilities in form of services and this is exactly the motivation behind the utility services. Examples of utility services can vary from credit card validation to address checking to any other common day-by-day transactions. However, such utility services are usually not discovered, underutilized, or not yet hosted in a project and need to be integrated and here comes the importance behind the utility services approach. The other bottom-up service design approach is the service enablement that transforms the legacy systems data and functions into services. However, this approach is considered short-sighted because it is often in conflict with enterprise requirements (cf. Papazoglou, 2008, p. 650). As a conclusion, one of the bottom-up design approach problems is that it is limited to a specific scope and can't consider wider enterprise context. Therefore, this approach is only used when existing systems need to be integrated or a project needs to be realized at the earliest possible opportunity (cf. Karakostas & Zorgios, 2008, p. 22).

### 4.2.3 Middle-Out approach

The top-down approach is considered impractical in providing more value to the enterprise including many unnecessary details. On the other hand, the bottom-up approach is so specific to the point that it might deliver services that are rarely reused in the enterprise's sections. This is somehow in conflict with the SOA reusability principle. Therefore, it looks like none of these two approaches is perfect and the need for another com-promise service design approach is inevitable. This approach is called the middle-out approach (Usländer, 2010, p. 126). It tries to overcome the drawbacks accompanied with the other two approaches by pushing up towards the enterprise scope and down towards the immediate project scope. This is done by creating higher-level business and information architecture in which the service design process includes activities, artifacts, repositories, and governance.

Activities have different goals and can be carried out in the business modeling, service identification, design, and implementation. The artifacts represent the outputs of the activities and can vary from specifications to models to executable codes to test plans. Repositories are merely the storage means in which the artifacts can be stored and managed. Last but not least, the governance is the aspects that manage the activities, the artifacts, and the repositories. Proper governance indicates for example which artifact is needed at which section and where it is stored and managed (Braude, 2004).

Based on the abovementioned explanation of the different service design approaches (see Figure 4.4), the rest of this section is going to explain which of these approaches is chosen to be adopted in SESOA to design its services.


Enterprise
Business Process Model
Enterprise System Analysis
Top-down
Cerate Services with SESOA Reference Architecture
Middle-out
Project
Discover External Services
Legacy Systems-based Services
Bottom-up

Based on Figure 3-7 in **(Rosen et al., 2008, p. 107)**

**Fig. 4.4:** Service Design Approaches

The central part of the drawing above indicates that the services are created using the SESOA reference architecture. The main service design approach that is used to create these services is the middle-out approach. This approach takes into consideration the requirements that are identified both at enterprise and project levels. It represents the joint between the top-down and the bottom-up service design approaches. While enterprise system analysis and business process model approaches are applied at the enterprise scope, legacy systems-based and external services can be integrated at the project scope. These considerations enhance the overall system design and make it capable of being extended or enhanced at any point in the future.

Adopting middle-out approach assures that the implementation of the identified services is in-line with the Enterprise Architecture and suits its business models. Moreover, it provides an overall SOA-based solution that supports service reuse.

## 4.3 Summary

This chapter illustrated the two research methods in information systems science that have been followed to manage the research behind this work. The design science research method has been followed as the core method in this research. SESOA represents the main research entry point that is resulted from following the DSRM method in this work. SESOA is considered as an objective-centered solution. The other research method followed in this work is the information systems research framework in which the conducted research reflects the relevance to the business environment and adds to the knowledge base in information systems science as a whole.

On the service level, three different service design approaches have been investigated. Top-down and bottom-up approaches lacks the efficiency in fulfilling the requirements of both enterprise and its individual projects. Therefore, the middle-out service design approach has been adopted to create SESOA services.

Based on the design considerations derived from the methods and approaches in this chapter, the following chapter initiates the conception phase of this work and defines all of its requirements in details. Moreover, the requirements of an accompanying business case are also defined in Chapter five.

# 5 Conception and System Requirements

One major requirement behind this work is the semantic enablement and organization of Web Services. In any productive enterprise environment, the number of Web Services is gradually changing. These services might be implemented internally within an enterprise or might be supplied by many external service providers. Since these services are coming from different business sectors and realize heterogeneous functionalities, demands for techniques to manage, process, and organize these services need to be met.

This chapter starts with a general definition of the proposed system. Web Service grouping in assemblages together with semantic enrichment of services are then introduced in this chapter. Based on the analysis of the existing SOA-based solutions presented in Chapter one, the general system requirements are then defined. The requirements in this work are split into core system requirements and accompanying business case requirements. These requirements are classified as functional and non-functional requirements. This represents the first stage of the conception phase behind this work. System modeling and the resulted reference architecture are explained in the Chapter six.

The organization of this chapter is as follows: Section 5.1 defines SESOA concept. In Section 5.2, the term Web Services assemblages and the semantic enrichment of Web Services are introduced. Section 5.3 defines the requirements of the proposed system. Brief conclusion that summarizes this chapter is then placed in Section 5.4.

## 5.1 Definition of Lightweight Semantic-enabled Enterprise SOA

As introduced in (Mahmoud, Marx Gómez, & von der Dovenmühle, 2011; Mahmoud, Petersen, et al., 2012; Mahmoud & Marx Gómez, 2010; Mahmoud, 2009): **Lightweight Semantic-enabled Enterprise Service-Oriented Architecture (SESOA)** is an enterprise solution based on Web Services. It links IT-System landscape to external systems based on business process and SOA concepts. It represents a lightweight Web development system that annotates the Web Services coming from different service providers with semantics. Thereby, the indexing and discovery of these services can be more dynamic and comprehensive.

SESOA can be considered as an enhanced SOA concept in which the user system (service consumer) consumes the provider’s services after discovering them in the SESOA’s “semantic service repository”. The Web Services are published in this repository after being annotated using RDF statements. These services can be queried using any RDF query language like e.g. SPARQL (Prud’Hommeaux & Seaborne, 2008). Furthermore, SESOA enables the traditional Web Service discovery provided by former approaches like the common UDDI registry (Clement et al., 2004).

In this way, SESOA dynamically adapts to potential changes that might accompany any enterprise requirements, goals and visions. This is realized by easing the phases of Web

Services discovery, selection, invocation, composition and monitoring in a distributed and open environment.

## 5.2 Semantic Support of Web Services

Semantic enablement of Web Services is one major output of this work. RDF statements are utilized to enrich Web Services with semantics. This section explicates the definition of the term Web Service assemblage (WS-assemblage or assemblage) and depicts the semantic support of Web Services.

*A WS-assemblage is a holder or mini repository that groups Web Services coming from different providers.* This grouping is based on the business sectors or domains to which these services usually belong.

All Web Services of one WS-assemblage have the same business classification. The relations between services and assemblages are semantically annotated using RDF statements. The assemblage's attributes that are going to be detailed here in this section are similar to the so called: "communities" proposed by (Medjahed, 2004) in his dissertation: "Semantic Web Enabled Composition of Web Services". While these "communities" serve in his work to facilitate the dynamic service composition, assemblages in this work are more related to the design aspects in classifying services based on the business domain to which they belong. As depicted in Figure 5.1, The WS-assemblage has the following properties (cf. Medjahed, 2004):

- *ID:* Is an identifier that contains a unique name and a text description. It indicates the assemblage's characteristics.

- *Category:* It specifies the WS-assemblage's classification within a specific business domain. All the Web Services that belong to one WS-assemblage have the same category. The categories are accessible to all service providers in order to let them registering their services within it.

- *Members:* It refers to the list of WS-assemblage members. Being member of a WS-assemblage, service providers assure that their Web Services are registered in the respected assemblages.

- *Operations (Properties):* Are abstract operations that depict the main functions that the WS-assemblage's members have to specify. The Web Service developer in the proposed architecture has the duty of defining WS-assemblage operations. These operations are implemented by the WS-assemblage members (i.e., the actual Web Services) that are interested in offering the functionalities assigned by these operations. The operations (properties) can be classified as syntax, semantic and qualitative operations.

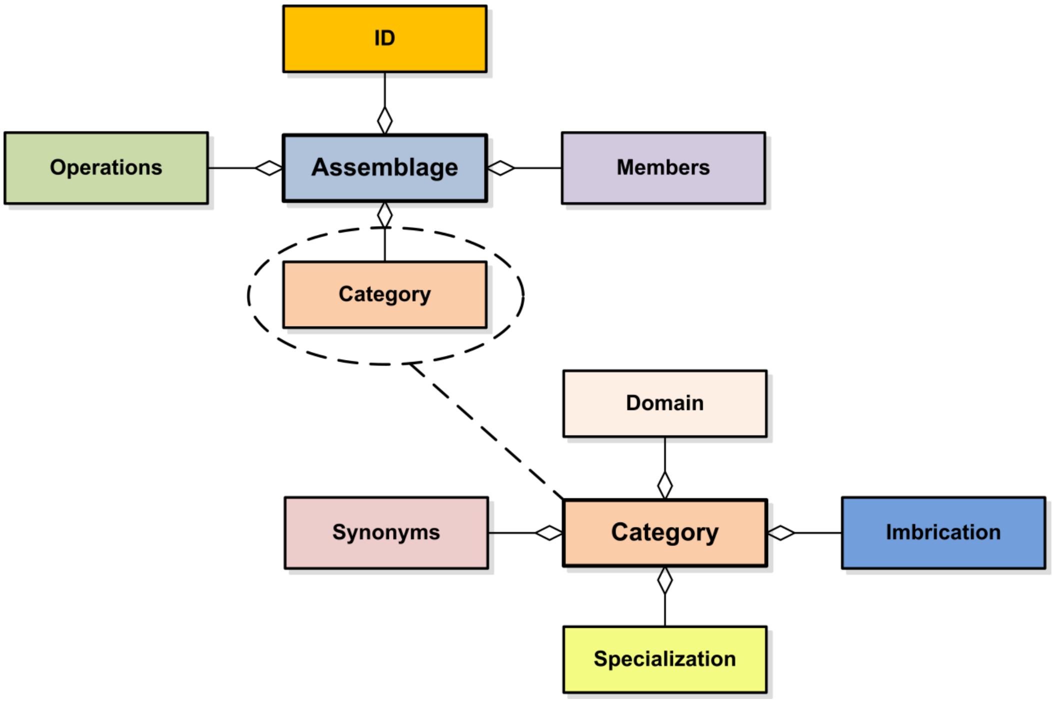


**Fig. 5.1:** Web Service Assemblages

Figure 5.1 above shows that the WS-assemblage's category has four properties: *domain*, *synonyms*, *specialization*, and *imbrications* (cf. Medjahed, 2004). A category's *domain* represents the main WS-assemblage's business area (e.g., financing). It is possible that different WS-assemblages adopt different taxonomies to define their category attribute. *Synonyms*, the second category's property, contain similar or optional domain names for a WS-assemblage. For example, "funding, business finance and public finance" are synonyms of "financing". The third category's property is the *specialization* that is a set of WS-assemblage's domain characteristics. For example, "personal finance" and "employee" are specialization of "financing". This means that the WS-assemblage provides personal financing services for employees. The last category's property is the *imbrications*. Since WS-assemblages can generally have something in common, they are linked to each other through relationships specified in the imbrications attribute. It contains the list of WS-assemblages' categories that overlap with each other. For example, an operation that belongs to a WS-assemblage with a category domain "personal finance" might relate to another operation that belongs to a WS-assemblage with a category domain "temporary contract employees". This provides personal finance for temporary contract employees. The Web Service developers are responsible of identifying related categories to assign them to the imbrications property.

Figure 5.2 shows the WS-assemblage's operations. As mentioned before, these operations are classified into syntax, semantic, and qualitative operations. Each of these classes is specified separately. For the syntactic and semantic operations, they are classified as messages, operations and as inter- and intra-operations.

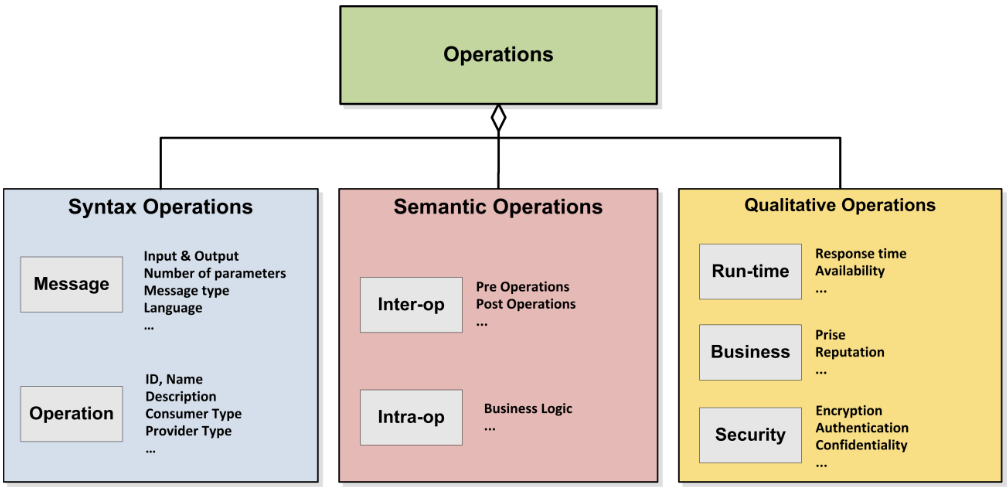


**Fig. 5.2:** Classification of WS-Assemblage Operations

In comparison with the classification done by (cf. Medjahed, 2004), Table 5.1 gives an overview of some of these operations.

**Tab. 5.1:** List of WS-Assemblage Operations

| Operation Type | Property | | Managed By |
|---|---|---|---|
| **Syntax** | Message | Input | Web Service Developer<br>Service Provider |
| | | Output | Web Service Developer<br>Service Provider |
| | | Number of Parameters | Web Service Developer<br>Service Provider |
| | | Type | Web Service Developer |
| | | Language | Web Service Developer<br>Service Provider |
| | Operation | ID | Web Service Developer |
| | | Name | Web Service Developer |
| | | Description | Web Service Developer |
| | | Consumer Type | Web Service Developer |
| | | Provider Type | Web Service Developer |
| **Semantic** | Inter-Operations | Pre-Conditions | Web Service Developer<br>Service Provider |
| | | Post-Conditions | Web Service Developer<br>Service Provider |
| | Intra-Operations | Business Logic | Web Service Developer<br>Service Provider |
| **Qualitative** | Runtime | Availability | Third Party |
| | | Response Time | Third Party |
| | | Reliability | Third Party |
| | Business | Price | Service Provider |
| | | Rating | Third Party |
| | Security/Privacy | | Third Party |

All operation properties (attributes) are listed in the second column. Examples of these properties are ID, name, and description that are operation-related syntax operations. In addition, the numbers of input and output parameters together with the number of parameters are accorded as message-related syntax operations. The third column in Table 5.1 summarizes who is the responsible entity that has the right to assign values to the operation's properties. In this work, there are three different entities that can do this: the service developer, the service provider and third parties. Service developers are one of the system's main actors, who are mainly responsible of developing the system's Web Services. Service providers are the external entities that supply Web Services. The values of security/privacy properties like encryption, authentication, confidentially and non-repudiation are normally assigned to third parties.

Each Web Service is member of one or more assemblages. The relationships between the services and assemblages are semantically annotated using RDF statements. Each RDF statement is realized as a subject-predicate-object triple in which the subject is the assemblage, the predicate is the relationship (e.g. "hasMember") and the object is the Web Service. Each of these parts can be identified using a distinct URI. These RDF statements are published in the system's semantic service repository in which any RDF query language like SPARQL can be used to query services. An important point that has to be taken into consideration is that the semantic querying in this system is taking place at the workflow management system level where the business processes are constructed. This ensures the dynamicity and consistency of the system and opens the possibilities to include up-to-date services in the ever-changing business processes.

## 5.3 Requirement Definition

In this section, the main business needs required to develop the SESOA system are defined. In addition, system goals, boundaries, and sets of functional and non-functional requirements are described.

### 5.3.1 General System Requirements

The starting point for developing SESOA is the existence of several software providers. The common goal of these providers is to provide functionalities in specific enterprise systems forwarded to a specific target group of end-user companies. These functionalities are realized using the Web Service technology. This cooperation of suppliers provides a network that serves wide range of users. The main benefits that describe the nature of SESOA over conventional SOA-based systems are:

1. The support of lightweight semantic annotation of Web Services relations using RDF statements
2. Distribution of software elements over a network
3. Autonomy of service providers through their independent self-administration

4. Assembling Web Services in groups based on their business domains
5. Enabling process orientation by implementing business processes as workflows
6. The support of service reuse

Service specifications together with the grouping and annotation of services have to be listed following the system's information model[40]. Service specification includes service name, description, provided and required interfaces, service protocol, constraints, qualities of service, and the policies for using services.

In the next three sections, functional and non-functional set of requirements are going to be defined together with defining the requirements for the accompanying business case.

### 5.3.2 Core Functional Requirements

The essential base to achieve the agile implementation and development of SESOA is the distribution of its functionalities using lightweight semantic annotated Web Services. This distribution is facilitated by having guidelines of the system's architecture and the system requirements. For this purpose, the first part in this section defines and clarifies the basic concepts in SESOA in order to introduce the requirements definitions in a clearly and comprehensively way. After that, the issues of access control and user management system are explained. Then, the requirements to handle the interactions between system components are illustrated. After that, the general requirements for the entire system from a user perspective are presented. The remaining parts of this section include further requirements of the different SESOA's subsystems like workflow management, GUI, Web Services, semantic annotations and database management.

#### 5.3.2.1 Comprehensiveness: Main Terms

The main terms that are used in developing SESOA are processes, functions, and tasks. Table 5.2 gives distinct definitions of these terms (cf. Deeken et al., 2007, p. 15). In the last three decades, the economy and the strategy of enterprises were relatively static because of many reasons include the stable customers' needs, the clearly defined national and regional markets, durable products, and the well-known competitors. Nowadays, the core of the enterprise's strategy does not exist in the structure of its products and markets, rather in the dynamics of its behavior. This comes from the rapid change in customers' needs and the very fast development and desertion of products, markets and even industries (Stalk, Evans, & Sgulman, 1992). This forms a motivation in this work to have the shift from functional to process orientation.

[40] The information model has to describe the elemental entities and the relationships between them. It has to concern about information, documents and actors to create classes and relationships between them.

**Tab. 5.2:** Processes, Functions, and Tasks

| Term | Characteristics |
|---|---|
| **Process** | ✓ Represents object-related complex sequence in the hierarchy level<br>✓ Realizes the main SESOA functionalities<br>✓ Implemented as a workflow and can be initiated by the user |
| **Function** | ✓ Is a complex activity within a process in the hierarchy level<br>✓ Represents a functionality provided as a Web Service<br>✓ Calling a function represents an activity in a process |
| **Task** | ✓ Reflects an interaction as a simple activity within a process in the hierarchy level<br>✓ Represents an individual user interaction like for example data entry<br>✓ Accomplishing (handling) a task represents an activity within a process |

Process in this dissertation is a kind of cross-referencing between the definition of (Nordsieck, 1934) who was the first one referring to this term in the 1930s and the other definition from (Seidlmeier, 2010, p. 3). Hence, a process is as a mixture of executions and objects or more precisely the executions of objects. It terminologically defines a continuable consequence of informational functions (activities) with precisely defined inputs and outputs. The functions within processes can be defined as occupational services. Finally, the tasks in the resulted prototype represent the elementary functions that implement specific activities that cannot be meaningfully decomposed.

#### 5.3.2.2 User and Access Control Management Requirements

The user and access control management in SESOA permits and allows the administrator to add new users to the system, modify their data, and delete them permanently. Each user in the system has traditional credentials like a user name and a password. With the name/password combination, the user can login at the login interface to the system. Furthermore, the administrator assigns rights to each user in respect to his/her functional role in the system. For this purpose, a role-based access control is provided. It contains pre-defined roles that in turn provide information about which users can perform what tasks. The system has the following types of users:

- *Administrator:* An administrator is the key user (super user) who is responsible of the communication infrastructure within SESOA. He/she can add new users to the system, modify and delete them. Moreover, administrators have full control of all system's functionalities including the management of Web Services, their semantic annotations and business processes administration.

- *Web Service Developer:* This user implements different business functionalities as Web Services. Moreover, he/she groups the services in assemblages based on the business domains to which they belong. Finally, the Web Service developer is responsible of annotating the relations among these services semantically using RDF statements.

- *Workflow Developer:* Workflow developer is responsible of managing all the workflow aspects within SESOA. Creating and modifying objects as services within the developed workflows are from this user's responsibilities. In other words, the main role of this user is to design more agile communication flows. The workflow developer uses a workflow editor to realize the business process functionalities. Designing Workflows is actually based on the main terms explained in the previous section: processes, functions and tasks.

- *End User:* Is the user who can access the GUI and can reach all public documents. This user can be either anonymous or registered user. This means that help files, general system information and exemplary Web Services are reachable to all anonymous users. An anonymous provider can use the Web Service test system to validate its own services to be registered in the system's repository. Furthermore, a provider can read service advertisement entries. Registered users can reach some areas that are not public based on the rights assigned to them by the system administrator. Restrictions can vary following the system's rules and examples of these restrictions can be system usage or explicit access to specific areas/functionalities.

#### 5.3.2.3 Main Interactions among System Components

The following figure gives a broad overview of the system architecture. It shows the various components that make up the whole SESOA system. The main system's key users are depicted in Figure 5.3. The system's core component is the workflow management system that deals with business processes.

Business processes are excluded from the enterprise's application scenarios and translated into a proper workflow format. This is done with the help of the workflow editor component managed by the workflow developer user. This user is responsible of managing the workflows that represent the enterprise's business processes. All the workflows are administered by the workflow management system. As mentioned in Table 5.2, the processes are composed of functions and tasks. The functions are realized as Web Services and semantically annotated using the semantic Web Service annotation system component and published in the semantic service repository. Creating Web Services, annotating their relations with semantics, and publishing them in the semantic service repository are the responsibilities of the Web Service developer user. The semantic service repository, the workflow management system and the semantic Web Service annotation system have links to the system's database managed by the administrator. This user administers the workflow editor and the Graphical User Interface (GUI) that is made available to the end users.

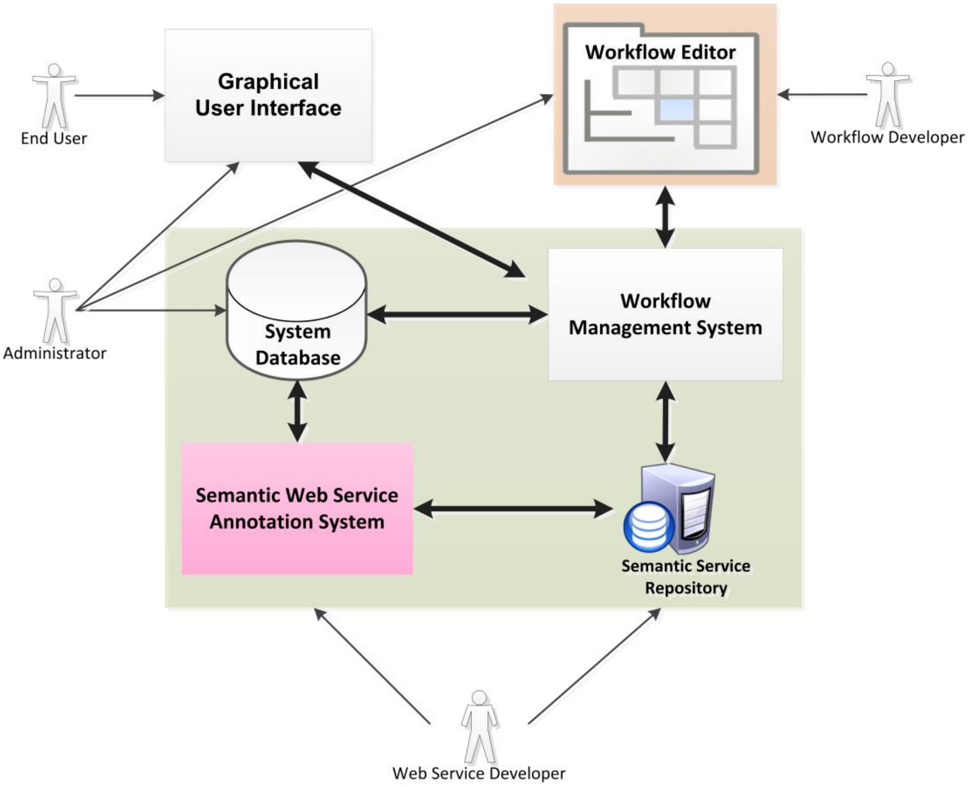


**Fig. 5.3:** Main Interactions between System Components

To understand the system requirements from a user perspective, use case diagrams are employed to represent these requirements. These use cases are illustrated in Figure 5.4.

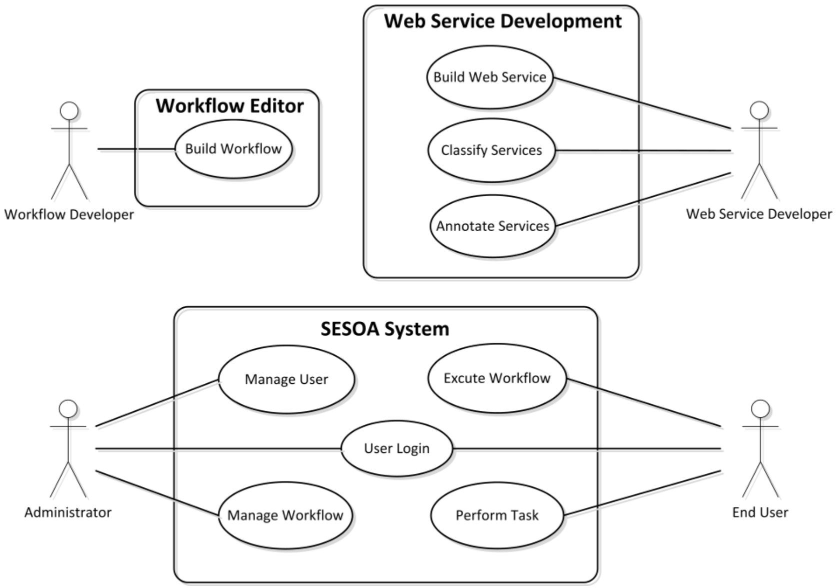


**Fig. 5.4:** System Requirements from User's Perspective

The requirements are grouped into three main systems named SESOA system, workflow editor, and Web Service development. The administrator manages SESOA system available to end users. This system has five main use cases: user login, manage user, manage workflow, execute workflow, and perform task. SESOA system together with the workflow editor and the Web Service development compose the overall solution. The workflow editor is managed by the workflow developer who deals with one use case that is build workflow. The Web Service developer controls the Web Service development and deals with three use cases. These use cases are: build Web Service, classify services, and annotate services. Table 5.3 gives a brief description of all of the system's use cases.

**Tab. 5.3:** The Main System's Use Cases

| Use Case | Description |
|---|---|
| **User Login** | The end user gets the login screen in this generic use case. If the user is registered in the system, his/her username and password combination provide the access to the system. |
| **Manage User** | The administrator is responsible of creating and managing users. Different users with distinctive rights interact with SESOA system. The administrator creates those users and assigns for each of them a user name and a password. Furthermore, users are created with certain rights to reflect different responsibilities. |
| **Manage Workflow** | Workflow management is part of the administrator duties. Creating, updating, and deleting workflows (when they are no longer in use) are done by the system administrator. |
| **Execute Workflow** | Most of the workflow activities in the system are realized as Web Services. The end user can execute workflows and eventually call their Web Services. Moreover, the workflow management system controls calling and locating services. |
| **Perform Task** | Business processes in the system are realized as workflows. Parts of these workflows (tasks) require the intervention of end users. Using appropriate input fields, needed information can be displayed to the end user. To perform a task, the user has to fill in these input fields. |
| **Build Workflow** | The workflow developer is responsible of transforming the enterprise's business processes into workflows using a workflow editor. |
| **Build Web Service** | The system's workflow functions are realized as Web Services. The Web Service developer is responsible of creating these functions as Web Services. Some of the workflows themselves can be also realized as Web Services. |
| **Classify Services** | The Web Service developer classifies the services in assemblages. Only the Web Service developer has the rights to create Web Services. Moreover, managing service assemblages (add, modify, and delete) is part of the Web Service developer responsibilities. |
| **Annotate Services** | After classifying services in assemblages, the Web Service developer annotates the relations between the services and the assemblages semantically using RDF statements. |

#### 5.3.2.4 Graphical User Interface Requirements

Since SESOA is based on Web Service technology, the graphical user interface has been realized as a Web application that is compatible with wide range of Web browsers. The main reason behind that is to reduce the deployment's complexity. Based on the general functional requirements of the entire system from the user perspective, the derived requirements for the GUI are firstly described and then the possible future requirements for its realization are proposed.

*Login Interface View:* User authentication module is used to grant access rights and handles user management. The main argument behind this decision is that many vendors provide it as a standard component. Hence, the user is asked to type his name and password and then press a login button to access the system to the so called "user control" view. Registered users in the system have the following user details: login name, e-mail, password, first name, last name and some optional data.

*User Account Information View:* Like user identification information, user account information is realized within the user login component. Moreover, to decouple the system from this specific component, an interface for inner-system communication is introduced. After successful login, the user control view is displayed to the user in which all the admitted functions are shown. The information about user identity is in the database. This information is forwarded via the workflow management system to the GUI to be shown at runtime in the user control view. User settings are included in this view as well.

*Admin View:* The requirements of the admin interface include viewing the current user, the main groups and roles in the system together with the derived visualization of information about the current security state of the system, i.e. the distribution of access rights to different users, groups and/or roles. All of these are viewable and modifiable using the admin interface. Save and cancel buttons for basic actions in this view are also available.

*Possible Future Requirements:* Some of these requirements include the user information about the current state of the system including system utilization and performance case studies. In addition, buttons like back, cancel, and possibly save for the simple and intuitive navigation can be always presented. Another functional but optional GUI requirement is to provide a multilingual interface with the possibility of changing between different languages.

#### 5.3.2.5 Workflow Management System Requirements

In this section, the main tasks assigned to the workflow management system in conjunction with the other system's components are described. Based on the general functional requirements of the entire system from the user perspective, the derived requirements for workflow management system are schematically defined as follows:

*Verify Credentials:* After the end user enters his/her credentials (user name and password) to login into the SESOA system, it is verified to check whether the given data are correct or not. To do so, the workflow management system compares the input data with the stored data in the database system.

*Start Session:* When the login data are entered correctly, a certain value is returned. Using this value notifies the workflow engine to instantiate a new session. Each registered user runs a session and this requires that these sessions have to be managed. Moreover, the logged in user has a list of the system available processes. However, the displayed processes are just the ones that the logged in user is authorized to run. The workflow management system must provide a function to achieve this.

*Create a Workflow:* Creating a new workflow includes several sub-requirements like the list of involved Web Services and the process of adding RDF annotation to these services. Moreover, many user interfaces have to be designed for the realization of each individual step. An implemented workflow must be able to get the endpoints of the involved services and the assemblages to which these services belong based on the RDF representations stored in the database system.

*Create a Workflow Instance:* From the available list of processes, the user selects a process like “place order”. The workflow engine then starts a workflow instance. Using the provided object data given by the user, individual activities are processed and possible return values are delivered back to the user.

*Modify a Workflow:* A workflow can be modified by changing the process description. It is ensured that the current database can still be adjusted to deal with it.

*Terminate a Workflow:* In the case of inaccurate data inputs or exceptions, there is a possibility to terminate a workflow. As a result, the corresponding workflow instances together with its activities are deleted. Similar to a transaction in database systems, all object properties have to be recovered in order to get the system’s pre-execution state.

*Upload/Delete Workflow:* Workflows can be uploaded and removed. Only users with authorized rights (administrators) are allowed to perform these actions. Deleted workflows are marked as “deleted” but they are still available for system’s coherency purposes (the existing users who dealt with such workflows). However, the deleted Workflows are not shown anymore.

*Assign Activities:* Activities in workflows are specified as functions or tasks (see Section 5.3.2.1). Tasks are the normal data inputs or transactions provided by end users. The functions behind the workflows’ activities are actually realized using Web Service technology. This means that the workflow developer has to specify the properties of the requested Web Service(s). To implement a function (workflow activity) using a Web

Service, specifications of what data - in the context of a workflow instance - are required by each Web Service parameter. Moreover, the mapping between the return values of the Web Service back to the workflow activities has to be specified so that the responses can be reused in other similar activities. The discovery of a desired Web Service is done through the workflow management system that locates the service information in the semantic service repository and binds the workflow with this service.

*Assign a Data Access Activity:* Data need to be retrieved from the database system in the context of a workflow instance and the results from the execution of functions are stored. Access to the data has to meet the ACID (atomicity, consistency, isolation, durability) properties as defined in (Haerder & Reuter, 1983, pp. 289–290).

Finally, since the workflow management system comprises the main communication bridge between all other components in the system, the issues of how to initiate, control, interrupt and terminate processes' instances are specified[41].

#### 5.3.2.6 Web Services Requirements

In this section, the main requirements for Web Services and the semantic annotations of their relations are defined. Based on the general functional requirements of the entire system from the user perspective, the derived requirements for (semantic-) Web Services can be schematically defined as follows:

*Discover Web Services:* After a user runs a specific workflow, all the required Web Services needed in this process need to be looked up in the service repository. Then, the requesting component (workflow management system) gets a list of the requested Web Services with their endpoints (addresses). This list is also provided with all other services' necessary details.

*Create an Assemblage:* Each assemblage is theoretically specified with four properties: ID, category, members and operations. A Web Service is grouped as members in an assemblage that has a distinct ID and category. Services in assemblages implement some or all of its main operations.

*Group Web Services in Assemblages:* While registering a Web Service in one or more assemblages, it is important to firstly load a list of previously registered Web Services with all their related data to avoid duplicating service entries in the repository and respectively in the assemblages. A Web Service can be registered twice[42] in one or more assemblages and published in the semantic service repository. Upon service registration,

[41] Windows Workflow Foundation (WF) is used in the prototypical implementations to manage the system's workflows (cf. Collins, 2009).

[42] If a Web Service is registered twice in one or more assemblages, it must have different IDs to ensure the uniqueness of the Web Services.

RDF statements that annotate relations between assemblages and services as entities are created automatically.

*Define Web Service Description:* Each Web Service has to be well defined before being published in the service repository. This definition is done by assigning RDF triples to the service's and the assemblage's information. Thereafter, each Web Service can be semantically queried by all requesting components.

*Invoke Web Services:* To run a workflow successfully, it is imperative that all necessary Web Services are made available. As soon as their endpoint information is available, the Web Service can be invoked (called) directly by the workflow management system.

#### 5.3.2.7 Validation and Evaluation Requirements

Web Services within SESOA system can be validated and evaluated. From the *validation perspective*, the requirements are:

*Validate Assemblage Data*: Each assemblage utilizes validation tests to validate its data (from technical perspective). A validation test has to check data types and ranges. Incorrect data have to be identified within each test. As a result, the test must identify failures and exceptions (like out of range exceptions) and try to handle them.

*Validate Web Service Data*: A Web Service may allow validation tests to run in order to validate the correctness of its properties. Figure 5.5 illustrates how such validation tests are applied to properties of both assemblages and Web Services.

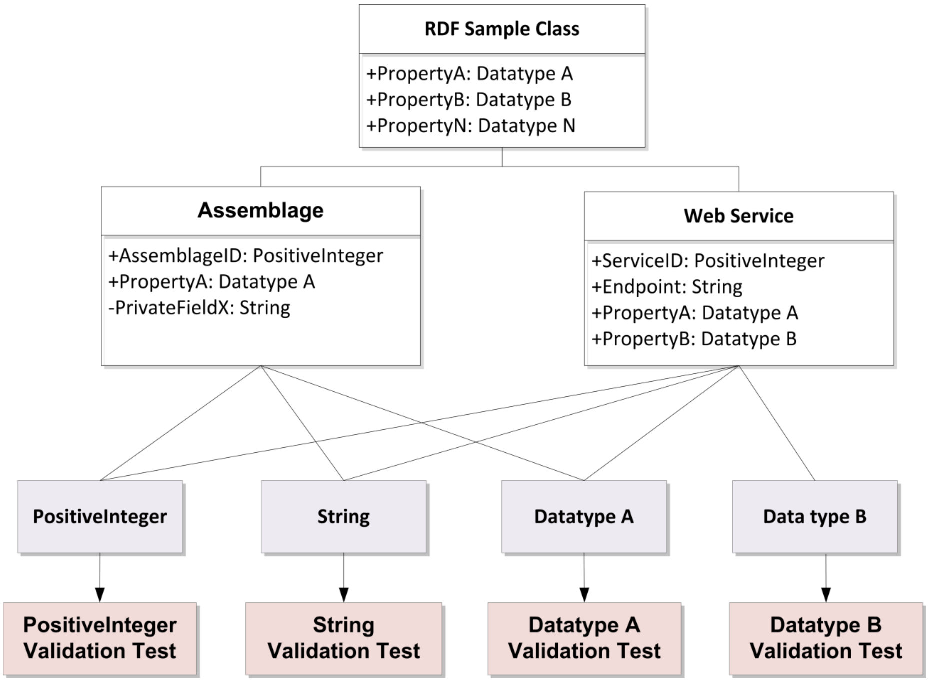


**Fig. 5.5:** Validation Tests

Validation tests (von der Dovenmühle, 2009) are applied to some of generic data types. The investigated data types are Integer, Decimal, String, and DateTime where primitive validation tests are applied. For the validation purposes, the Integer data type is further split into four sub-data types. The validation of these sub-data types is realized as a set of Web Services. As depicted in Figure 5.6 A, these sub-data types are positive, non-negative, negative and non-positive Integers. While the positive Integer includes all the positive Integer values except zero, the non-negative Integer is the generalization of positive Integer data type. It includes all the positive Integer values plus the zero value. Opposite to the positive Integer, the negative Integer includes all the negative Integer values except zero. Eventually, the non-positive Integer is the generalization of negative Integer data type and it includes all the negative Integer values with the zero value included. The generic Integer data type with its sub-data types are associated with validation tests delivered by the Integer validation Web Service.

Similar to Integer, the Decimal data type is also split into four sub-data types. Decimal validation is also realized as a set of Web Services. As depicted in Figure 5.6 B, these types are positive, non-negative, negative and non-positive Decimals. While the positive Decimal sub-data type includes all the positive Decimal values except the zero value, the non-negative Decimal is the generalization of the positive Decimal data type. It includes all the positive Decimal values plus zero. The negative Decimal includes all the negative Decimal values except the zero. Finally, the non-positive Decimal is the generalization of the negative Decimal data type. It includes all the negative Decimal values plus the zero value.

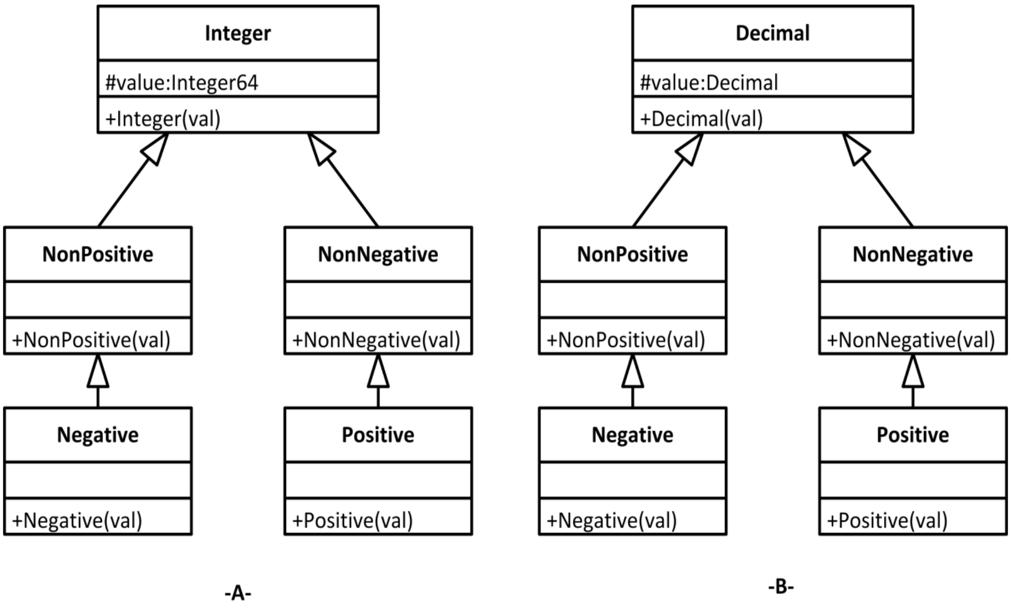


**Fig. 5.6:** Integer and Decimal Validation

Figure 5.7 illustrates how the String data type is organized for validation. A Web Service is used to validate this data type.

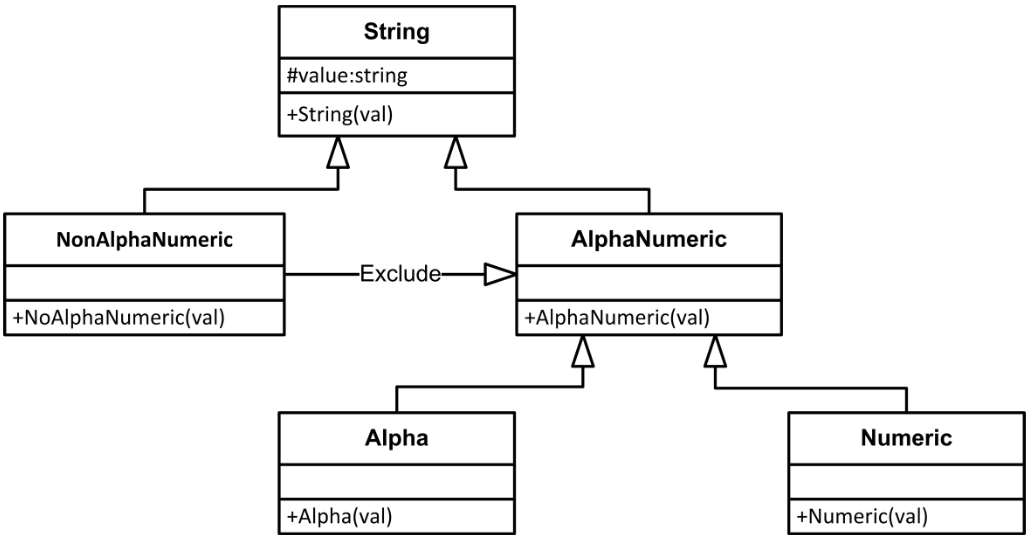


**Fig. 5.7:** String Validation

The String data type is split into four sub-data types namely alpha, numeric, alpha numeric, and non-alpha numeric Strings. The alpha string sub-data type includes all the String values that contain just alphabetical letters. The numeric string values include just number values (plain numbers with “-” to indicate the negative numbers or “.” to indicate the decimal separator) without any other special characters. Alphanumeric sub-data type generalizes both alpha and numeric data types and its values include both letters and numbers without special characters. The last String sub-data type is the non-alpha numeric. Its values include String values with special characters. This distinction between the String sub-data types is quite helpful for validating different fields and inputs like names, numbers, emails…

Finally, the last data type investigated for validation is the DateTime. Checking the validity of this data type is realized by the DateTime validation Web Service. The DateTime data type values are either valid or invalid. It can be classified into two sub-data types: future date or past date as depicted in Figure 5.8. These two types can be tested together with testing the generic DateTime data type. Range of DateTime formats are supported for validation like (MM/YY, MM.YY, DD/MM/YY, DD.MM.YY, DD/MM/YY HH:MM:SS, DD.MM.YY HH:MM:SS or easily HH:MM:SS)[43].

[43] HH stands for hour, MM for minute, SS for second, DD for day, MM for month and YY for year.

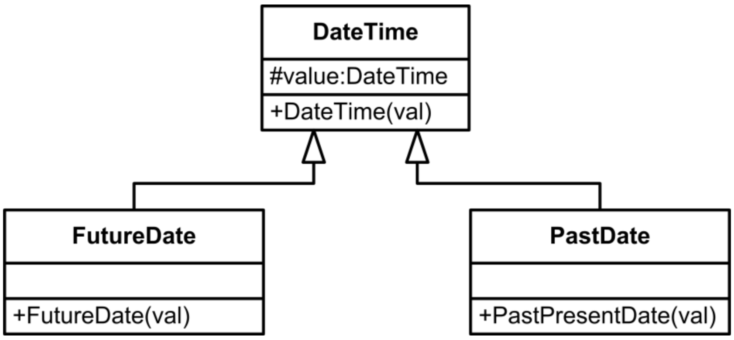


**Fig. 5.8:** DateTime Validation

From the *evaluation perspective*, a security protocol is designed and implemented to evaluate Web Services. After each Web Service call, the evaluation process must be carried out automatically based on a security protocol developed in the master work of (Hasan, 2010). In comparison with the work done by (Brehm & Marx Gómez, 2010), the secure service rating protocol used here is based on this work and updated to have a machine-to-machine evaluation protocol. It meets the confidentiality, integrity and authentication aspects. Using this protocol, the evaluation is done automatically based on two Web Service's criteria namely the response time and availability. The service's evaluation values are controlled by a third party called Evaluation Processing Authority (EPA). As depicted in Figure 5.9, the service consumer is bound with a service supplied by specific service provider. Upon invoking the service, the consumer's machine automatically rates the service based on the abovementioned two criteria, and sends the rating value to the EPA.

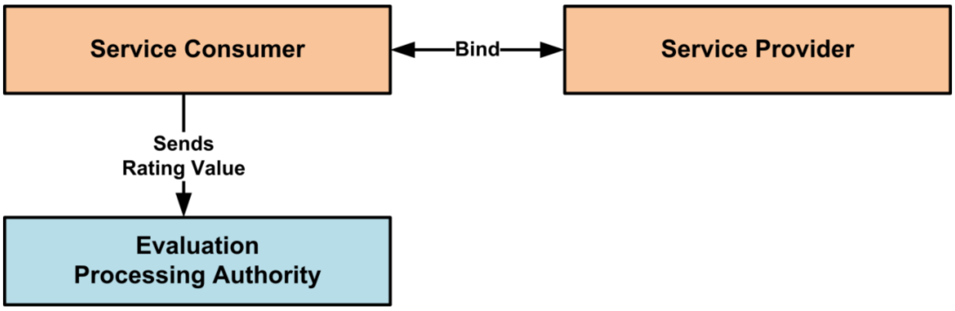


**Fig. 5.9:** Service Evaluation Overview

Using this evaluation protocol, the service provider cannot manipulate the service rating value because it resides at the EPA side. Eventually, what the provider can do is just to check the current value of its services' ratings.

#### 5.3.2.8 Database Requirements

Data access time is crucial factor for the system's overall performance. However, there are many join-operations in the system's data. All RDF statements are represented as in-memory stores copied to the database and retrieved from it when needed.

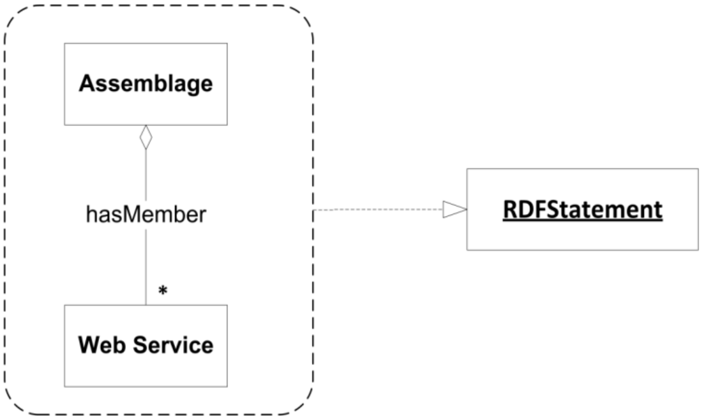


**Fig. 5.10:** Representation of RDF Statements

As illustrated in Figure 5.10, each assemblage is related with its members (Web Services) through “hasMember” relationship. The “Assemblage hasMember Web Service” relation is realized as in-memory RDF statement. This statement object is implemented as a memory store that splits the statement into three entities parts namely assemblage, relation, and Web Service and stores them in the core system database.

Based on the general functional requirements of the entire system from the user perspective, the derived requirements for the database system are schematically defined as follows:

*Retrieve Requested Data:* Running a workflow instance requires the process-specific data to be retrieved from the database. To do so, the workflow management system sends a request to the system database that delivers the process-specific data needed to execute the workflow instance.

*Verify User Input:* When a user logs in, his/her input data is compared with the stored information in the database to verify whether the user is authorized to have access to the system or not. In the case of conflict between the input and stored data, the access to the system is denied.

*Manage User Data:* The user management system is part of the system database responsible of managing users’ with roles and rights. Required user-specific data within the system are stored and retrieved from the system database.

*Store Data:* All users’ inputs must be stored persistently in the system database. The “assign a data access activity” (Section 5.3.2.5) set by the workflow management system achieves this keeping in mind that the existing stored data will be overwritten by the new inputs.

*Supply Requested Data to the Workflow Management System*: This is to provide data as results to the queries coming from the workflow management system. Workflow data are stored in the SESOA database. The request to retrieve data from the database is done by the workflow itself.

*Delete Data:* Existing data in the system database can be deleted by the user who has the authorized permissions. Workflow-specific data can be deleted from the system database as well. However, only administrators who have rights to manage workflows are permitted to delete such data. Without backup, the deleted data cannot be retrieved.

### 5.3.3 Business Case Requirements

Enterprise systems or more specifically ERP systems have normally three business processes clearly connected to create and deliver products and services (Magal & Word, 2011). These processes are the buying (or procurement) process, the producing (or production) process and the selling (completion or fulfillment) process. This work is more concerned about modeling and implementation of one of these processes that is the selling business process depicted in Figure 5.11. This business process is considered as the accompanying business case to SESOA. It is mainly implemented to utilize the system's functionalities. This section here is dedicated to define the main requirements of the selling business process.

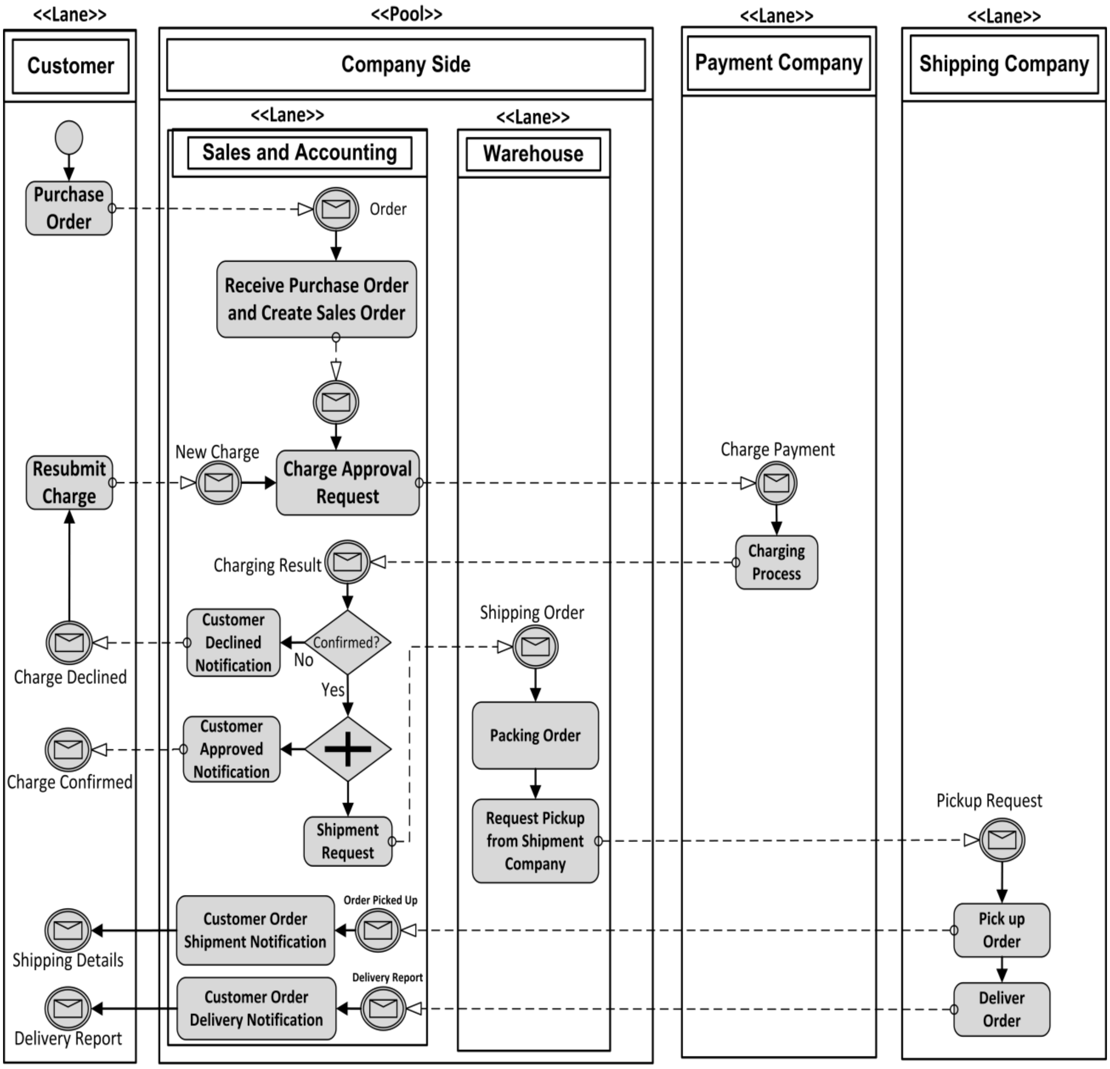
The main goal of this process is to design and implement a generic completion business process on top of the system's prototype. Most of the activities in this process have been realized using service orientation concept and Web Servicers technology. This enables individual services to be exchanged quickly and easily without huge costs of change. One more virtue behind realizing this process using Web Services is the enablement of using services supplied by external service providers. Therefore, process outsourcing can be effectively performed.

Figure 5.11 shows the business process model of the investigated business case where Business Process Modeling Notation (BPMN) (White & Miers, 2008; White, 2004) had been used as a business process modeling language. The resulted prototype implements this selling process as an online shop that is made available to customers for ordering its products. The process starts once a customer places an order. The sales and accounting department receives this order and creates a sales order waiting for the customer's payment. The payment is checked by an external payment company. Upon charge approval, a shipping employee from the warehouse department packs the order and requests a shipment pickup number from another external shipping company. Eventually, the shipping company picks up the order and finally delivers it to the customer.

Some of the functions implemented in this business case are:

- Show product information
- Show customer-product information record
- Show pricing conditions like discounts, offers etc.
- Create quotation for a specific product
- Create customer purchase order

- Create sales order based on the customer purchase order
- Process payment for a sales order
- Process shipment for a sales order
- Process billing for a sales order
- Display document flow that shows all of the documents associated with the steps that have been accomplished for an order
- Display work lists that associate the tasks that are ready for completion
- Display online lists that can show some documents like list of sales orders for a specific combination of customers and products or list of delivery documents for a specific customer.



**Source:** Based on Figure 4-6 in **(Rosen et al., 2008, p. 139)**

**Fig. 5.11:** Business Process Model for Selling Process

All of these functions have been implemented taking into account that products have to be always available to the customer who has to register and login to the system. Moreover, the consumer can add, modify or delete items in a shopping cart. The user can set his/her invoice and shipping address for each order and can also choose between different payment methods (credit card, bank debit, advance payment) and different shipping companies to ship the bought goods. Furthermore, the user's order processing notification together with historical browsing of the purchased products are made also available to the customer.

The whole selling process is realized using workflows and Web Services. The activities of these workflows are following the core system's requirements (presented in Section5.3.2). For example, both workflow's payment and shipment activities are executed by calling Web Services in payment and shipping WS-assemblages respectively. Many similar services can be found in those assemblages and it is the decision of the workflow developer to choose which service to call based on specific rules and criteria.

### 5.3.4 Non-Functional Requirements

The system together with its accompanying business case must take into account a set of non-functional requirements to be defined and met accordingly. The individual non-functional requirements are grouped and sorted as either mandatory or optional requirements. Below is the list of the non-functional requirements that have been taken into account in this work:

*Modularity:* The overall system is composed of different components that collaborate together to provide the system's functionality. Most of the system's functions are realized using standardized RDF-enriched Web Services. Therefore, modular software design is a mandatory requirement. This leads to an appropriate customized software solution in which components can interact, connect and share resources by using modular or standardized interfaces.

*Usability:* It is a mandatory requirement. An intuitive and coherent design of system's GUI is crucial. Each user interface must be designed in a clear and unambiguous way. These interfaces should be based on common usability concepts, so that all system parts can be easily used without a need for long time training.

*Performance and Efficiency:* While creating the first versions of the prototype, system functionalities are actually appearing in the foreground. Therefore, system's performance and efficiency accorded as optional requirements. The same applies to the system's effectiveness and efficiency in terms of resource consumption.

*Scalability:* Based on the aforementioned modular software design requirement and the fact that the system's components can be easily replicated or even replaced, the entire system can cope with the changes that might accompany any enterprise requirements, goals and visions. Hence, the scalability is a mandatory requirement. Likewise, downsizing is enabled by removing any unnecessary component(s).

*Reliability:* The resulted prototype has to be as error-free as possible. If necessary, occurring errors must be handled in a way that the system can still working or permit a (partial-) shutdown. Error messages should be meaningful and documented. System data must not get lost or left inconsistent because of these errors. User input errors caused by system's users must be intercepted and checked immediately so that any invalid input is handled with a meaningful notification message back to the user who caused it.

*Correctness:* Data correctness is a mandatory requirement of the system. In the context of the planned application, correctness means that the delivered data values and results (following the system's rules) are adequate and conform in quality with defined business requirements and criteria.

*Security:* It is a mandatory requirement. System's users with their rights provide a reasonable degree of security. Web Services rating and securing the communication channel between service providers and consumers provide another level of security. The prototypical implementation tries to protect existing data against unauthorized access. Furthermore, it tries to protect data sources against unauthorized access and modification. When the system runs in a production environment, this requirement must take a higher priority.

*Internationalization and localization:* This is an optional requirement. The already designed user interfaces and the prototype as a whole are initially implemented using the English language (font encoding, date formats, etc.). A later productive system has to meet with subsequent internationalization and localization (i18n) requirements. Based on these requirements, multilingual interfaces with possibility to switch between different languages have to be provided.

*Maintainability:* Several technologies have been used to implement the system. The created software components are well defined and their interfaces are fully documented. At run-time, new components can be added and/or other active components can be deactivated and potentially removed using a pre-defined process. Therefore, maintaining the resulted prototype with its accompanying business case is important. However, the maintainability requirement in this work is partially optional because maintenance is done just on some administrative GUIs and at the database level (by the system administrators).

Table 5.4, summarizes the aforementioned non-functional requirements and sort them as mandatory, partially optional or optional requirements.

**Tab. 5.4:** List of Non-Functional Requirements

| Number | Non-Functional Requirement | Sort |
|---|---|---|
| **1** | Modularity | Mandatory |
| **2** | Usability | Mandatory |
| **3** | Performance and Efficiency | Optional |

| | | |
|---|---|---|
| **4** | Scalability | Mandatory |
| **5** | Reliability | Partially Optional |
| **6** | Correctness | Mandatory |
| **7** | Security | Mandatory |
| **8** | Internationalization and localization | Optional |
| **9** | Maintainability | Partially Optional |

## 5.4 Summary

In this chapter, the conception phase behind this work has been presented. The concept of lightweight Semantic-enabled Enterprise Service-Oriented Architecture (SESOA) has been defined. An introduction to the semantic support was then presented. The idea of grouping Web Services in assemblages and the definition of the later has been then explicated.

Core system requirements divided into functional and non-functional requirements together with accompanying business case requirements have been defined. The resulted system based on these requirements is a process-oriented system in which business processes are realized as workflows. Some of the activities within these workflows are implemented using Web Services. Specifications of the SESOA reference architecture for Web Services are presented in details in the next chapter.

# 6 Reference Architecture of Semantic-enabled Enterprise SOA

This chapter is the central part of the thesis and proposes semantic Web-based architecture for Web Services (Mahmoud et al., 2011; Mahmoud, Petersen, et al., 2012; Mahmoud & Marx Gómez, 2010; Mahmoud, 2009). It explicates in details the reference architecture of the lightweight Semantic-enabled Enterprise Service-Oriented Architecture (SESOA) together with the internal architecture of its core building components. Moreover, the main interactions between these components, scenario of registering services within WS-assemblages, service validation and evaluation are detailed in this chapter.

The detailed layered architecture, the architecture overview and the component-based architecture of SESOA are explained in section 6.1 with highlights to the internal system's components. Section 6.2 explicates the scenario in which Web Services are registered within assemblages. Following that, service validation and evaluation are detailed in sections 6.3 and 6.4 respectively. Section 6.5 illustrates the semantic annotation of Web Services relations using RDF statements together with the main interactions among the system's components. Section 6.6 shows how the business case architecture is modeled in context of the system architecture. Then, Section 6.7 lists the main outcomes that can be obtained from this work. Finally, this chapter briefly summarizes the main issues explained in its sections.

## 6.1 Semantic-enabled Enterprise SOA

### 6.1.1 The Layered Architecture

This section illustrates the system's reference model as a logical software architecture that does not refer to any specific technology. A reference model as defined by (MacKenzie et al., 2006, p. 29) is: "an abstract framework for understanding significant relationships among the entities of some environment that enables the development of specific architectures using consistent standards or specifications supporting that environment. It is independent of specific standards, technologies, implementations, or other concrete details". SESOA architecture is designed as a tier architecture that applies the separation of concerns architectural principle by separating presentation from logic, and logic from data. This layered architecture arranges the system's logical tiers (layers or views) and the relationships among them. As any other tier-based architecture, the SESOA layered architecture has the presentation, business and resource tiers. From an architectural point of view, the only difference in this architecture is the division of its business tier into two sub-tiers namely: service and process layers.

Figure 6.1 illustrates the layered architecture of SESOA with its three main tiers. The presentation layer is mainly the system GUI by which user can login to the system using different devices (computers, smart phones…). User authentication is required at this point. Based on his/her rights, the system provides the user with a list of workflows. Execution of workflows is initiated by the user who can choose the desired workflow to execute. These workflows are managed by the workflow management system (WfMS).

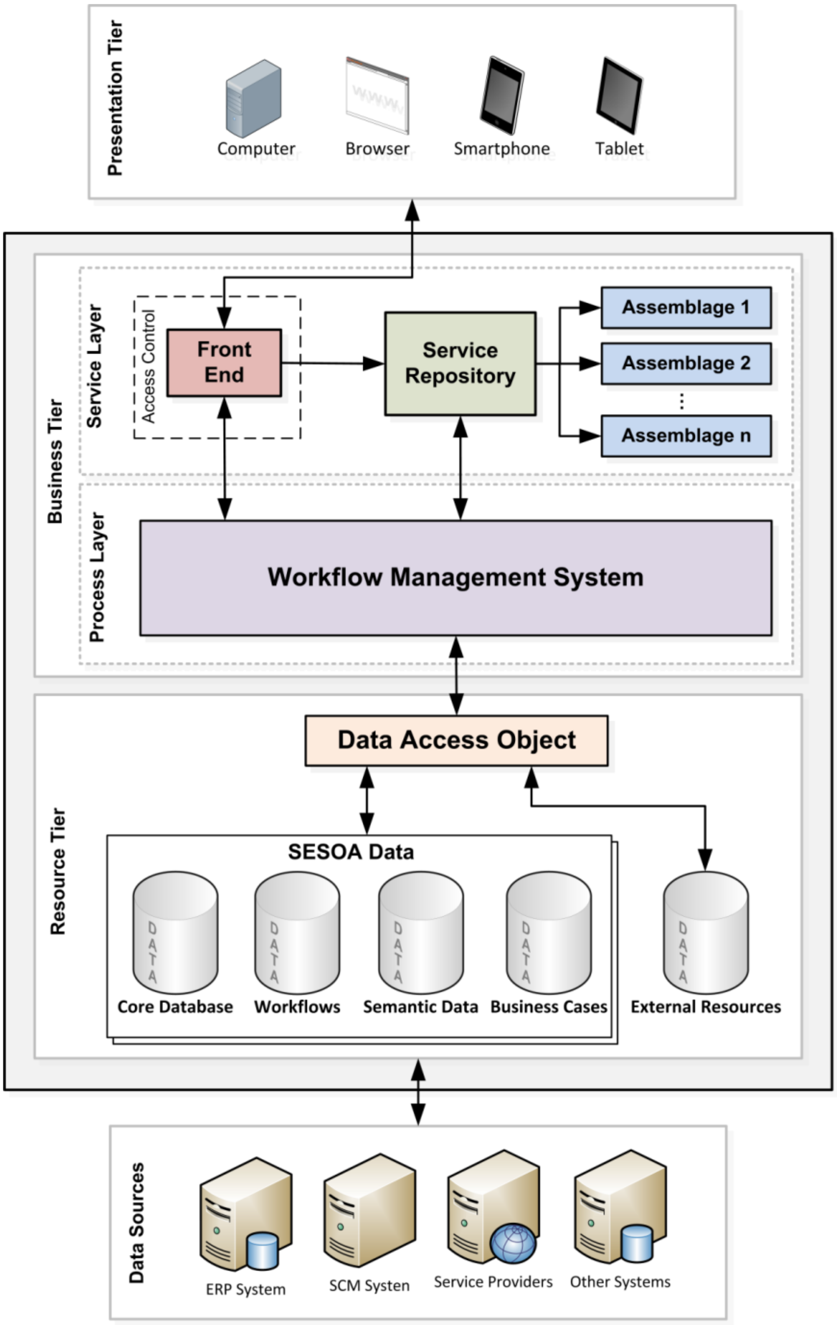


**Fig. 6.1:** SESOA-Reference Model as Layered Architecture

Most of workflows' activities are implemented as Web Services. Upon workflow execution, the WfMS discovers the service repository to fetch a list of Web Services required to execute the workflow's activities. These services are supplied by service providers as data sources (refer to Figure 6.1).

The results of executing workflows are presented back to the user via the system's front end. All the necessary connections to the databases are realized in the Data Access Object's interface (DAO component) that links the WfMS to the SESOA's different set of databases. SESOA data are stored and managed by four main databases. These databases are:

- The core database in which all user management, system-relevant data like Web Services data (IDs, endpoints, and description…) are stored.
- The workflows database that manages the workflows data to enable persisting and tracking of the workflow's instances.
- The semantic database that includes all the RDF statements required to represent the relations between Web Services and their WS-assemblages. It stores them in form of subject-predicate-object statements where subjects, predicates, and objects RDF entities are stored using unique IDs.
- Business cases database that includes the relevant and needed data to create and run the business cases that use the system's functionalities.

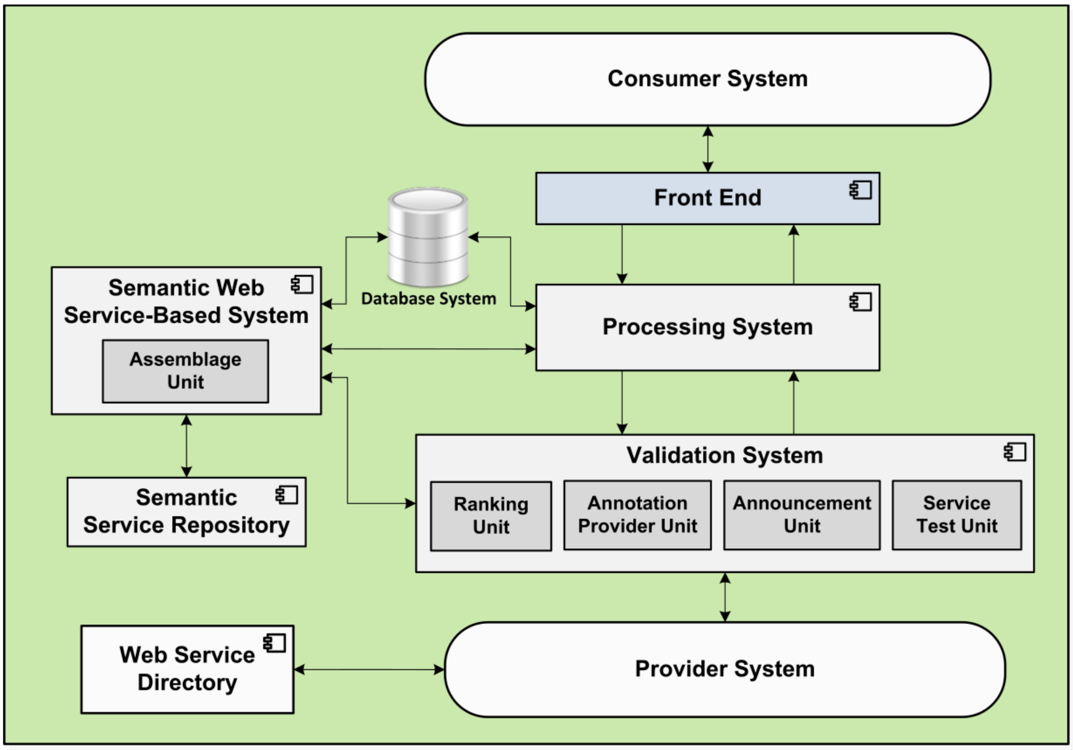
Moreover, different external systems (from the data source layer) like ERP or SCM systems might be integrated with the system. Just authorized external systems with proper rights can use SESOA data. To achieve this integration, a specific business scenario includes the involved external system is realized in form of workflow. The resulted workflow is then managed by the WfMS to process interactions (inputs and outputs) with the intended external system.

### 6.1.2 Architecture Overview

This section describes the main aspects of the proposed reference architecture with its components (Mahmoud et al., 2011). The main idea behind this section explains the semantic-enabled architecture for Web Services that has the role of dealing with Web Services and the semantic annotations of their relations. As mentioned in Chapter five, the Web Services are grouped in WS-assemblages. Service classification is based on enterprise's business domains or areas of interest. In designing the proposed architecture, the main aspects in designing service interface and implementation have been defined respectively. Firstly, the general concepts, goals and concerns related to this architecture have been specified. Other related concepts of what processes, channels, external systems, services and WS-assemblages are defined as well. Secondly, the system's business requirements are identified and illustrated using some use case, sequence, and detailed scenario (activity and partitioned activity) diagrams.

Based on the reference model described as layer architecture in the previous section, this section shows how this reference model can be transformed into reference architecture. The term architecture generally describes the wide term Software Architecture that represents the structure of a software system. This structure comprises the software elements (components), their properties and their relationships (Clements et al., 2010). As defined in (MacKenzie et al., 2006, p. 29), a reference architecture is "an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements". The aim of describing reference architecture is to provide an abstract architectural template in a specific problem domain. By considering general requirements that are identified in a particular domain, reference architecture provides the possibility of conceptual adoption or adaptation of system structures.

To conclude, the reference architecture described in this thesis is an abstract software architecture that can be applied to multiple business domains. The limitation applied to this reference architecture is that by adopting it, the operational flow logic is explicitly done in form of workflows. The activities within these workflows are realized as Web Service calls that execute available operational functions provided by many organizational structures of a company and/or set of companies. Figure 6.2 depicts the SESOA reference architecture.



Source: Figure 2 in **(Mahmoud, Petersen, et al., 2012, p. 186)**

**Fig. 6.2:** SESOA-Reference Architecture

The overall system consists of five main subsystems besides the three typical SOA subsystems (consumer, provider, and Web Service directory). These subsystems are connected to each.

The overall architecture is built on top of SOA concept in consolidation with business processes and semantic enrichment of Web Services. The architecture's subsystem can be described briefly as follows:

*The Consumer System* is actually the system's end user who discovers, selects and invokes the available Web Services. Moreover, the consumers are able to discover some WS-assemblages (based on their rights and privileges) to locate potential Web Services. Consumers can also initiate workflows that reflect the actual business processes managed by the processing system.

*The Front End* represents the system's GUI through which users can access the system and use its functionalities. Logged in users are provided with list of available Web Services, WS-assemblages, and workflows. The workflow execution is initiated by the user and managed by the processing system that is connected to the front end. Set of validation tests at the data type level are provided to the users as Web Services to check the validity of the user's inputs. Semantic RDF statements validation is also available using the W3C RDF validation service.

*The Provider System* is merely a Web Service provider. It creates Web Services and publishes them in one or more Web Service registries (UDDI or any other registry) in which service consumers can locate services to potentially consume them. Whereas service provider implements some the services functional (like messages and binding) and non-functional (like price) properties, third parties assign values to the rest set of properties (like reputation, availability, confidentiality…).

*The Web Service Directory* represents the big container of services supplied by different service providers. It is the medium between service consumers and providers. On the one hand, consumers discovers directories to find potential services and get direct binding with the providers of these services. On the other hand, service providers publish their services in service directories to market and make them available to wide ranges of service consumers.

In other words and following SOA concept, the consumer system (via the system's front end) discovers the Web Service directory to find some services meeting his/her needs (passed as keywords or highly defined requests). Upon finding a service capability fulfills its request the consumer system binds directly to the provider system that supplied this service in order to invoke it.

*The Processing System* is the centric subsystem that encapsulates and realizes the business logic in this reference architecture. It represents the SESOA's workflow management system that manages the system's workflows. Several business processes can be translated into workflows and steered by this subsystem. The translated business processes, i.e. the workflows, include several activities that are going to be executed by calling the other architecture's subsystems connected to the processing system. Via data connections, the processing system is connected to the following architecture's subsystems:

- Consumer system via the front end
- Database system
- Semantic Web Service-based system
- Provider system via the validation system

All the system's workflows are provided to the user based on their rights and privileges. Most of the workflows activities are realized as Web Services. The consumer system can discover, find and invoke services via the processing system. This system controls the data flow among the reference architecture's subsystems.

*The Semantic Web Service-based System* is the subsystem that handles the grouping of Web Services in assemblages and the semantic annotation of their relations. The assemblage unit manages all the WS-assemblages by creating, modifying and deleting assemblages based on changes at the business needs. Moreover, the relations between WS-assemblages and services are semantically annotated in this subsystem using RDF statements. The statements entities are stored in the database system. These RDF statements are also made available at the semantic service repository. A data connection between this subsystem and the processing system is necessary to provide the latter with list of services (from the semantic service repository) needed for executing the system's workflows.

*The Semantic Service Repository* includes the basic information about WS-assemblages and their actual Web Services exposed as RDF statements. When service providers register their services in the system in one or more WS-assemblages, an automatic RDF statement is created indicating this relationship. The entities of this statement are stored in the database system. The semantic service repository is then updated to include this statement. Moreover, several RDF semantic query languages like SPARQL can query this semantic repository to locate service information.

*The Validation System* provides set of validation tests for Web Services. It has the ranking, the annotation provider, the announcement, and the service test units. The ranking unit provides secure Web Service rating based on automatic machine-to-machine evaluation protocol. The annotation provider manages the semantic annotation of relations between Web Services and WS-assemblages in which they are registered. The announcement unit has an interface with the provider system to advertise for new functionalities the system's services cannot provide. Finally, the service test unit provides some data type validation set of tests.

*The Database System* gathers all the system's databases. It includes user management, semantic, workflow, and business cases databases. Different set of users can access and maintain these databases. While executing workflows, many activities read and/or write data in one of the database system's databases. Therefore, most of the database interactions are performed by the processing system

### 6.1.3 The Component-based Architecture

This section provides an overview of the system's component-based architecture. Generally, component architectures aim at making clear separation of technical and business requirements of an information system. Component architecture has the advantage that different implementations of the same type of components can be supported simultaneously. Business requirements are realized through the implementation of accompanying business cases that utilize this component architecture.

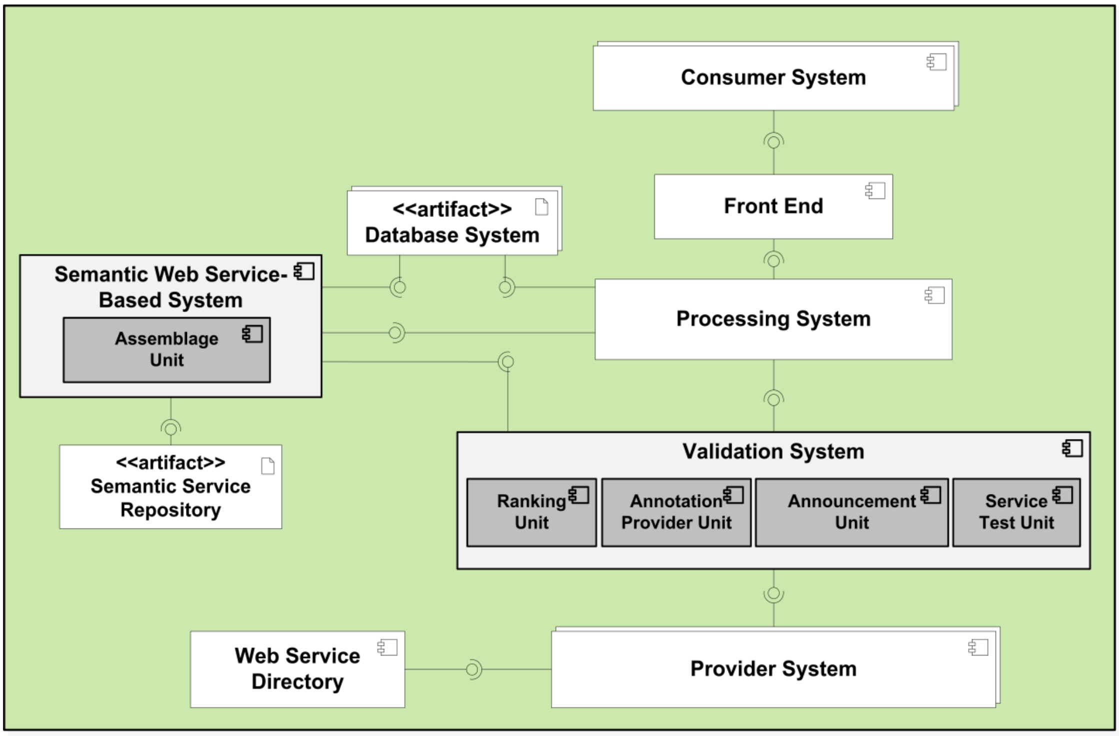


**Fig. 6.3:** SESOA-Reference Architecture as Component-Based Architecture

Figure 6.3 depicts the SESOA reference architecture as a component-based architecture. As a modeling language, Unified Modeling Language (UML) has been chosen (OMG, 2009). Component diagrams are used also to model the internal subsystems of SESOA reference architecture. Moreover, in all parts of this work UML version 2.2 (see http://www.omg.org/spec/UML/2.2/) has been used. In the following sections, more details are given to the interfaces among the system's components

As mentioned in the layered architecture, the main entity in the process layer is the WfMS that manages the workflows. In this component-based architecture, the processing system component has this responsibility. It provides four main interfaces:

- Interface with the system's front end: Consumer system initiates the process of workflow execution. While executing workflows, several requests are tunneled via the system's front end to the processing system. After processing the execution of the workflow with its activities, the processing system generates the corresponding response back to the consumer system via the front end. In case of failures, the processing system generates a response back to the consumer system indicating proper failure message.

- Interface with the semantic Web-Service-based system: Executing workflows by the processing system requires fetching list of workflow-relevant Web Services. This is achieved by the interface with the semantic Web Service-based system where Web Services are grouped in WS-assemblages. The list of services is then retrieved by the processing system to complete the workflow execution. In case of failures, proper handling is done by the processing system.

- Interface with the provider system via the validation system: The connection to the external (and internal) service providers is established via this interface. The services supplied by different provider systems are tested by the validation system before being called by the processing system. Moreover, absent functionalities that do not exist yet in the system are forwarded to service providers by the validation system's announcement unit.

- Interface with the database system: Executing some of the workflow's activities requires transactions with the system's databases. These interactions are handled using this interface.

Requests for Web Service are handled by the semantic Web Service-based System. It uses semantic-annotated Web Services that are published in the semantic service directory to fulfill users' requests or to orchestrate new services using the existing ones. It notifies and updates the processing system about the services' availability in order to reduce the overall response time. This subsystem implements also four main interfaces:

- Interface with the semantic service repository: The assemblage unit within this subsystem creates the WS-assemblages and registers the services as members within these assemblages. Moreover, the process of annotating the relations between assemblages and their services is done by this system. The whole semantic RDF description is then published at the semantic service repository using this interface.

- Interface with the processing system: The main role of this interface is to form a communication bridge with the processing system to receive service requests. Proper response is then sent back to the processing system indicating either list of services or a failure response.

- Interface with the provider system via the validation system: As mentioned before, the services supplied via different service providers are registered within WS-assemblages managed by the assemblage unit in the semantic Web Service-based system. The service providers may benefit from the validation system to validate their services before registering them in desired WS-assemblages. The validation system offers data type tests and secures service rating.

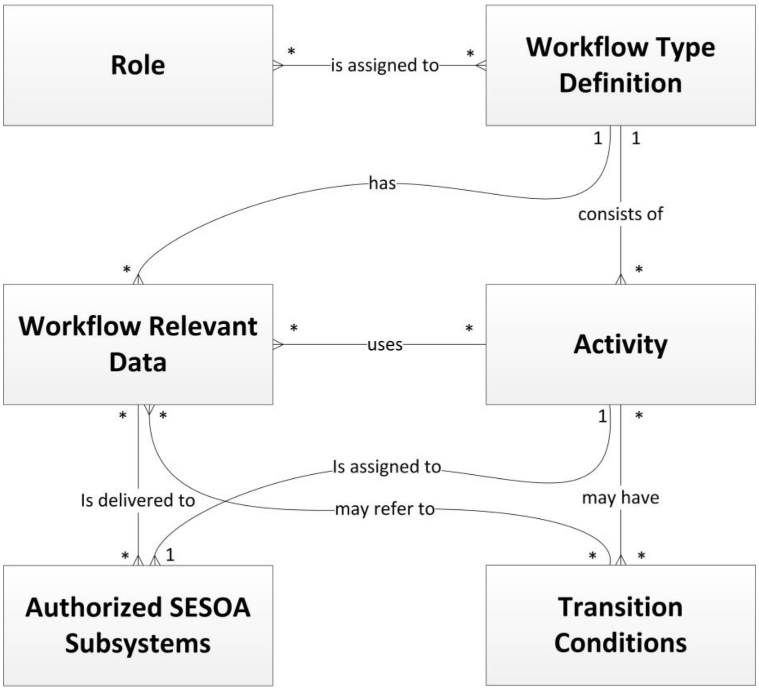
- Interface with the database System: All data managed by this subsystem are stored in the database system. WS-assemblages and services data are stored in the core database. RDF statements of the relations between these assemblages and their services are stored in the semantic database. In these RDF statements, each of these assemblages is accorded as a subject entity, the relationships with the services are accorded as predicate entities, and the actual Web Services are accorded as object entities. Subject, predicate, and object entities are then stored in the semantic part of the database system. This storage is done automatically upon registering Web Services in assemblages.

The next following sections illustrate the internal component-based architectures of each of the abovementioned subsystems.

#### 6.1.3.1 Processing system

Main design issue in the processing system component is the process definition and how can business processes be transferred to workflows. Based on (Hollingsworth, 1995, p. 30), Figure 6.4 shows the SESOA basic process definition meta-model in which each workflow is defined as set of activities and assigned to different roles (cf. Brehm, 2009, p. 111). Workflow type definition includes the main workflow properties like workflow process name, process start and termination conditions, version number, security and other control data. Different set of users can execute different types of workflows.

Roles in this meta-model refers to the name or the organization entity that can execute the workflow. Moreover, each workflow has its own workflow relevant data (like data name and path, data types…) that its activities use. Each activity has a name, type, pre- and post-conditions besides some other scheduling constraints. As mentioned before, many workflow activities are realized as Web Services and therefore each activity is assigned to some SESOA subsystems and may have some transition (flow or execution) conditions. SESOA consumer, database, and semantic Web Service-enabled systems are the subsystems that deal with workflow activities. Consumer system initiates the process of workflow execution and gets back responses from the processing system via the system’s front end. While executing activities, several interactions with the database system are required. More specific, the workflow database provides the workflow-relevant data, the semantic database provides RDF statements of Web Services-assemblages relations and the core database provides the required information to locate those Web Services.



Based on Figure 10 in **(Hollingsworth, 1995, p. 30)**

**Fig. 6.4:** SESOA Process Definition Meta-model

As mentioned before, different business processes are realized as workflows to be managed by the processing system component. The following part explicates the internal architecture of this component in relation to the other SESOA components. The main tasks of the processing system component include modeling and managing process models in the form of process definition and execution of workflow instances. Figure 6.5 shows the component structure of the processing system. The following sections clarify this structure by describing its individual components.

The processing system component represents the SESOA’s WfMS. As depicted in Figure 6.5, it is composed of the workflow engine, workflow manager, workflow editor and the associated Data Access Object (DAO) subcomponents. Moreover, it is linked with the SESOA’s front end, semantic Web Service-enabled system, semantic service repository, validation, and database system components.

*The Workflow Engine* subcomponent represents the heart of the processing system. It provides a runtime environment for executing the system’s workflow instances. The main characteristics of this subcomponent are derived from the workflow reference model (Hollingsworth, 1995, p. 20) developed by the Workflow Management Coalition (WfMC) (see http://www.wfmc.org/).

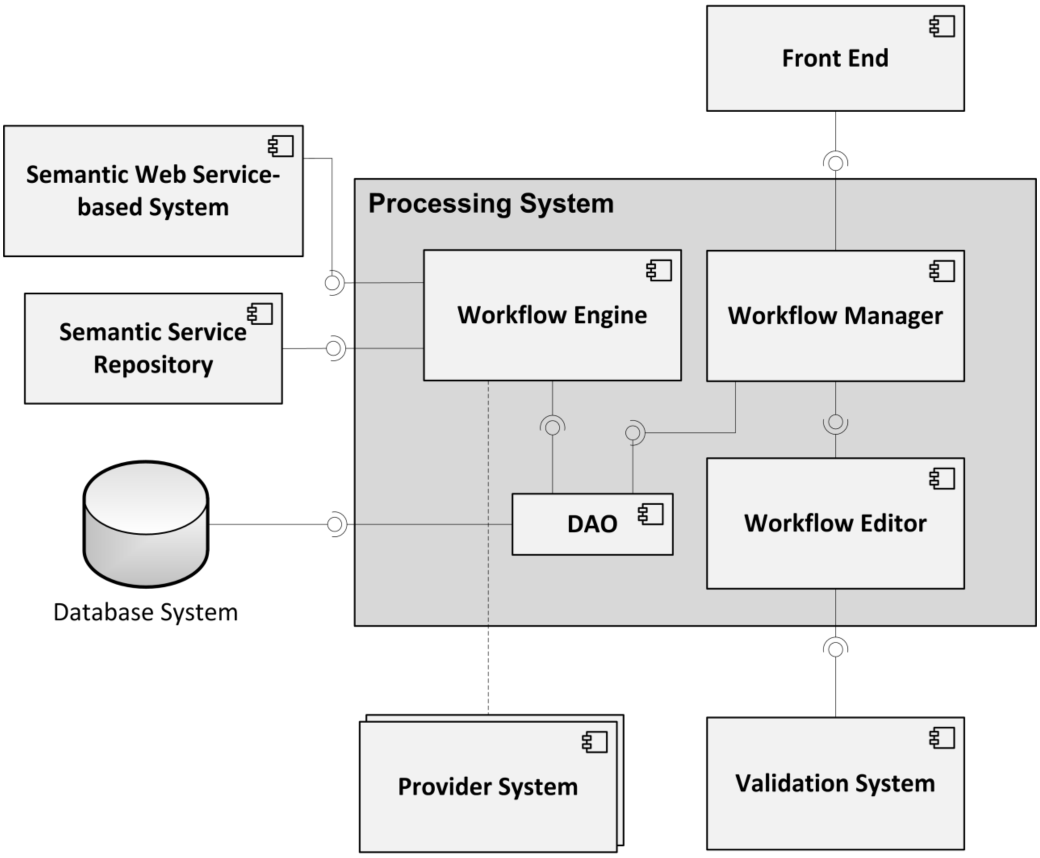


**Fig. 6.5:** Processing System Component Structure

In accordance with (Hollingsworth, 1995, p. 22), Table 6.1 summarizes the list of main functionalities covered by the processing system's workflow engine. These functionalities include the process definition interpretation, workflow instances controlling, navigation between workflow activities, work items identification, workflow data maintenance, external applications support, participant management, and last but not least the supervisory actions.

**Tab. 6.1:** Workflow Engine's Main Functionalities

| Name | Description |
|---|---|
| **Process Definition Interpretation** | Interpretation of process models is enabled following the specification of the used process description language and the management of workflow control data. |
| **Workflow Instances Controlling** | Includes the control of workflow instances creation, activation, suspension, termination, etc. |
| **Workflow Activity Navigation** | Indicates the navigation between different workflow activities, which may involve sequential or parallel operations, deadline, scheduling, interpretation of workflow relevant data, etc. |

| | |
|---|---|
| **Work Items Identification** | This functionality deals with the identification of work items for user attention and provides an interface to support user interactions. |
| **Workflow Data Maintenance** | Is responsible of workflow control data and workflow relevant data maintenance. This functionality enables the process of passing workflow relevant data to/from applications or users. |
| **Support External Applications** | The support of external applications is enabled by providing an interface to invoke external applications and link any workflow relevant data. |
| **Participant Management** | Provides the sign-on and sign-off management of specific participants. |
| **Supervisory Actions** | This functionality comprises supervisory actions for control, administration and audit purposes. |

As a conclusion, the processing system's workflow engine is mainly responsible of executing set of workflow activities, sub-activities, or instances. Since some of the activities are realized as Web Services, the workflow engine connects via two interfaces with both the semantic service repository and the semantic Web Service-based system to discover the required services within WS-assemblages and locate these services to call them as part of workflow execution. Therefore, there is an indirect connection between this subcomponent and the provider system that supplies these services. Finally, the workflow engine interacts via the DAO interface with the database system that provides the required basic database functions required for workflow execution.

*The Workflow Editor* subcomponent is used as a process modeling tool. Following the SESOA process definition meta-model (see Figure 6.4), the workflow editor creates workflows out of business processes. It translates company's business scenarios and processes into workflows. Any workflow editor can be used to realize this subcomponent's functionalities (e.g. graphical or textual editors). In other words, the main task of the workflow editor is to support the workflow developer who is responsible for translating business processes into the internal representation format used for describing workflows. In this work, workflows are represented graphically and can be edited to include hardcoded custom activities on demand. These workflows are then managed by the workflow manager.

*The Workflow Manager* administers the already created workflows by the workflow editor. It organizes the workflows and assigns IDs to them in order to be stored in the workflow part of the database system via the DAO interface. Moreover, the workflow engine can load the workflows based on their IDs to be executed[44]. Workflow manager provides set of functions covering workflow administration besides main functions proposed by (Hollingsworth, 1995, p. 45) including user, role, and audit management operations, resource control operations, process supervisory and process status functions. Table 6.2 lists the workflow manager's main functionalities:

**Tab. 6.2:** Workflow Manager's List of Functionalities

| Name | Description |
|---|---|
| **Workflow Administration** | This functionality includes adding, modifying or deleting workflows from the workflow part in the database system. |
| **User Management Operations** | Includes establishment, deletion, suspension or amendment of privileges of system's users or workgroups. |
| **Role Management Operations** | Reflects the management of the participant relationships including defining, deletion or amendment of roles besides setting or unsetting role attributes. |
| **Audit Management Operations** | This is optional functionality to query, print, start, or delete audit trail or event log. |
| **Resource Control Operations** | Process or activity concurrency levels in this functionality can be set, unset or modified. Moreover, it interrogates resource control data (like counts, thresholds, usage parameters, etc.). |
| **Process Supervisory Functions** | Workflow definition and/or its instances may change their operational status using this functionality. It manages workflow definition different versions and enables the management (change, assign attributes or terminate) of all process or activity instances states that are from a specified type. |
| **Process Status Functions** | In enables management of workflow or activity instances status queries in addition to retrieve details of such instances using special filtering criteria. |

What can be concluded from the abovementioned details is that the workflow manager enables workflow-related administrative tasks such as adding, updating, and deleting workflows and their instances. The DAO encapsulates all required operations to access the workflow part of the SESOA database system.

By realizing the processing system component, standard workflow extensions used for persisting and tracking the system's workflow activities are enabled. Custom extensions as code activities can be also implemented to add operations to the already existing ones

[44] Workflow engine is designed primarily to process just the registered workflows within the system. The semantic service repository and the provider system support the workflow engine to execute workflow's activities that are realized as Web Services.

in the workflow editor. This brings the benefit of using such extensions in enhancing the resulting prototype. Moreover, the usage of database transactions across activities is enabled to ensure that the system's data are updated consistently.

Persistence of workflow instances or more precisely SQL persistence to a DBMS is enabled as well. The states of all workflow instances coming from the workflow engine can be stored in the workflow part of the SESOA database system.

Another type of workflow extensions are the tracking ones that are utilized to track events as workflow executes. This is helpful to monitor the execution of workflows, to trigger external processing, and to leave audit trails for future diagnostics. Moreover, a mechanism to enlist the system's workflow activities on the same database transaction to ensure that updates are performed in a consistent manner is supported as well.

Finally, implementing workflows themselves as Web Services is also enabled in this system. Further technical details are provided in Chapter seven that gives insights to the SESOA prototypical implementation.

#### 6.1.3.2 Consumer system

The consumer system component represents the system's end user who can initiate and execute workflows, and benefit from the business cases implemented on top of SESOA reference architecture. It implements all the necessary functions that are called by the end user interfaces and contains the schema definitions and functions that are needed for discovering and calling Web Services provided by different service providers. This subsystem is able to generate user screens at runtime. Via the system's front end, this component is linked with the processing system component that deals with the business processes that are described in an appropriate XML-based workflow language. This front end provides the consumer system with an interface that can graphically call all the system's functionalities to the end user. The view that the consumer system has differs from one user to another based on the rights and privileges the user has. The delivered SESOA system differs between three types of users namely the end user, the service provider and the system administrator. Table 6.3 details the main characteristics of these three types of users:

**Tab. 6.3:** System User Types

| User Type | Description |
|---|---|
| **End User** | ✓ Can access the public services published in the semantic service repository<br>✓ Can benefit from the business cases built on top of SESOA<br>✓ Can initiate and execute public workflows or other workflows if having proper rights. |

| | |
|---|---|
| **Service Provider** | ✓ Can publish services in the semantic service repository<br>✓ Can register services in WS-assemblages<br>✓ Can modify service information in the semantic service repository |
| **System Administrator** | ✓ Has full control on the system and its business cases<br>✓ Manages the system users<br>✓ Can modify or delete service and/or WS-assemblage information<br>✓ Administers the SESOA database system<br>✓ Can modify or delete the semantic annotations of service relations |

Figure 6.6 depicts the internal architecture of the consumer system component. It has one subcomponent called GUI manager that is connected to the processing system component via the system's front end. Moreover, the consumer system includes all the required business and administrative functions related to the user's communications.

*The GUI Manager* is responsible of generating graphical user interfaces for the aforementioned types of users. There are different types of interfaces that are shown to the user. These types include dynamic user interfaces without user interactions or user interfaces with user interactions and exceptions. The interface generation depends on the workflow activity that the user deals with. Proper exception handlings with user-friendly responses are generated back to the user in case of improper user inputs or system failures.

The generic user interface is available to all user types with different views depending on the user type. System administrators have all the graphical elements to manage and control the overall SESOA prototype. Service providers have restricted views where they can query all services information, the semantic description of service relations, and access set of validation tests. System administrators can add, modify, or delete service and assemblage information from the semantic service repository and database system besides having full control on the overall system.

As depicted in Figure 6.6, the consumer system is linked to the processing system via the system's front end. User information is retrieved via the DAO interface from the database system. The users can initiate then the workflows using the GUI available to them (managed by the GUI manager).

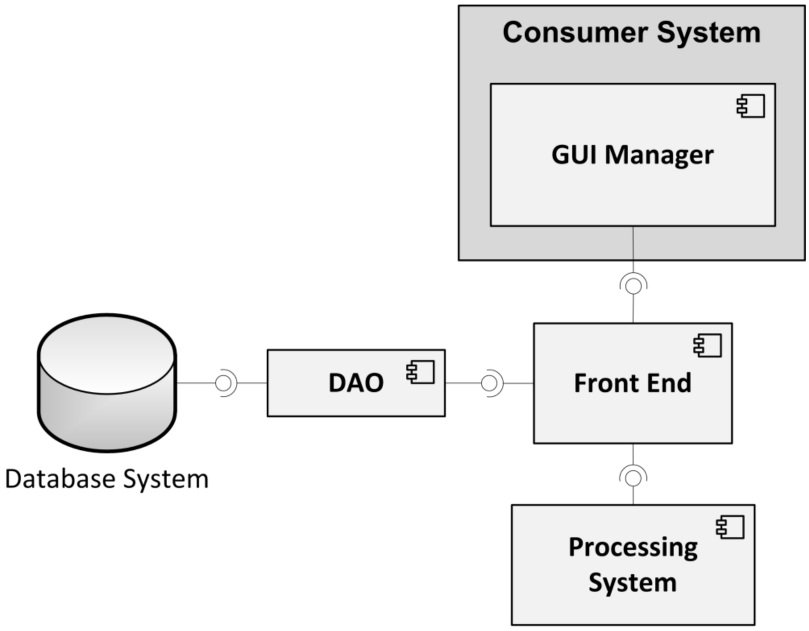


**Fig. 6.6:** Consumer System Component Structure

The interactions between the SESOA consumer system and the processing system can be either static or dynamic interactions. Static interactions can be for example calls to specific Web Services retrieved from the semantic service repository where services' end points are important to achieve such static call. Dynamic interactions are done automatically while the user executes some workflow activities that do not require any user interaction.

#### 6.1.3.3 Provider system

Most of the SESOA business process functionalities are realized using Web Services technology. These services are supplied by different service providers. Therefore, the provider system in SESOA represents all the internal and external service providers that are supplying Web Services to the system. It deals with HTTP[45] incoming and outgoing user's requests and contains functions required for providing Web Services. In addition, it has two interfaces with the Web Service directory and the semantic service directory to allow service publishing. The first interface with the external service directory is direct and any XML-based registry like UDDI can be used. It has also interfaces with the SESOA semantic service directory, the validation system and the semantic Web Service-based system or more precisely its assemblage unit. The interface between the provider system and validation system is to optionally validate the supplied Web Services. Web Services are registered in WS-assemblages and their relations are then annotated using RDF statements to be published in the semantic service directory.

[45] HTTP stands for Hypertext Transfer Protocol (Fielding et al., 1999).

Figure 6.7 illustrates the internal component architecture of the provider system. It differentiates between internal and external service providers. On the one hand, the internal service provider delivers business-specific and foundation services. These internal services are hosted on a Web Server and connected to SESOA database system via DAO interface. On the other hand, the SESOA reference architecture enables external service providers to supply their services to the system. These providers are also hosted by individual Web Servers and have access to their own databases via DAO interfaces. Both types of service providers publish their services at the system's semantic service repository and other traditional Web Service registries.

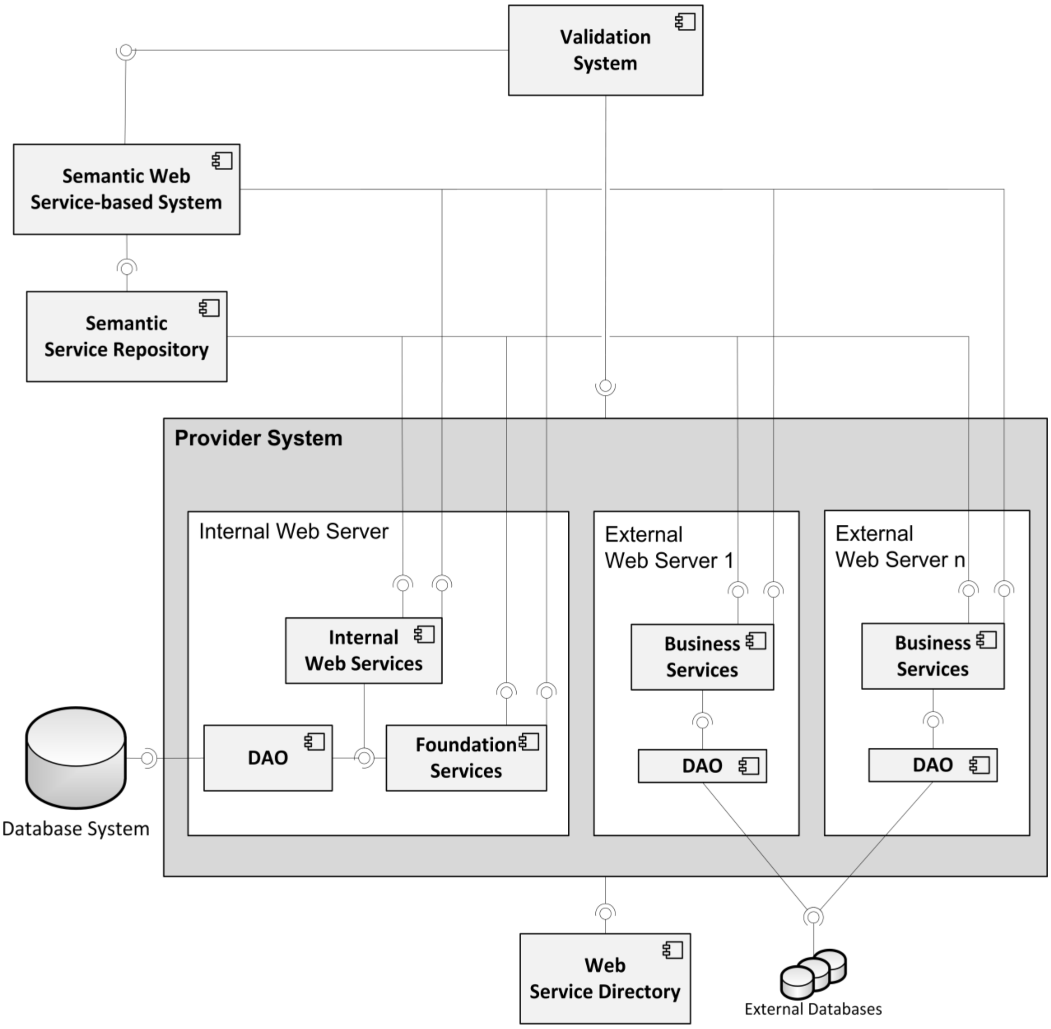


**Fig. 6.7:** Provider System Component Structure

The process of adding Web Services to the system starts by creating a Web Service, registering it in WS-assemblage to be later published at the semantic service repository.

### 6.1.3.4 Web Service Directory

This section illustrates the internal structure of the Web Service directory component. This component is part of the typical service-oriented paradigm and it is not part of the

SESOA system, rather it is together with the provider system considered as external components. Web Service directory represents the yellow pages where advertised Web Services coming from different service providers can be published and then discovered by service consumers. Upon service publishing, service consumers can discover the Web Service directory component to consume the services that fulfill their needs. Figure 6.8 shows the internal architecture of the Web Service directory component. It is hosted by a Web server that has a unique address. This address is known by all the service providers (SPs) that are linked to the "look-up/registry" Web Service directory's internal component. All service information can be located at this component. Service consumers can be bound to the services supplied by service providers after discovering them in this component.

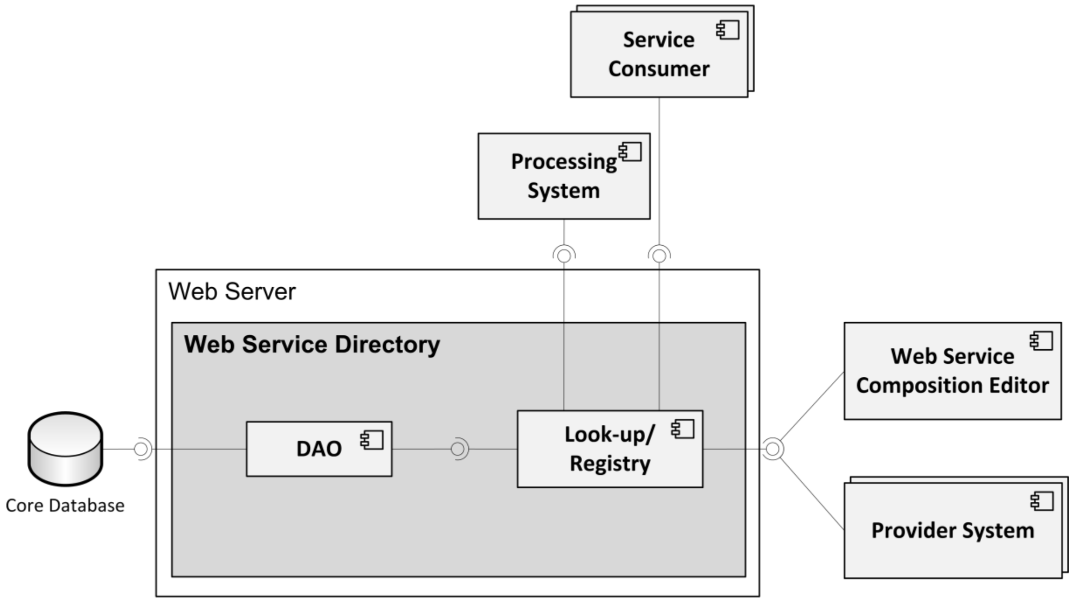


**Fig. 6.8:** Web Service Directory Component Structure

Since the system is based on different business processes and services using workflows and Web Services, the processing system contacts the look-up/registry subcomponent to fetch list of services needed to execute workflows. Moreover, all information the Web Services advertised in the Web Service directory component are stored in the SESOA database system via DAO interface. Finally, the Web Service composition editor can discover the registry subcomponent to determine at design time what are the required functionalities offered by the existing Web Services needed for composing new Web Services. A service that is registered in the Web Service directory can be used as input for a composition of new services. Figures 6.9 and 6.10 show the sequential composition and the parallel composition of Web Services respectively.

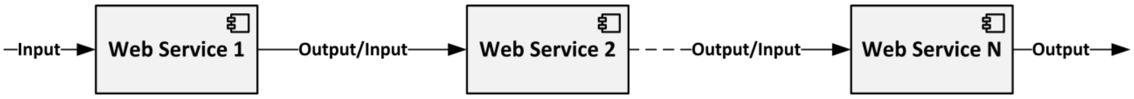


**Fig. 6.9:** Sequential Composition of Services

In the sequential Web Service composition, the output of a Web Service can be used as input for the next one. In the above example of Figure 6.9, the output of Web Service 1 is used as input for Web Service 2 and so forth until the sequence of services ends.

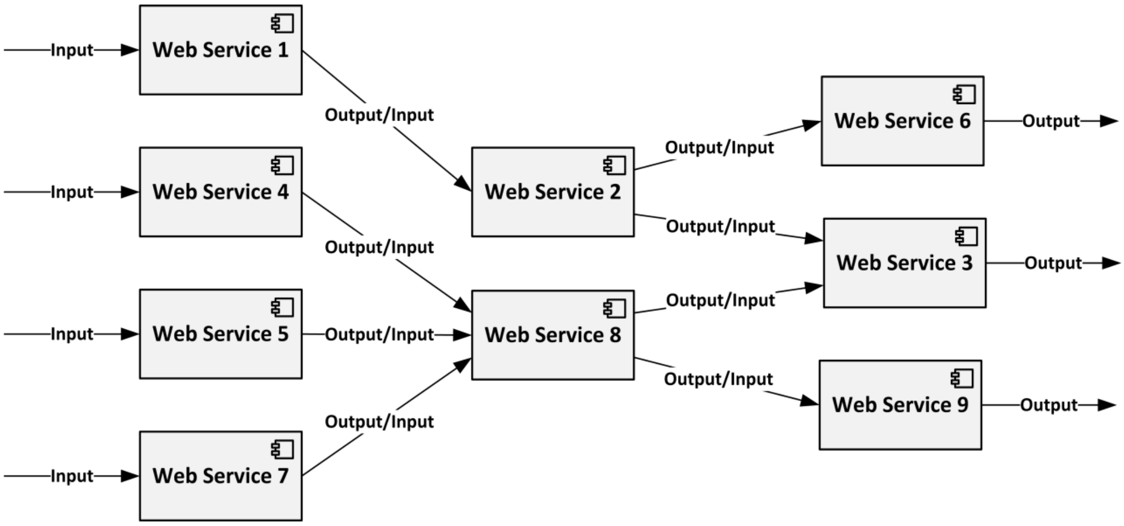


**Fig. 6.10:** Parallel Composition of Services

In contrast to the sequential composition, in the parallel composition of Web Services outputs of several services can be used as input for one or more subsequent service as depicted in Figure 6.10.

Composing Web Services requires making clear distinction between service orchestration and service choreography. While service orchestration describes the organization of service interactions including the business logic and the interactions order, service choreography describes the sequence of messages between participants focusing more on the public exchange of messages and conversational state.

***Service Composition Architectural Styles***: This section lists some common architectural styles used for service composition as they are implemented in SOA (Rosen et al., 2008, pp. 279–292). The two styles that are commonly investigated are firstly the hierarchical and conversational composition and secondly the conduct-based and peer-to-peer composition.

*Hierarchical and Conversational Composition*: In the hierarchical service composition, the composition implementation is entirely hidden from service consumers. It is seemed to be more as a “black box”. This composition style is used to implement solutions that do not have human or any other interactions from the solution invoker. As for the conversational composition, the implementation is still hidden from the consumer but selected execution results when the consumer needs to have control on the execution are exposed. This composition approach is often used when the path of composition execution cannot be determined without additional inputs from the service consumer based on intermediate execution results.

*Conductor-based and peer-to-peer composition*: In the conductor-based composition approach, there is a conductor interacts with the service consumer. The whole execution control of the different services needed in this composition style is managed by the conductor. As for the peer-to-peer composition approach, every service involved in the composition has a specific orchestration responsibility in the overall composition process. Relying on the individual rules of each service in the composition process, a central coordinator is not required.

There are common approaches used for the implementation of the abovementioned service composition styles. These approaches include programmatic composition that is taking usage of statically writing and compiling code in general-purpose languages like C#, Java, etc. Other approaches are event-based, Service Composition Architecture (SCA), and orchestration engine-based compositions.

#### 6.1.3.5 Semantic Web Service-based System

The semantic Web Service-based system represents the SESOA component in which the semantic annotation management of Web Service relations is the main task. This subsystem handles Web Service requests required to execute workflow activities coming from the processing system component (the execution of workflow activities is initiated by the consumer system via the SESOA’s front end). This component coordinates the semantic annotation of Web Services relations and publishes these annotations in the semantic service repository. The responses to the processing system with the services’ availability and information are managed by this component as well. This is done using the assemblage unit subcomponent that has an interface with the processing system component to assist it in executing workflows by responding with services’ endpoints. It has another interface with the semantic service repository to publish the semantic annotations. Moreover, assemblages and Web Services information are stored in system’s database via the DAO interface between the assemblage unit and the SESOA database system. The internal architecture of this component is depicted in Figure 6.11.

The management of Web Service registration as assemblages’ members within the assemblage unit subcomponent is done with the help of two operators namely the membership operator and the assemblage operator. Via the validation system, each service provider sends a request to register his services within a desirable assemblage to the membership operator. This operator updates the assemblage list that contains a list of all assemblages with the registered services coming from this service provider. The membership operator forwards then the register request to the assemblage operator that allows the service to join the members list that contains all the assemblage-members (the registered services) information. Upon successful registration, the membership operator updates the description of the assemblage in the assemblage unit and the new information is made available in the semantic service repository. More information about the service registration is provided in Section 6.2.

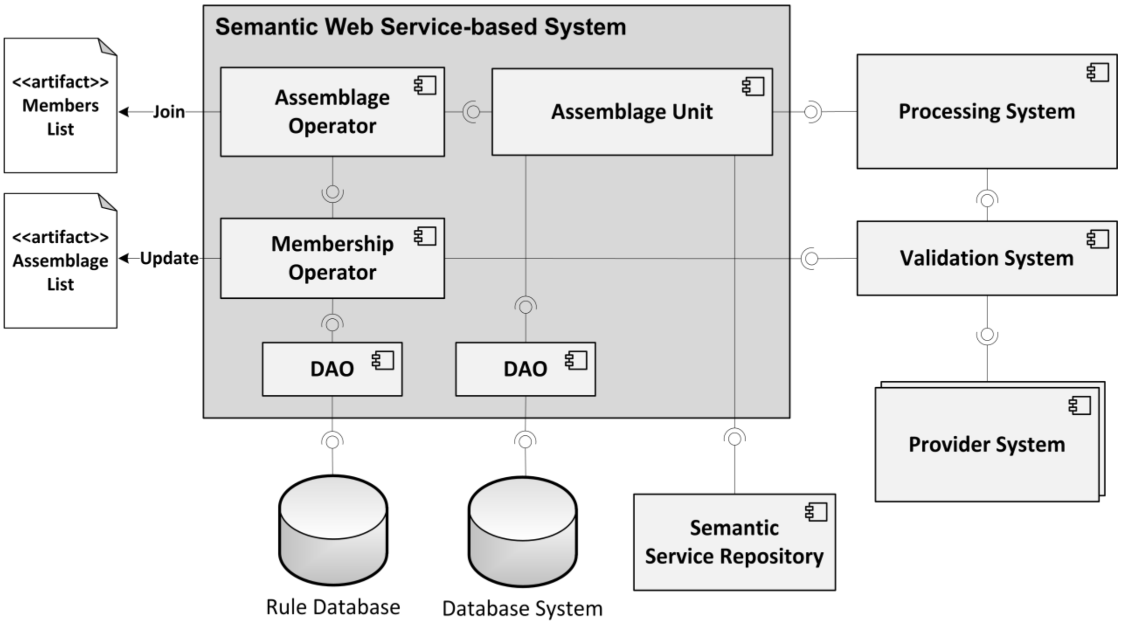


**Fig. 6.11:** Semantic Web Service-based System Component Structure

Finally, the membership operator has an interface with the rule database. This database actually contains all the required rules that enable this operator to react to potential changes that might arise in the definitions of assemblages.

### 6.1.3.6 Validation System

SESOA's validation system component provides set of validation tests to the Web Services supplied by different service providers. It is liable to adapt external Web Services provided by the provider system to enrich them with the correct semantic annotation. Four subcomponents within this system realize its functionality. The internal architecture of the validation system is depicted in Figure 6.12.

The subcomponents in the validation system are the service test unit, the announcement unit, the semantic annotation provider and the validation repository. The service test unit subcomponent provides an interface to the provider system for testing the adapted Web Services at quantitative and qualitative levels. Quantitative tests are compatible with the classical availability tests like checking the maximum load, response times or availability of a specific Web Service. Qualitative tests are validating the Web Service's compliance. If a Web Service passes the validation test, a service identifier is given to the annotation provider unit to start the process of registering the service as member in one or more assemblages and to annotate their relations with proper RDF statements.

This annotation process can be done optionally either before or after validating the Web Service using the annotation provider unit. Unlike other approaches, annotations provided by Web Services themselves are ignored and the system deals only with the annotations made by the annotation provider unit subcomponent. The main benefit behind this is to enable Web Services reusability within the system.

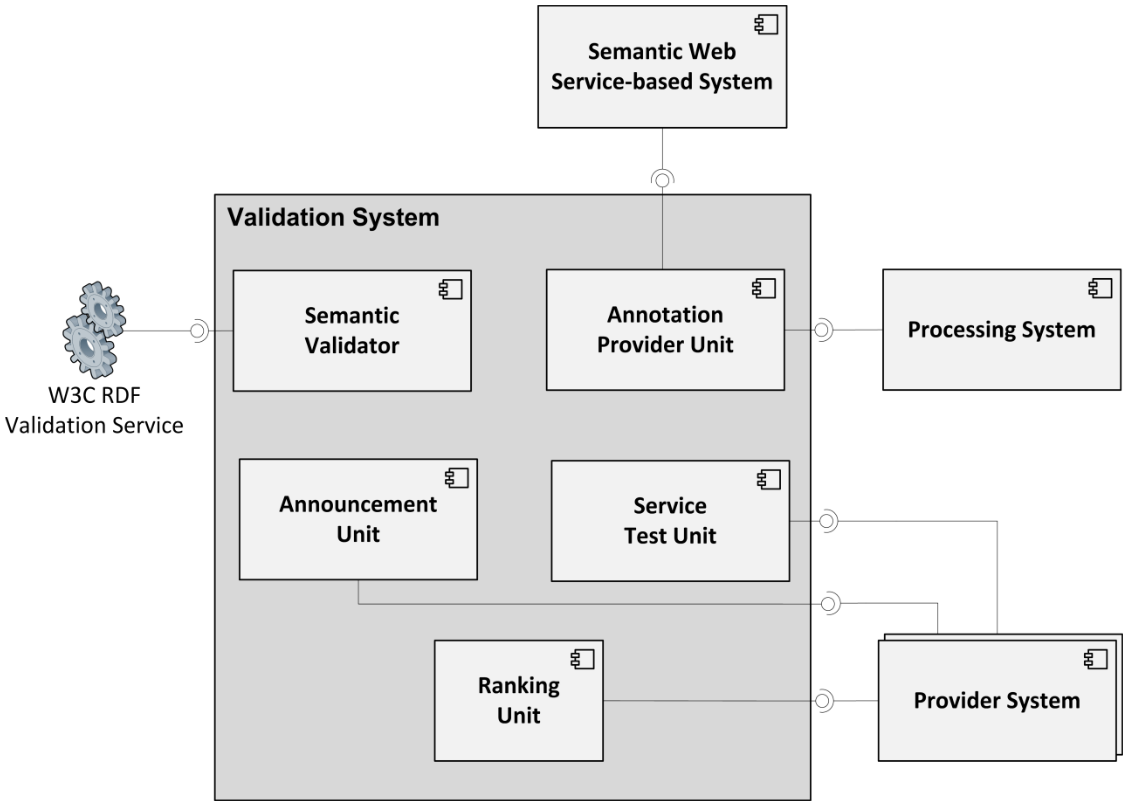


**Fig. 6.12:** Validation System Component Structure

Web Service registration in assemblages in form of workflow execution is enabled using the interface between the annotation provider unit and the SESOA processing system component. Ranking unit subcomponent is used to provide automatic ranking of provider system's Web Services based on secure rating machine-to-machine evaluation protocol. This protocol is based on two criteria: the response time and the availability of Web Services.

Via the interface with the provider system, the announcement unit subcomponent exposes new functionalities (that are not available in the system) and implementation requests to the service providers. This opens the road to SMEs and even individuals to implement such functionalities and make profit out of that.

Finally, testing RDF statements that annotate the relations between Web Services and assemblages can be achieved using the semantic validator that applies RDF validation using the RDF validation service provided by the World Wide Web Consortium (see http://www.w3.org/RDF/Validator/).

#### 6.1.3.7 Semantic Service Repository

The SESOA semantic service repository is the component responsible for publishing the lightweight semantic Web Services annotations to make them available to both service providers and consumers. Unlike traditional Web Service directories, what is stored in the semantic service repository is the semantic representation in form of RDF statements for the Web Services' relations. The semanticized relations are advertised in this repository because typical UDDI registries do not handle the semantic representations done by the annotation provider unit within the validation system.

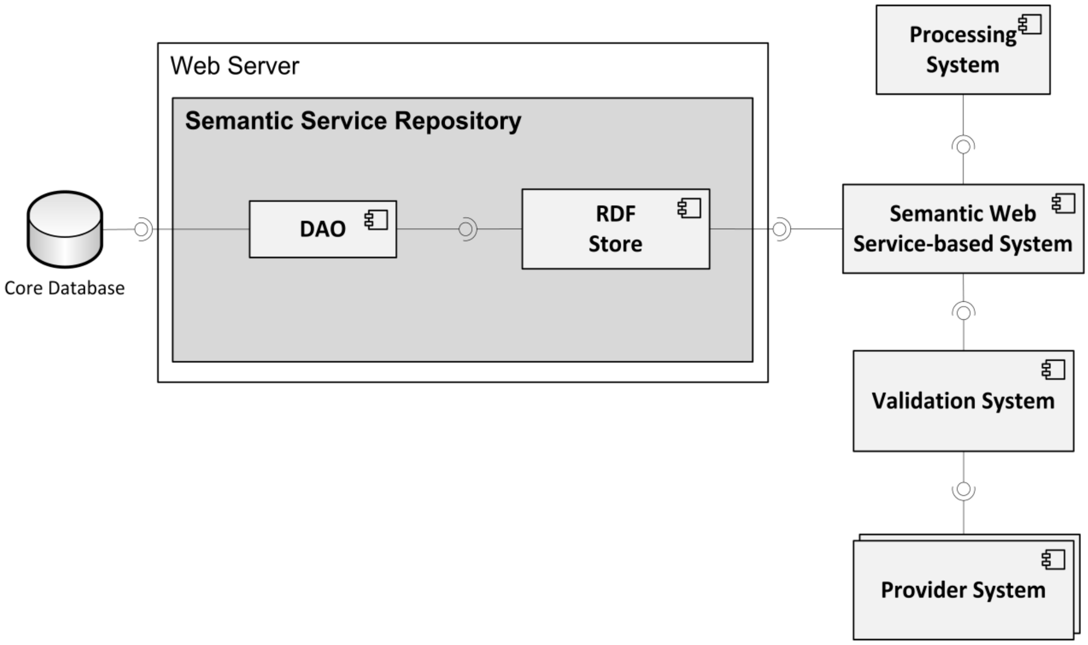


**Fig. 6.13:** Semantic Service Repository Component Structure

As illustrated in Figure 6.13 above, the RDF store subcomponent within the semantic service repository is responsible of representing the semantic RDF statements of Web Services-assemblages relations and exposes them in RDF-readable format.

While executing workflows in the processing system component, some activities need to be executed in form of calling Web Services published at the semantic service repository. The semantic Web Service-based system (the assemblage unit subcomponent) gets the Web Service call requests, look up in the RDF store to find the service information and responds back to the processing system. Furthermore, the provider system can allocate in the RDF store, via the validation system, the assemblages in which its Web Services are going to be registered. New Web Services that are registered in assemblages are also published in this repository.

### 6.1.3.8 Database System

This section describes some of the SESOA database system's interfaces as illustrated in Figure 6.14. The main focus in this section is given to the semantic representation of Web Services and assemblages relations.

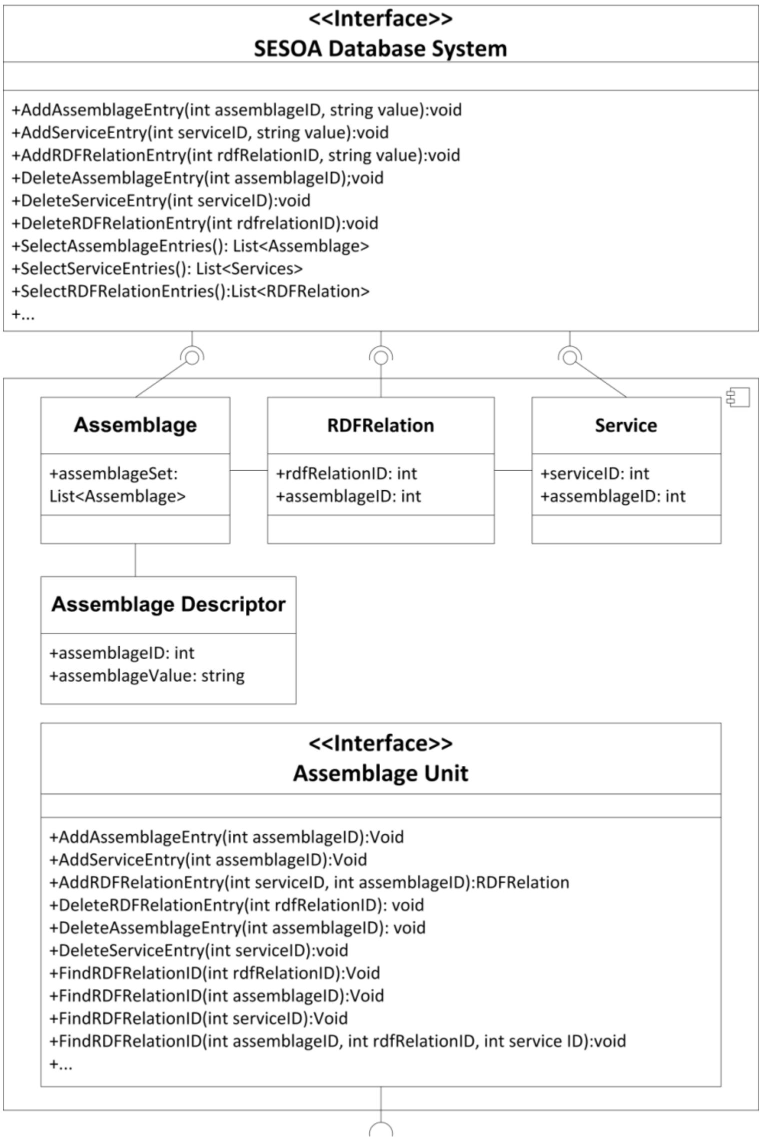

<<Interface>>
SESOA Database System
+AddAssemblageEntry(int assemblageID, string value):void
+AddServiceEntry(int serviceID, string value):void
+AddRDFRelationEntry(int rdfRelationID, string value):void
+DeleteAssemblageEntry(int assemblageID);void
+DeleteServiceEntry(int serviceID):void
+DeleteRDFRelationEntry(int rdfrelationID):void
+SelectAssemblageEntries(): List<Assemblage>
+SelectServiceEntries(): List<Services>
+SelectRDFRelationEntries():List<RDFRelation>
+...
Assemblage
+assemblageSet: List<Assemblage>
RDFRelation
+rdfRelationID: int
+assemblageID: int
Service
+serviceID: int
+assemblageID: int
Assemblage Descriptor
+assemblageID: int
+assemblageValue: string
<<Interface>>
Assemblage Unit
+AddAssemblageEntry(int assemblageID):Void
+AddServiceEntry(int assemblageID):Void
+AddRDFRelationEntry(int serviceID, int assemblageID):RDFRelation
+DeleteRDFRelationEntry(int rdfRelationID): void
+DeleteAssemblageEntry(int assemblageID): void
+DeleteServiceEntry(int serviceID):void
+FindRDFRelationID(int rdfRelationID):Void
+FindRDFRelationID(int assemblageID):Void
+FindRDFRelationID(int serviceID):Void
+FindRDFRelationID(int assemblageID, int rdfRelationID, int service ID):void
+...

**Fig. 6.14:** Interfaces of the SESOA Database System

As shown in the previous figure, the database system offers several interfaces to the assemblage unit subcomponent of the semantic Web Service-based component. In this latter, three main entities are used to interact with the database system. These entities are the assemblage, the RDF relation and the service. The main functions that are implemented at the database system's interface are listed and described in Table 6.4. It includes just some functions that are used to create the semantic relations as RDF statements between assemblages and Web Services and does not include all of the database system's functions.

**Tab. 6.4:** Function Set of Database System - Assemblage Unit Interface

| **Function Name** | **Description** |
|---|---|
| **AddAssemblageEntry** | Only system administrators can add assemblages to the system |
| **AddServiceEntry** | Service providers can add their services' information to the system and register them within suitable assemblages |
| **AddRDFRelationEntry** | Upon service registration, automatic semantic RDF-based relation is created between the Web Service and the assemblage in which it has been registered |
| **DeleteAssemblageEntry** | Only system administrators can delete assemblages from the system taking into account that deleting assemblage includes the deletion of all the registered services within it |
| **DeleteServiceEntry** | Delete service information is performed by system administrators |
| **DeleteRDFRelationEntry** | Relations are deleted automatically when services or assemblages are deleted |
| **SelectAssemblageEntries** | Query for a specific assemblage |
| **SelectServiceEntries** | Query for a specific Web Service |
| **SelectRDFRelationEntries** | Query for a specific relation between assemblage and Web Service |

Similar functions are available at the assemblage unit to achieve the semantic annotation. However, what have been explained so far includes just the highlights to the important functions required to apply the semantic annotation at the SESOA database system. In the following parts of this section, some details are given regarding user management with user rights together with the main data model.

### *User Management*

As mentioned before, the whole management-related data of system users are stored in the SESOA database system. Based on that, an appropriate component for managing the users with their roles and rights constitutes an integral component of the system prototype. User management supports the system administrator to create, modify and maintain user accounts and it enables also the allocation of rights for selected users and groups (roles). Each registered user in the system has to be assigned with at least one role or user group. Group permissions can be inherited by individual registered users. When a new service is registered within the system, it will be assigned with the suitable rights.

### *Data Model*

In the following the terms user, workflow, activity, service, assemblage are going to be formally defined. Any user in this system has a user ID, name, password, email, and other optional data. Proper authorization method and roles to be assigned to each user have to be defined as well. To summarize that, the user is defined as follows:

$\boldsymbol{User}\ (\boldsymbol{userID}, \boldsymbol{name}, \boldsymbol{password}, \boldsymbol{email}, \ldots, \boldsymbol{authorization}\ [\boldsymbol{Authorization}], \boldsymbol{role}\ [])$

Workflows in this system are also defined in a way in which each workflow has workflow ID, name, description, hierarchy, and status. The formal definition of system's workflow is defined as follows:

$\boldsymbol{Workflow}\ (\boldsymbol{workflowID}, \boldsymbol{name}, \boldsymbol{description}, \boldsymbol{hierarchy}, \boldsymbol{status})$

Hierarchy here represents [0...n] activities that constitute the workflow with links between them. As for a workflow status, it can be running, sleeping, bookmarked, or stopped and is defined as follows:

$\boldsymbol{WorkflowStatus} = \{\boldsymbol{running} \mid \boldsymbol{sleeping} \mid \boldsymbol{bookmarked} \mid \boldsymbol{stopped}\}$

The workflow status guarantees the robustness of the system and enhances its responsiveness while users interact with the system. As mentioned above, each workflow contains activities that form transient members, depend on user interactions, and can be realized as Web Services. A workflow activity is defined as follows:

$\boldsymbol{Activity}\ (\boldsymbol{User\ Interaction} \mid \boldsymbol{Web\ Service})\ //\boldsymbol{transient}$

Based on activity definition, the Web Service definition can be excluded. Every Web Service can be registered as a member in one or more specific assemblages [1…n]. Generally, each Web Service has an ID, name, description, assemblage ID, and endpoint. The data model gives the possibility that authorization can be applied on the service level. Service Definition is described as follows:

$\boldsymbol{Service}\ (\boldsymbol{serviceID}, \boldsymbol{name}, \boldsymbol{description}, \boldsymbol{assemblageID}, \boldsymbol{AUTHORIZATION})$

Service description includes detailed description of the functionality besides detailed description of the formal parameters and return values (including error cases) of this service. Service grounding whether to use SOAP-based or RESTful-based grounding approaches is determined here as well. Eventually, service endpoint is set by the service provider to locate the service information (service contract) based on specific URL.

Finally, the assemblages in which the Web Services are grouped are defined where each assemblage has assemblage an ID, name, description, and domain. WS-assemblage is defined as follows:

$\boldsymbol{Assemblage}\ (\boldsymbol{assemblageID}, \boldsymbol{name}, \boldsymbol{description}, \boldsymbol{domain})$

Assemblage's domain represents the wide business domain in which broad set of services can be registered. This makes it easier for the service providers to decide in which assemblage their services have to be registered.

The main focus in this section was given to the organization of Web Services and assemblages and how the data model handles them. User management was also discussed in this section. Other data that are managed and stored in the SESOA database system are the business case data together with the entities resulting from the assemblages and Web Service RDF statements.

The following sections of this chapter explain the Web Service registration scenario and give some details on how the services are annotated using semantic RDF statements. Furthermore, service validation and evaluation are described as well.

## 6.2 Web Services Registration

This section explains the scenario in which Web Services are registered within assemblages. The whole registration scenario is depicted in Figure 6.15. System administrator creates assemblages that cover different business domains within the enterprise to which the system is going to be applied. These assemblages are published in the semantic service repository and made available to the service providers via the validation system's interface.

Service provider can search this repository via the validation system interface to find the suitable assemblage(s) that fulfill the provider's business needs. Service providers register then the Web Services in one or more desired WS-assemblages after passing an optional validation process. In the case that there is no assemblage matches what a service provider seeks a request to the system administrator via the assemblage unit interface within the semantic Web-Service-based system will be created. Upon proofing this request, the system administrator creates a new assemblage and advertises it in the semantic service repository to be made available to all service providers via the validation system's interface.

In the case that the service provider finds the desired WS-assemblage that suits his interests and after passing an optional validation process, a registration request is forwarded to a membership operator. The membership operator is responsible of the following tasks:

- It negotiates another operator called the assemblage operator to complete the registration process.

- It updates WS-assemblages list that includes all the assemblages in which the Web Services of this specific service provider are registered.

- It has a link to the rule database that contains set of rules enabling the membership operator to react to any changes might be issued by the semantic Web-Service-based system (more specifically, the assemblage unit).

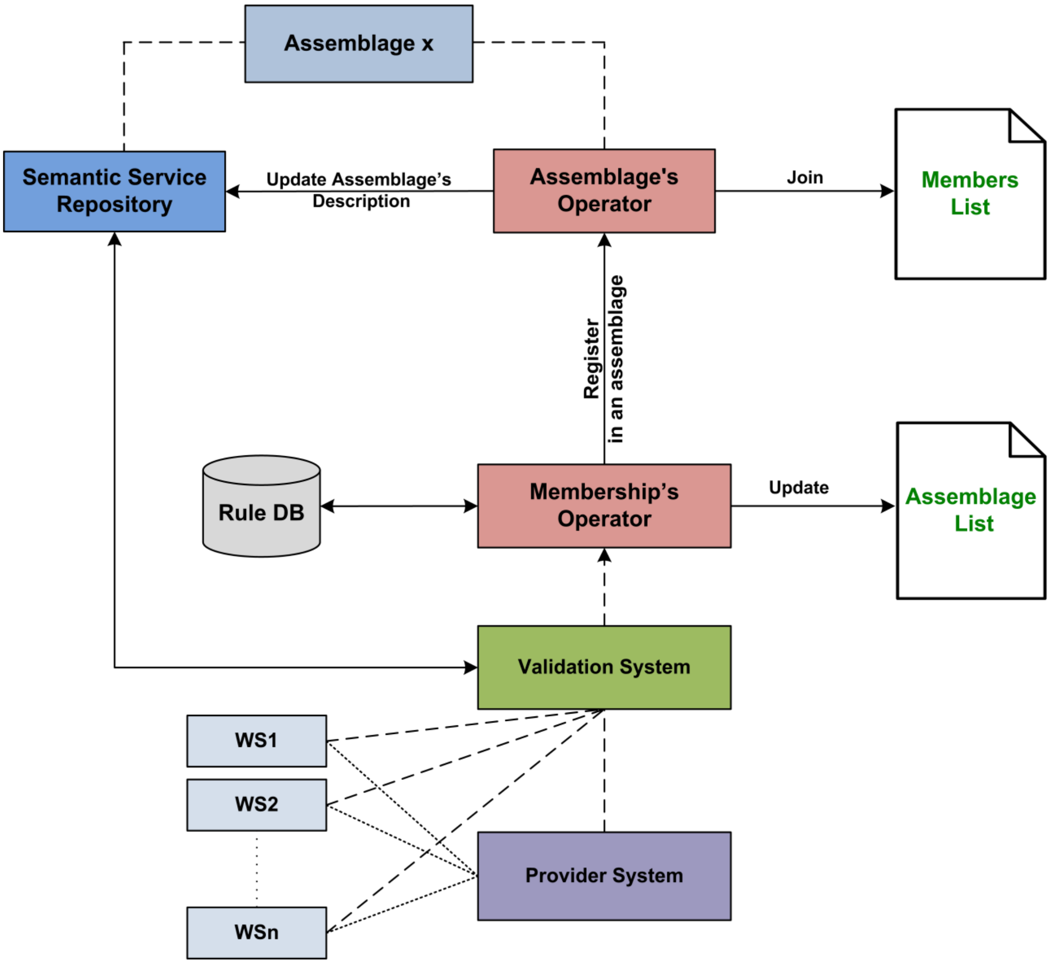


**Fig. 6.15:** Web Service Registration Scenario

As for the assemblage operator, it has the following responsibilities:

- It receives the registration request from the membership operator to finalize the Web Service registration within the desired assemblage.

- It updates the assemblage's member list that contains all the already registered Web Services.

- It updates the WS-assemblage's description to be re-published in the semantic service repository.

## 6.3 Web Service Validation

One goal behind the SESOA reference architecture is to annotate information with semantic meta-data. From semantic annotation point of view and in comparison with other annotation concepts like the ones in (Handschuh & Staab, 2003), the annotation used in this work is not a part of the Web Service itself. Instead of this, the annotation is done at the relation level between the services and the assemblages. An optional requirement in this work is to validate the Web Services based on specific validation criteria. These criteria were focusing mainly on the functional level of services including some data types (like Integer, String…) that the services use to realize their functionalities. Different sets of validation tests were built following specific type of Web Service compliance. Based on (Mahmoud, von der Dovenmühle, & Marx Gómez, 2009), the Web Services compliance can be classified into four main different types:

- Exact compliance in which a Web Service is able to comply with the predefined requirements (Web Service properties)
- Over-exact compliance in which a Web Service has a higher compliance than expected
- Partial compliance where a Web Service is able to comply with the predefined requirements fractionally
- Improper compliance where a Web Service fails in complying with the predefined requirements.

An exact compliance is the ideal situation in which a Web Service fulfills exactly all the expectations. An over-exact compliance happens when the Web Service provides a higher level of quality more than what expected by the service consumer.

The partial compliance of a Web Service happens when the requirements can be divided into logical parts. If the Web Service complies with a part, then it can be used to handle the information in collaboration with another Web Service. Eventually, failure compliance happens when a Web Service does not comply at all with the request or even with parts of that request.

The following example clarifies ideas behind Web Service compliance types: suppose a customer data set that has to be validated and stored. It contains his first name, last name, email address, and birth date. The validation of the email address and the birth date can be done partially and independently from storing the customer's information.

$$\boldsymbol{O_{request} \coloneqq \{firstName, lastName, emailAddress, birthDate\}}$$

$$\boldsymbol{O_{divided} \coloneqq \{d_1(firstName, lastName); d_2(emailAddress); d_3(birthDate)\}}$$

The following Web Services: $WS_1$ and $WS_2$ are partially complaint with the request where $WS_1$ complies with the customer's email address and $WS_2$ complies with his birth date. They are used to handle the validation. Another Web Service $WS_3$ is an exact match since it complies with all of the customer's information.

$$\boldsymbol{Mapping_{partial}\{d_2 \Rightarrow WS_1; d_3 \Rightarrow WS_2\}}$$

$$\boldsymbol{Mapping_{exact}\{d_1, d_2(WS_1), d_3(WS_2) \Rightarrow WS_3\}}$$

From a provider's perspective, this gives a possibility to integrate low capacity Web Services into ambitious requests by making crossing among the possible requests. However, the challenge in such context is to analyze the structure of the information not only by humans, rather by humans and machines. To do so, the person object itself has to be broken down into properties.

Typically, literals are used to store information like the name of a person and in this way, a first name can be everything described as an array of characters. However, in the real world this is not totally true because there are small quantities of character combinations that are valid to be first names. Considering this, the correct data type for the property first name would be an enumeration or an object that contains a table of valid entries. The person's email address is not a random literal too and the previous example showed that the design of the object has to start at the level of the properties not the level of entities.

## 6.4 Web Service Evaluation

This section illustrates how the secure Web Service evaluation is applied within the SESOA prototype. This evaluation is based on a machine-to-machine security protocol. This security protocol has been developed in the thesis of (Hasan, 2010) to evaluate Web Services in SOA-based systems. The main aspects of this protocol are derived from the one designed by (Brehm & Marx Gómez, 2010) and adapted to meet the requirements of the new SESOA service reputation. The evaluation process must guarantee that the service provider can't make any manipulation to the reputation value of any of its Web Service.

### 6.4.1 Security Protocol for the Evaluation of Web Services

Since different criteria are taken into account in the process of calculating the reputation value of an invoked Web Service, this security protocol uses the Multi-Attribute Utility Theory (MAUT) algorithm to determine this value (Schäfer, 2001). This section describes this protocol with its main participants. Moreover, it determines which security algorithm is selected to be applied within this protocol.

Three participants are involved in this security protocol. These participants are the service provider, service consumer and the Evaluation Processing Authority (EPA). The service provider makes the service URL and the EPA's public key available to the service consumer. The consumer measures the response time value of the called Web Service and sends it to the EPA. This value is sent only if the Web Services is available; otherwise the service consumer notifies the EPA that the Web Service is not available. This is also applied to the availability of the called Web Service. These entries (the service's response time and availability) are then encrypted using the EPA's public key. The EPA represents a trusted third entity for both service consumer and provider. It receives the encrypted entries that are sent by different service consumers and then decrypts these entries using its own private key. Eventually, using the decrypted values, the EPA calculates the final reputation value of the corresponding Web Service using the MAUT algorithm.

This security protocol makes use of many security algorithms that are necessary to fulfill the security objectives. These algorithms are briefly presented in Table 6.5.

**Tab. 6.5:** The Security Algorithms used in this Work

| Algorithm Name | Description | Usage in This Work |
|---|---|---|
| **AES**[46] | A symmetric encryption algorithm | XML Encryption |
| **RSA**[47] | An asymmetric encryption algorithm | XML encryption<br>XML digital signature |
| **SHA-1**[48] | A cryptographic hash algorithm | XML digital signature |

The AES algorithm is used in this security protocol because compared with Data Encryption Standard (DES) and Triple-DES algorithms, it is accorded faster and more secure (Pachghare, 2009). RSA is used in this protocol as an asymmetric encryption algorithm. Moreover, SHA-1 is used in this work to calculate the hash values because recently this algorithm is widely used as the standard hash algorithm (Stallings, 2010).

The main concept of this security protocol is divided into three different phases. In *the first phase,* the inputs are calculated to be sent to the EPA. During the invocation of a Web Service, the consumer calculates the values (the inputs like response time) that are required to determine the evaluation value in the EPA. The calculation of the evaluation value is done at the EPA using different evaluation (reputation) criteria. Examples of

[46] Advanced Encryption Standard (AES) was developed in 2000 by Joan Daemen and Vincent Rijmen (Daemen & Rijmen, 2002).

[47] RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who defined this algorithm firstly in 1978. According to (Kaliski Jr & Redwood City, 1991), RSA is part of the Public-Key Cryptography Standards (PKCS).

[48] SHA-1 stands for Secure Hash Algorithm (Eastlake & Jones, 2001).

these criteria are the Web Service's response time, availability, and reliability. However, this security protocol considers only the response time and the availability as evaluation criteria.

Based on that, the service consumer calculates the response time as an evaluation criterion. The response time is measured by determining the difference between the time when the request was sent and the time in which the response has been received. The response ($\Delta_t$) time is defined as follows:

$\Delta_t = t_{response} - t_{request}$

Where ($t_{request}$) is the time of SOAP-request and ($t_{response}$) is the time of SOAP-response. Using an encrypted SOAP message, the service consumer sends then the $\Delta_t$ value as a parameter to the EPA in the case that the Web Service is available. If calling the Web Service was not successful (with failures like "timeout" or "HTTP status code (404) not found"), the service consumer assigns zero (0) to the $\Delta_t$ value referring that the Web Service was not available and sends this value to the EPA. To sum up the first phase activities, the service consumer:

- Assigns the zero value to the response time: $\Delta_t = 0$
- Calculate the $\Delta_t$ of the invoked Web Service
- Generates a symmetric key $k$
- Conducts XML encryption using $k$ and $EPA_{publicKey}$ by encrypting an XML element in the SOAP message that represents the response time. This results in an encrypted SOAP message that contains the encrypted response time.
- Sends this encrypted SOAP message to the EPA

While calculating the response time, one of two values is sent:

$\Delta_t > 0$ ⇒ The Web Service is available or

$\Delta_t = 0$ ⇒ The Web Service is not available.

In *the second phase*, The EPA receives the inputs calculated in the first phase as encrypted SOAP messages. It uses its private key to decrypt the messages. Therefore, the EPA firstly decrypts the symmetric key that was used by the Web Service consumer to encrypt the XML element using his private key. Then the EPA uses this symmetric key to decrypt the encrypted XML element.

Upon having the decrypted response time of the invoked Web Service, the EPA calculates the reputation value for the Web Service based on the response time and availability as evaluation criteria. Table 6.6 presents the evaluation criteria considered in the security protocol used in this work.

**Tab. 6.6:** The Evaluation Criteria

| Criteria | Parameter | Evaluation | Value |
|---|---|---|---|
| **Response Time** | $\Delta_t$ | $E(\Delta_t)$ | 1 to 5 |
| **Availability** | $A$ | $E(A)$ | 1 to 5 |

On the one hand, the EPA evaluate the response time if $\Delta_t > 0$ using a scale of 1 to 5 that represents excellent, very good, good, satisfactory, and unsatisfactory respectively. The EPA increases (by one) the number of successful requests and the total number of requests for the corresponding invoked Web Service. Then the Web Service availability $A$ as defined by (Zeginis & Plexousakis, 2010) is measured by the EPA as follows:

$$A = \frac{successful\ requests}{total\ number\ of\ requests} * 100$$

On the other hand and if the $\Delta_t = 0$ the EPA sets the worst rating (5) for the evaluation value of the corresponding Web Service's response time. The EPA then increases only the total number of requests of the corresponding Web Services by 1. Subsequently, the availability of Web Services is calculated as mentioned in the previous equation.

As a result, the EPA has the rating for the response time. What applies for the response time is applied one-to-one to the availability using the same 1 to 5 scale where 1 is the best availability and 5 is the worst availability.

As listed in Table 6.7 and upon having the evaluation value for each of the evaluation criteria (the response time and the availability), the EPA can calculate the evaluation value (score) for the invoked Web Service using the MAUT algorithm.

**Tab. 6.7:** Calculation of the Evaluation Criteria

| **Availability $A$** | $A \leq 50\%$ | $A \leq 65\%$ | $A \leq 80\%$ | $A \leq 95\%$ | $A > 95\%$ |
|---|---|---|---|---|---|
| $E(A)$ | 5 | 4 | 3 | 2 | 1 |
| **Response Time $E(\Delta_t)$ in milliseconds** | $\Delta_t \leq 3000$ | $\Delta_t \leq 5000$ | $\Delta_t \leq 8000$ | $\Delta_t \leq 15000$ | $\Delta_t > 15000$ |
| $E(\Delta_t)$ | 1 | 2 | 3 | 4 | 5 |

Similar to the formula written by (Schäfer, 2001), the evaluation value of a Web Service's single call is defined as follows:

$$E_r = \sum_{i=1}^{n} E(c_i) * w_i$$

Where:

- ✓ $E_r$ is the evaluation of a Web Service's single call $r$
- ✓ $E(c_i)$ is the evaluation of the criterion $c_i$
- ✓ $w_i$ is the weight assigned to the criterion $c_i$
- ✓ $n$ is the total number of evaluation criteria

Assigning weights to the evaluation criteria is up to the service providers and may follows specific business needs. However, whatever method used to assign weights to the evaluation criteria, the weights have to follow the following condition:

$$\sum_{i=1}^{n} w_i = 1$$

Based on that the overall evaluation value $E(WS)$ of the corresponding Web Service is:

$$E(WS) = \frac{(E_1 + E_2 + \cdots + E_m)}{m}$$

Where: $m$ represents the total number of individual evaluations.

The procedure for determining the evaluation value of a Web Service and all the steps used in this phase are summarized in the Table 6.8.

**Tab. 6.8:** The Calculation of a Web Service's Evaluation Value in the EPA

| Step Number | Step Description | Action |
|---|---|---|
| **1** | Set the evaluation criteria | $c_i; i = 1..n$ |
| **2** | Evaluate each evaluation criteria | $E(c_i) \in [1..5]$ |
| **3** | Weight each evaluation criteria | $w_i; \sum_{i=1}^{n} w_i = 1$ |

| | | |
|---|---|---|
| **4** | The EPA calculates the evaluation value for a single call | $E_r(WS) = \sum_{i=1}^{n} E(c_i) * w_i$ |
| **5** | The EPA Calculates the evaluation value of a Web Service using the individual values of the single calls | $E(WS) = \frac{(E_1 + E_2 + \cdots + E_m)}{m}$ |

The *third phase* in this security protocol is updating the evaluation value of the invoked Web Service in the semantic service repository. The sending of this value is done using a SOAP message digitally signed by the EPA. To illustrate this phase, the EPA:

- Calculates a hash value using the algorithm SHA-1
- Encrypts this hash value using its private key ($EPA_{privateKey}$)
- Identifies the XML Signature element
- Adds this signature to the SOAP message
- Sends the signed SOAP message to the semantic service repository
- The semantic service repository verified this SOAP message by using SHA-1 algorithm and the EPA's public key ($EPA_{publicKey}$)
- If the SOAP message is valid, the semantic service repository changes the evaluation value for the corresponding Web Service.

As a result, the evaluation value for each Web Service is published in the semantic service repository and can be controlled only by the EPA. Therefore, this protocol guarantees that service providers can't manipulate the evaluation values of their own Web Service at any rate. The application of this security protocol with its three phases to the SESOA system is explained in the following section

### 6.4.2 Web Service Evaluation within SEAOA

As mentioned in Section 5.3.2.7, the reputation is managed by the EPA. The new architecture including EPA is presented below in Figure 6.16. The first four steps belong to the conventional SOA where the service provider publishes a description for each Web Service in the semantic service repository.

This description contains the required information to call the Web Service. The Web Service consumer searches the repository for a desired Web Service. Upon finding the

Web Service with the desired functionality, he/she can call this service directly from the service provider.

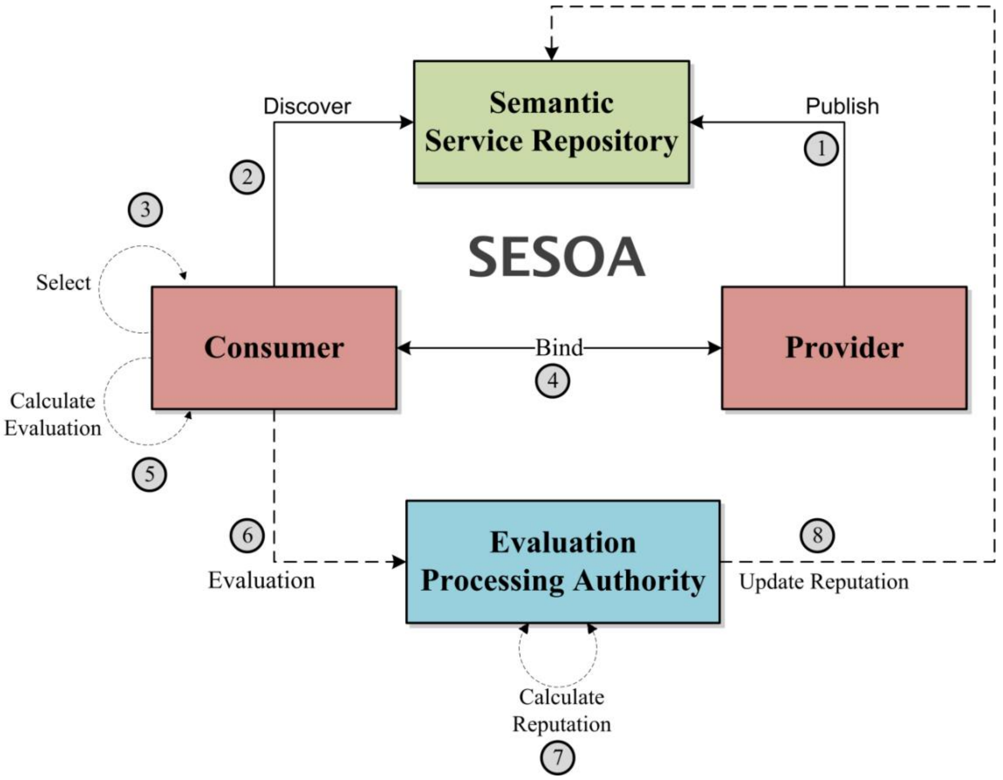


**Fig. 6.16:** Service Reputation in SESOA

After calling a Web Service, the consumer machine calculates the response time of this Web Service (step 5), and sends the calculated response time as an input to the EPA (step 6). The same applies to the availability criterion. EPA calculates then the reputation of the invoked Web Services based on these two criteria. After calculating this reputation instance, the EPA calculates the reputation value of the called Web Services as an average of all its individual reputation instances (step 7). The last step is to update the reputation value of the corresponding Web Service in the semantic service repository. The Steps (5-8) are based on the security protocol for the evaluation of Web Services (see Section 6.4.1).

## 6.5 Main System Interactions

This section gives a holistic overview of the system's interactions and how the interactions between its components are initiated using two sequence diagrams. Whereas the first sequence diagram shows the interactions among the SESOA components, the second one depicts the management of Web Services semantic annotation using RDF statements.

Figure 6.17 explicates how the main business logic is carried out within SESOA. It shows the message exchange between the main components in the reference architecture. The communication flow starts from the workflow engine subcomponent of the processing system. The reason behind that is that the business logic is represented as business processes and these processes are realized using workflows that are executed using the workflow engine.

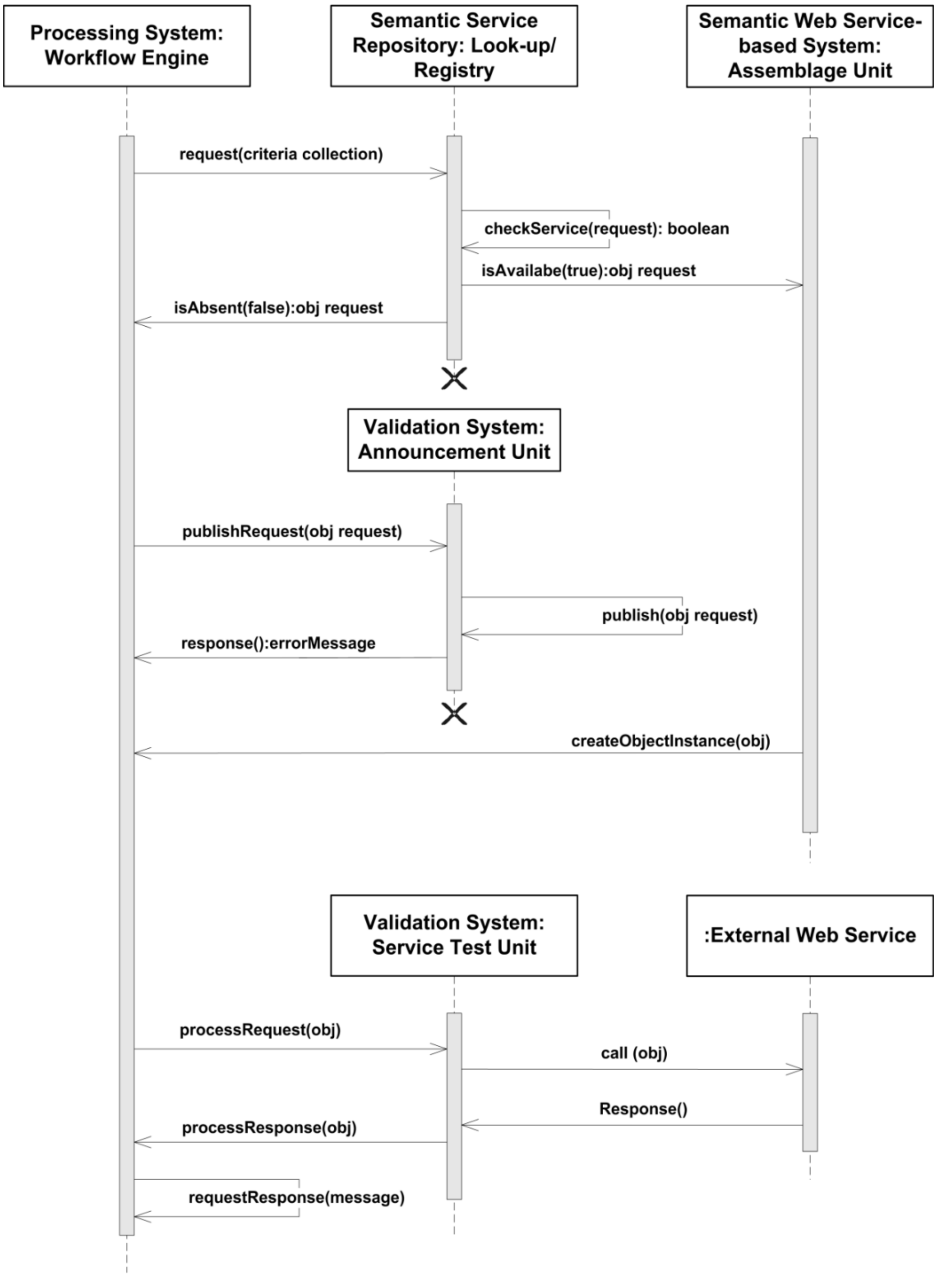

Processing System: Workflow Engine
Semantic Service Repository: Look-up/ Registry
Semantic Web Service-based System: Assemblage Unit
request(criteria collection)
checkService(request): boolean
isAvailabe(true):obj request
isAbsent(false):obj request
Validation System: Announcement Unit
publishRequest(obj request)
publish(obj request)
response():errorMessage
createObjectInstance(obj)
Validation System: Service Test Unit
:External Web Service
processRequest(obj)
call (obj)
Response()
processResponse(obj)
requestResponse(message)

**Fig. 6.17:** Main SESOA Component Interactions

The system's workflows have many activities that are realized using Web Services and the workflow engine needs to get the location information about these services in order to invoke them. For this purpose, the workflow engine sends a request to semantic service repository to locate the required services. More precisely, the look-up/registry subcomponent receives this request and checks whether the required service(s) information exists or not.

If the service is available, the request is forwarded to the assemblage unit of the semantic Web-Service-based system in order to check the RDF statement to know in which assemblage the service(s) located and to extract the endpoint information. The assemblage unit creates then an object instance request to be sent to the workflow engine to initiate the process of calling the service(s). At this point, the workflow engine sends a processing request to the service test unit within the validation system that in turn calls the external Web Service and gets the response from it. Eventually, the service test unit generates processing response back to the workflow engine that in turn continues executing the running workflow. In the case that the requested functionality is not fulfilled or offered by any existing service, the workflow engine gets this response from the semantic service repository and handles it with proper response by having a compensation activity or even by stopping the workflow execution. In parallel, the workflow engine forwards a publish request of the absent functionality to the announcement unit within the validation system. This request is then published by the announcement unit and made available to the different service providers connected to the system to potentially offer such functionality as a Web Service. As mentioned before, this opens the way to the SMEs and even the individuals to make profit out of the system by offering new services on demand.

The second general sequence diagram that complements the previously described one is the enrichment of relations between Web Services and assemblages with semantic annotations. Figure 6.18 depicts the process of adding semantic annotations to service relations illustrated as UML sequence diagram. When a service provider wants to register a Web Service in an assemblage, an automatic creation of RDF statements composed of assemblage, relationship, and Web Service entities is done. This is shown in the below figure and the process starts by checking the assemblage unit to see whether such relation does not already exist to prevent unnecessary duplication. Another checking at the semantic service repository component takes place to see whether the target assemblage exists and valid. These both functions return Boolean values and can be true or false.

If the relation exists in the assemblage unit or the assemblage in which the Web Service is going to registered does not exist, this means that the “addSemanticRelation” function will not be executed. Otherwise and if the relation does not exist and the assemblage exist, a new semantic relation between the Web Service and the assemblage is added to the assemblage unit. This relation is mainly RDF statement in which the subject is the assemblage, the predicate is the relationship (e.g. “hasMember”), and eventually the object is the actual Web Service that is already registered within the assemblage. The last step in this sequence diagram is storing the relation’s different entities in the SE-SOA database system

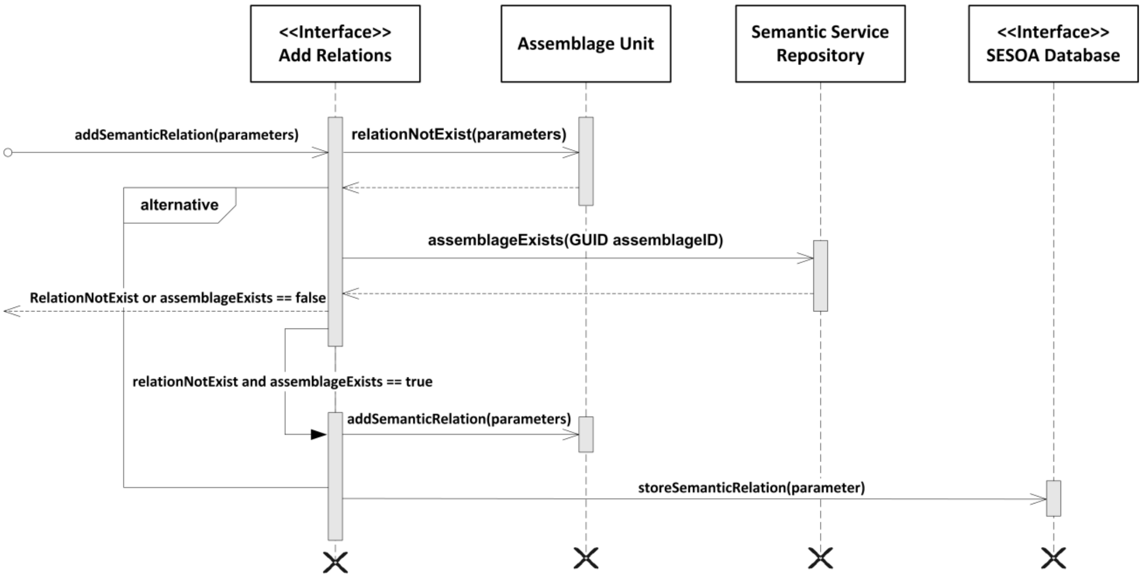

<<Interface>>
Add Relations
Assemblage Unit
Semantic Service Repository
<<Interface>>
SESOA Database
addSemanticRelation(parameters)
relationNotExist(parameters)
alternative
assemblageExists(GUID assemblageID)
RelationNotExist or assemblageExists == false
relationNotExist and assemblageExists == true
addSemanticRelation(parameters)
storeSemanticRelation(parameter)

**Fig. 6.18:** Adding Semantic Relation

Some RDF statements examples are listed as triples in Table 6.9[49] that reflects the semantic data model. Different assemblages like logistics, payment or shipping are semantically linked via the “hasMember” relationship with Web Services like for example “getOpenPurchases”, “sendInvoice”, and “createShipment” respectively. Service providers firstly registers these services within the aforementioned assemblages and the automatic semantic relations are then generated as listed in the below table.

**Tab. 6.9:** Examples of Data Model Triples

| Number | Subject | Predicate | Object |
|---|---|---|---|
| **1** | Logistics | hasMember | getOpenPurchases |
| **2** | Payment | hasMember | sendInvoice |
| **3** | Shipping | hasMember | createShipment |

The previous examples are excluded from the business case that is going to be explained more in details in the following section.

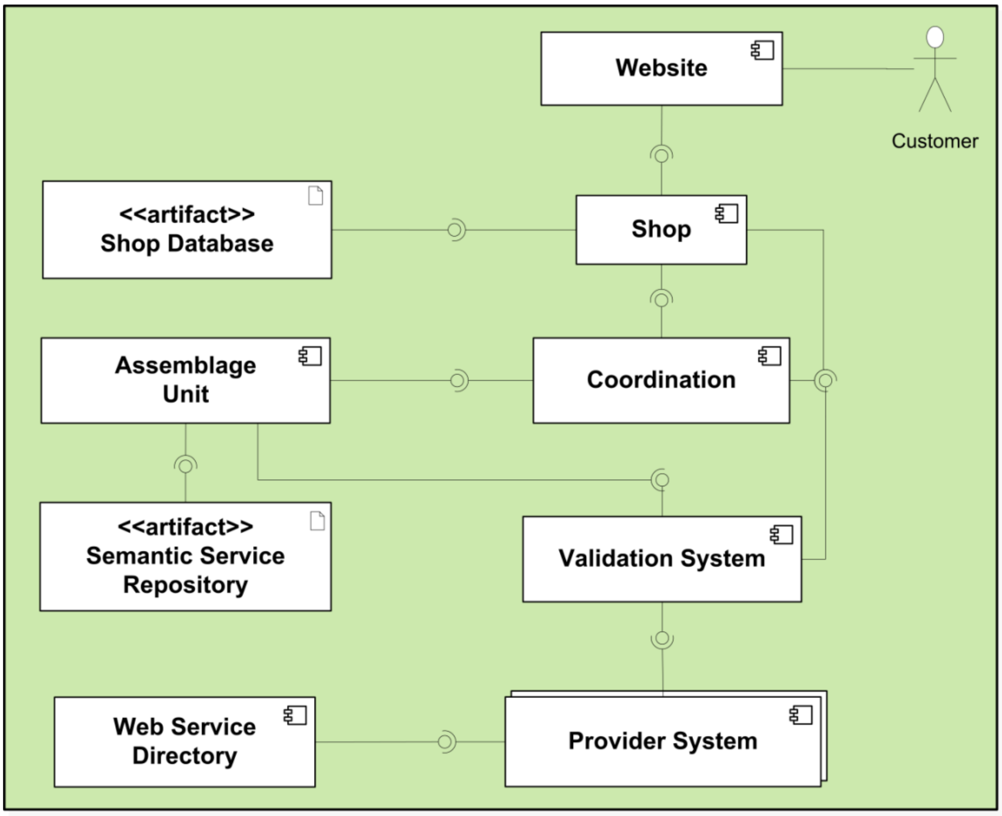
## 6.6 Business Case Architecture

Following the requirements defined in Section 5.3.3, the selling process (the business case used in this work) is divided into several components. This division helps in facili-

[49] In this work:

- All entity names of subjects are prefixed with: “http://asbl.wi-ol.de/sesoa/assemblage/”
- All entity names of predicates are prefixed with: “http://asbl.wi-ol.de/sesoa/”
- All entity name of objects are prefixed with: “http://asbl.wi-ol.de/sesoa/services/”

tating the overall implementation procedure by assigning individual tasks for each component following the main SESOA reference architecture.



Source: Figure 3 in **(Mahmoud, Petersen, et al., 2012, p. 189)**

**Fig. 6.19:** Selling Process Component Architecture

Figure 6.19 depicts the component architecture of the selling business process in compliance with SESOA reference architecture. The components of the selling business process are the website, shop, coordination, shop database and external Web Services supplied by the provider system. The Website component represents the front end of a fictional online shop that sells watches. It is responsible for the interaction with the end customer. All information displayed on the website (e.g. product information or customer information) are stored in a database, which is controlled by the shop component. This latter is fully realized as a set of Web Services which read or write data into the shop database. For example, if the front end wants to display product information or create a new customer, it calls one of the shop's Web Services to write the data into the database or to send the requested information back to the front end.

Upon creating a new order by the customer, the front end calls another Web Service that sends all information about this newly created order to the coordination component that uses a workflow engine to execute the selling business process. External payment and shipment services supplied by different service providers are used by the coordinator component to complete the selling business process.

All the services coming from external service providers have to comply with the SESOA requirements (see Section 5.3). Optional validation may be applied to these services before registering them within the SESOA's assemblage unit in one or more desired assemblages (like logistics, payment, shipment…). These services are then published in the semantic service repository to be discovered, selected and invoked by the selling business process or any other process. Insights to the implementation of this business process are presented in the next chapter.

## 6.7 Summary of System Outcomes

After explaining the main requirements of the SESOA reference architecture together with the internal architecture of its components, this section lists the main outcomes behind them both. The main added values can be summarized as follows:

*Lightweight Semantic-enabled Solution*: on the one hand, the problems of traditional SOA solutions can be enhanced by using the proposed reference architecture presented in this work. This is done by creating RDF statements of Web Services relations and enabling semantic query language to have better matchmaking of service requests. On the other hand, applying lightweight semantic annotation in this reference architecture reduces the overall complexity in the existing heavy-weighted semantic SOA frameworks.

*Process Orientation*: SESOA reference architecture is designed to provide a process-oriented solution based mainly on business processes and Web Services. This design aspect was taken into consideration because most enterprises nowadays are fully relying on several business processes. Implementing these processes using workflows and Web Services will ease the integration of different application to execute these processes side by side with enabling their reusability among departments or even enterprises.

*Reusability*: The proposed reference architecture resulted from this work offers the high ability of reusing the functionality since each component in this architecture can be considered as a standalone system and can be reused in other systems. This makes the proposed architecture generic enough to be applied in many business domains as a whole or enables some of its components to be reused in many applications. Some highlights to apply the SESOA reference architecture or some of its components to some business domains are presented in Chapter eight.

*Open the Market to SMEs*: Using this reference architecture is not restricted to the monopolized enterprises in service market. Each company regardless of its size can register its services in the system's assemblages to be offered to the customers. Furthermore, several business processes and scenarios can take benefit of this architecture to be realized in any company. The simplified marketing of the system's subcomponents helps in opening the market to include SMEs and improves the time-to-market factor.

*Registration of Web Services in Assemblages*: All Web Services supplied by different service providers are registered in the system’s assemblages in order to organize the semantic annotation of service relations. This provides another level of service classification together with the traditional one offered in the service directories. Both traditional requests and semantic-enabled requests can be initiated in the system. As a result, this enhances the overall Web Service searching mechanism and improves the response time of the discovery phase activities.

*Service Validation and Evaluation*: Other important outcomes from this reference architecture are the validation and evaluation of Web Services. Service validation is optional. It is offered to service providers to validate mainly the data types used to implement their services. Service evaluation is done automatically at the consumer side based on secure service ranking protocol. This protocol is machine-to-machine protocol and takes two evaluation criteria namely the response time and the availability of the corresponding invoked Web Service. To prevent service providers from manipulating the value of their services’ evaluations, all the values are controlled by a third party called EPA that calculates the evaluation values and updates them in the system’s semantic service repository.

*Advertisement of New Web Services*: In the case that a desired functionality requested by a service consumer is absent and not offered in the system, advertisements for a new Web Service that implements such functionality is forwarded via the SESOA validation system’s interface to the service providers. This enhances the point of opening the market to include SMEs or even individuals to make profit out of it.

To conclude, this chapter illustrated the main concepts of the SESOA reference architecture and gave an in-depth explanation for the internal architecture of its component. Web Service registration scenario with the semantic annotation of service relations were explicated. System interactions together with service validation and evaluation were presented as well. Finally, system outcomes were listed at the end of this chapter. Implementation aspects and highlights of the ideas and concepts presented in this chapter are presented in Chapter seven.

# 7 Prototypical Implementation and Evaluation

The fourth phase of the research methodology described in Chapter four is the demonstration phase. It demonstrates the characteristics of the system by producing -as a proof of concept- the prototype that demonstrates the SESOA reference architecture together with implementation of an accompanying business case. Based on the conceptual specifications of Chapters five and six, a concrete SESOA system is shaped and developed. This system contains a selected set of application elements and dynamic components (Web Services and business processes). Following the demonstration phase in the adopted research process, this chapter is devoted to describe the main characteristics of the final SESOA prototype together with the description of the accompanying business case (the selling business process) that have been implemented on top of it. Moreover, the evaluation aspects of the prototypical implementations in different business domains are going to be listed here as well.

This chapter starts with an overview of the selected techniques used to produce the prototype together with the architecture of the prototype itself. Based on that, a short explanation the technologies adopted in this work is provided. The following sections in this chapter are then dedicated for the SESOA prototype and its accompanying business case respectively. Highlights to the implementation of the validation services are then explained. The chapter then shows the possible directions to evaluate this work in different business domains. Finally, this chapter summarizes the main implementation issues explained in its sections.

## 7.1 General Overview of the Prototype Architecture

This section shows what techniques have been used to implement SESOA prototype. Insights to the prototype are then described to show how the internal components of the SESOA reference architecture have been realized. Moreover, some implementations highlights are going to be given then to the selling business process that had been described in the previous two chapters.

### 7.1.1 Choice of the Adopted Technologies

The choice about which specific technologies and products to use in developing the system prototype depended actually on different factors like compatibility, performance, licensing, availability, etc. Since the resulted prototype has to be used by different consumers that have disparate operating systems and platforms, it has to be rich enough to support all types of consumers.

From compatibility point of view, most of the enterprises nowadays are using Microsoft products. That pushes the decision for this prototype version to be implemented using Microsoft products as well. However, the resulted prototype is actually a set of Web-based applications that can be accessed using different Web browsers like Microsoft Internet Explorer, Mozilla Firefox, Opera, etc. This makes the underlying implementations of the system just details differs from one enterprise to another. The same prototype that is going to be described here can be implemented differently using any other software products or vendors. This is because it is totally based on the Web Services technology and its underlying semantic annotations.

The main base for achieving the agile implementation and development of the SESOA prototype is the distribution of its functionalities using lightweight semantic annotated Web Services. Microsoft Visual Studio 2010 has been used as the main IDE[50] with its .NET framework to implement the final prototype. Using the technologies provided by them both, most of the system's functionalities have been realized. From a compatibility perspective, Microsoft SQL Server 2008 R2 has been used as relational database server and Microsoft IIS[51] as a Web server. Furthermore, SemWeb.NET (Tauberer, 2010) has been used as RDF library supporting C#.NET to provide the semantic annotation in form of RDF statements to the system's Web Services. Moreover, W3C validation service[52] has been used to validate the system's RDF semantic annotations. Last but not least, Twinkle[53] has been used as to load, edit and save SPARQL queries.

Table 7.1 lists all the technologies that have been utilized in the proof of concept phase to implement the system's prototype with short description for the role of each of them.

**Tab. 7.1:** System's Enabling Technologies

| Technology | Version | Role |
|---|---|---|
| **Microsoft Visual Studio 2010 Ultimate** | 10.0.40219.1 - SP1Rel | It is the IDE used to realize the SESOA reference architecture. |
| **.NET Framework** | 4.0.30319.1 | It provides extensive libraries to bridge the interoperability gab among applications. |
| **Microsoft IIS** | 7.5.7600.16385 | It is the Web server that has many extension modules that serve to share the information among the system's users in any distributed network. |
| **Microsoft Visual C# 2010** | 4.0 | Visual C# has been used as the main programming language to support object and component orientation. |
| **Microsoft Visual** | 10.0.30319.1 | Visual Basic.NET has been used in the |

[50] IDE is an acronym for Integrated Development Environment.

[51] Microsoft IIS is an acronym for Internet Information Services and it was formerly known as the Internet Information Server.

[52] The W3C validation service is based on Another RDF Parser (ARP) that is part of the Jena toolkit (Carroll et al., 2004). It is embedded within the resulting SESOA prototype and is available online at: http://www.w3.org/RDF/Validator/.

[53] Twinkle (http://www.ldodds.com/projects/twinkle/) is a SPARQL query tool.

| | | |
|---|---|---|
| **Basic .NET** | | prototype to support the Web programming. |
| **ASP.NET** | 4.0.30319.1 (4.0) | ASP.NET is used to build the main Web applications, Web sites and some of the system's Web Services. |
| **Microsoft WCF**[54] | 4.0 | WCF is used in this prototype to design and deploy the system's distributed applications following SOA paradigm. |
| **Microsoft WF**[55] | 4.0 | WF has been adopted in the system as the main in-process workflow engine to realize the long-running business processes as workflows. |
| **Microsoft SQL Server 2008 R2** | 10.50.2500.0 - SP 1 | Microsoft SQL Server has been used as a relational database server to store and retrieve the system's data. |
| **SemWeb.NET** | 1.0.7 | SemWeb.Net C# library has been used to facilitate the automatic semantic annotation and storage of Web Services' relations. |
| **Twinkle** | 2.0 | Twinkle has been used as a SPARQL query tool to query the system's RDF documents and data. |
| **W3C RDF Validation Service** | 2-alpha-1 | W3C RDF validation service is embedded within the system to validate the resulting RDF annotations. |

As mentioned above and as a proof of concept, several applications have been implemented to deliver the prototype. These include the SESOA main Web application that implements all the system's functionalities and the accompanying business case Web application that uses these functionalities. The resulted prototype, developed in Microsoft Visual Studio 2010 Ultimate Edition (C# 4.0 and Visual Basic.NET as programming languages and ASP.NET 4.0 as Web application framework), accesses the databases (Microsoft SQL Server 2008 R2) to retrieve and store the information of the system's assemblages, Web Services and the accompanying business case data. On the one hand, the system's main Web application facilitates the process of creating WS assemblages and Web Services, registering external and internal Web Services within the assemblages, annotating the assemblages-Web Services relations, besides service validation. On the other hand, the accompanying business case (developed using ASP.NET 4.0 as well) is built on top of the main Web application to represent the selling business

[54] WCF is an acronym for Windows Communication Foundation which was known previously as Indigo.

[55] WF is an abbreviation for Windows Workflow foundation.

process from creating orders by customers till the end of product delivery. To realize the activities within this business process, several workflows have been designed and implemented (using WF 4.0) and several Web Services have been used to "wrap" these workflows' activities. WSDL description language has been used to describe the system's Web Services. All the Web Services, developed using ASP.NET 4.0 and WCF 4.0, are published in the system's semantic service repository. Querying these services based on their RDF semantic descriptions has been achieved using Twinkle 2.0. Semantic annotation of these services with predefined WS-assemblages is automatically done with the help of the SemWeb library. Table 7.2 reflects the already explained technologies (listed in Table 7.1) on the SESOA reference architecture's components (see Chapter six, Section 6.1.3).

**Tab. 7.2:** List of applied Technologies to SESOA components

| Component Name | Applied Technology |
|---|---|
| **Front End** | ✓ ASP.NET 4.0<br>✓ Visual Basic .NET<br>✓ WCF 4.0<br>✓ Twinkle 2.0 |
| **Processing System** | ✓ WF 4.0<br>✓ C# 4.0 |
| **Database System** | ✓ MSSQL Server 2008 R2 |
| **Semantic Web Service-based System** | ✓ SemWeb 1.0.7<br>✓ WCF 4.0 |
| **Semantic Service Repository** | ✓ LINQ to SQL[56]<br>✓ SemWeb 1.0.7 |
| **Validation System** | ✓ WCF 4.0<br>✓ W3C RDF Validation Service<br>✓ ASP.NET |

To have a kind of big picture behind these enabling technologies, Figure 7.1 depicts the prototype architecture and shows how the technologies were applied to each individual component of the SESOA reference architecture. To implement the processing system component, WF 4.0 and C# 4.0 as underlying programming language have been used. The accompanying selling business process has been realized as a set of workflows to be processed by WF 4.0. Several activities within the resulted workflows have been

[56] Language Integrated Query (LINQ) to SQL also known as DLINK is a provider converts normal LINQ query into SQL query to be sent to Microsoft SQL Server to be processed.

implemented as custom code activities[57] using C#. The system's front end is built using ASP.NET 4.0 and C# 4.0/Visual Basic.NET as underlying programming languages.

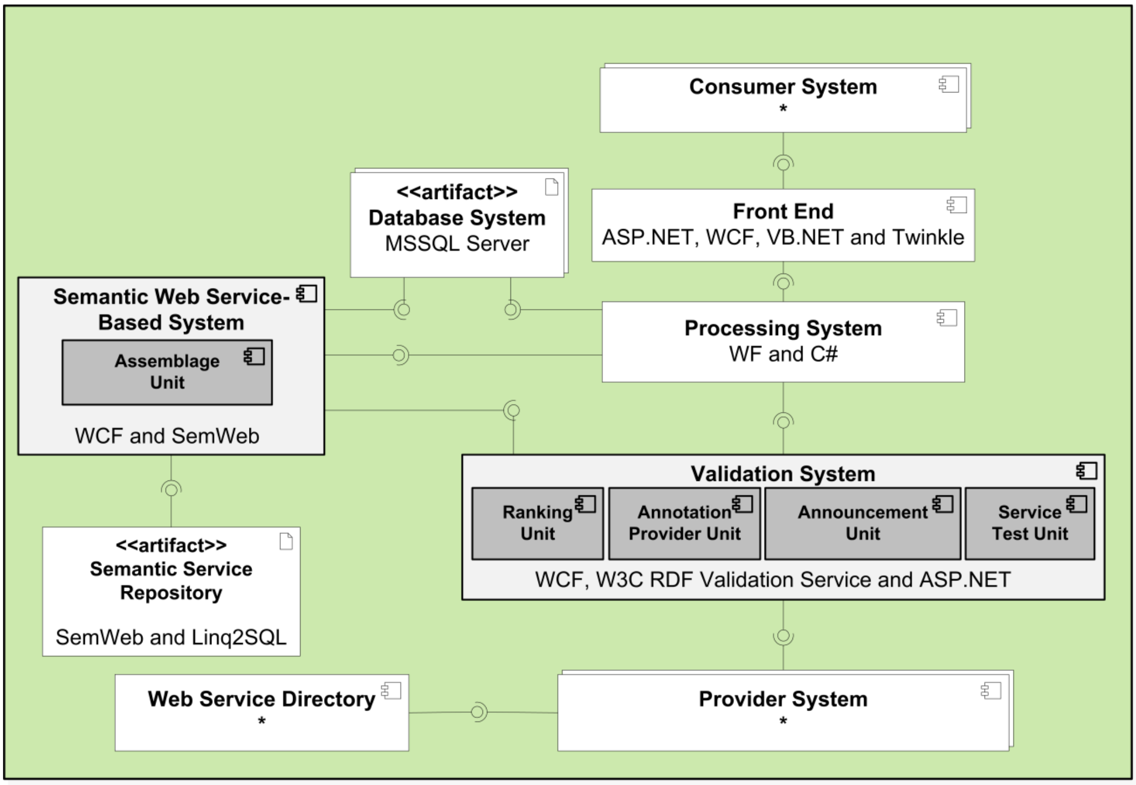


**Fig. 7.1:** The Prototype Architecture

The interface between the processing system and the semantic Web Service-based system component to retrieve the semantic annotation of the system's Web Services is realized using Twinkle 2.0. Assemblage and Web Service management and registration in the assemblage unit within the semantic Web Service-based system component are done using the SemWeb library and WCF 4.0. Some of the Web Services are implemented using the conventional ASP.NET Web Services (ASMX) for compatibility reasons.

ASP.NET 4.0, WCF Web Services, and W3C RDF Validation Service are the used technologies to realize the validation system component with its four subcomponents. SemWeb library and LINQ to SQL technologies are used to realize the semantic service repository component. Microsoft SQL Server 2008 R2 has been used as a database engine to realize the database system.

Since consumer system, provider system and Web Service directory components are external components, it is indifferent which technology is used to realize them. What really matters is that these components can access both the SESOA main and accompanying business case Web applications. Finally, all of these technologies can be substi-

[57] Custom activities normally are not included in the main functions provided inWF4.0. Rather they handle and process application-specific data.

tuted by other ones (like Java-based technologies) because big portions of the prototypical implementation are Web Service (in other words XML-based).

The following sections give more details and insights about the realization of each component supported by screenshots, code snippets and modeling diagrams.

### 7.1.2 The Selling Process Prototypical Considerations

The selling business process that has been built on top of the SESOA reference architecture has been divided into several components (see Figure 7.2). The separation of the selling process into individual components eases the management tasks and inevitably supports more agile implementation. The resulted Web application represents a fictitious online shop that sells watches. The application's has the following components: website, shop, logistics, payment, shipping, mail, and coordination.

The website component represents the online shop Web application for the selling business process. It is responsible for the interaction with the system's end users (the customers). All information that is displayed on the website (e.g. product or customer information) is stored in the selling process's database that is part of the SESOA database system. This database is controlled by the shop component.

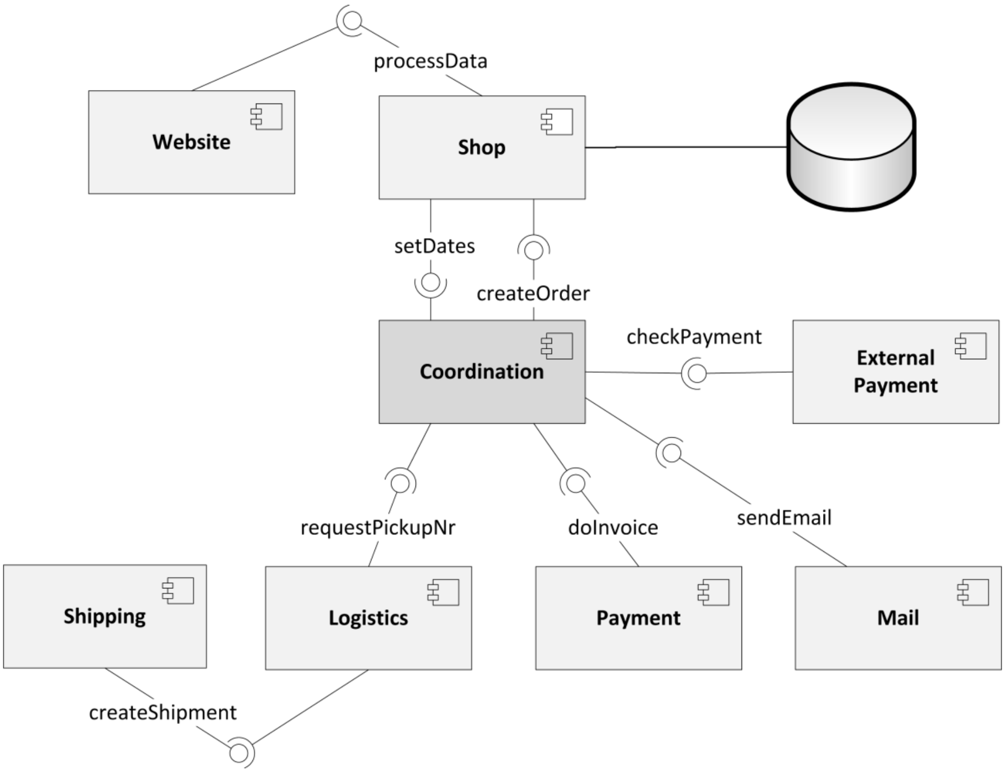


**Fig. 7.2:** The Components of the Selling Business Process

The shop component offers several Web Services, which read or write data from the selling process's database. For example, when the website displays the product information or creates a new customer, the website calls a Web Service implemented by the shop component and this Web Service writes the data into the selling process's database or sends the requested information back to the website.

Upon creation new orders by a customer on the website, it calls a specific Web Service from the shop component and this service sends all information about the order to the coordination component. The coordination component is responsible for processing the customers' orders. Depending on the chosen payment method (implemented at the payment component), the coordination component first calls the payment component and afterwards the logistic components to check the product availability in the warehouse or vice versa.

The payment component represents an external payment company that checks the customer's payment data. It creates an invoice for a given order and sends it back to coordination component to continue processing customer's order. The coordination component then calls the mail component to send the already created invoice to the customer. The mail component represents the customer notification system.

The logistics component represents the warehouse of the fictitious company behind this business process. As soon as the order is ready for shipping, it calls the shipping component that is responsible for the order shipment. Once an order has been shipped, using an external shipping company via the shipping component, the coordination component sends another email via the mail component to notify the customer.

The realization of this business case is based on the same technologies mentioned before in Table 7.1. One primary requirement behind this business case was to use technologies under the umbrella of Microsoft Visual Studio 2010 to be compatible with SESOA prototype. Visual Studio includes several possibilities to realize this business case and its subsequent workflow applications.

### 7.1.3 System Configurations

The Internet Information Services provided with Microsoft Windows Server 2008 R2 are needed to host all developed applications (the main SESOA and the accompanying business case Web applications). Every application has its own directory. The overall prototype is hosted on a single Windows Server computer with only one IP address. This leads to the decision of using different ports for every application. Table 7.3 shows the distribution of all components sites (applications) to their specific ports on the Web server.

**Tab. 7.3:** Distribution of System's Sites on Ports

| Site Name | Port Number | Description |
|---|---|---|
| **SESOA** | 1234 | This site hosts the main SESOA Web Application |
| **Validation** | 8099 | This site hosts the system's different validation Web Services |
| **Website** | 8088 | This site hosts the main business case's website |
| **Item Images** | 8086 | This sites is used just to host the products images that are used by the business case's website |
| **Shop** | 8080 | This site hosts the business case's application that manages the database interactions |
| **Logistics** | 8082 | This site hosts the main logistics application that deals with the business case's warehouse |
| **Payment** | 8084 | This site hosts the business case's application that deals with the company's internal payment application |
| **External Payment** | 8085 | This site hosts Web Services that represent an external payment application |
| **Shipping** | 8081 | This site hosts Web Services that represent an external shipping application |
| **Mail** | 8087 | This site hosts the business case's application that deals with customer notification |
| **Coordination** | 8090 | This site hosts the business case's application that manages all of its applications |

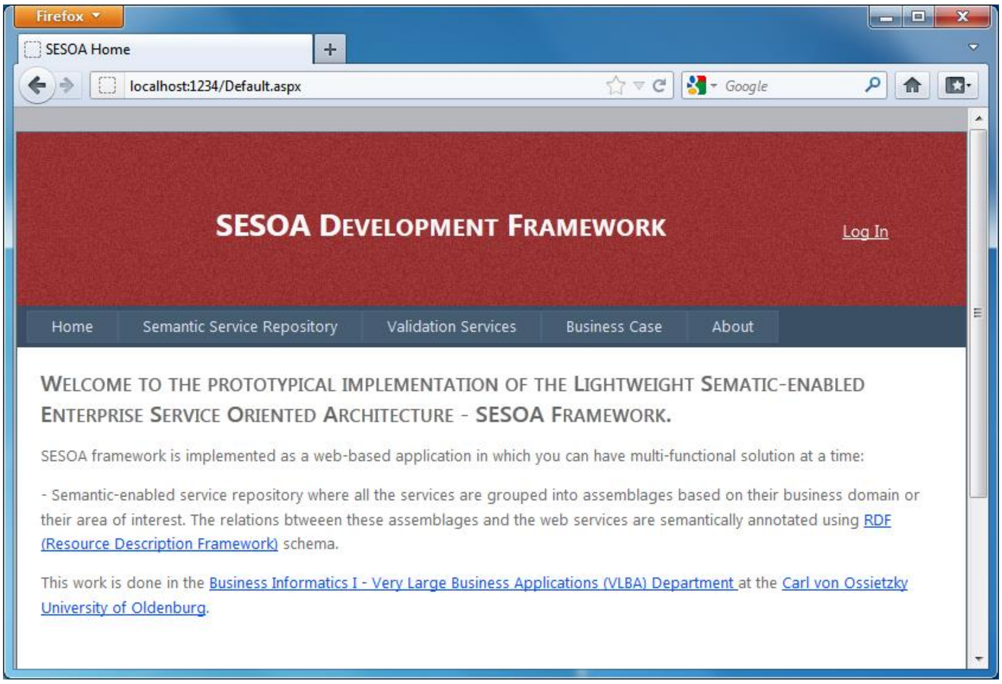
In the following sections, all implementation details of the SESOA main Web application and the business case applications are detailed using bunch of screenshots, listings, and UML diagrams.

## 7.2 SESOA Implementation

SESOA is built with a holistic perception of enterprise Web Services as part of workflows that represent different enterprise's business processes. It aims at providing a semantic-enabled Web Service middleware that allows different business stakeholders to access bunch of business services via uniform, rich, and flexible interfaces. The main design and development process of SESOA implementation is directed by four key guidelines: the realization of enterprise's business processes to be managed by an in-process workflow engine, the use of Web Service standards (e.g., WSDL, SOAP), the semantic enrichment of these services (using RDF statements), and extensibility (i.e., the ability to add new functionalities). The main focus in SESOA prototype is on implementing the automatic semantic annotation techniques proposed in this dissertation.

### 7.2.1 SESOA Web Application

All of the functionalities listed at the phase of requirement definitions (see Chapter 5) are implemented in the SESOA Web application. The main GUI is shown in Figure 7.3.



SESOA DEVELOPMENT FRAMEWORK

WELCOME TO THE PROTOTYPICAL IMPLEMENTATION OF THE LIGHTWEIGHT SEMATIC-ENABLED ENTERPRISE SERVICE ORIENTED ARCHITECTURE - SESOA FRAMEWORK.

SESOA framework is implemented as a web-based application in which you can have multi-functional solution at a time:

- Semantic-enabled service repository where all the services are grouped into assemblages based on their business domain or their area of interest. The relations btweeen these assemblages and the web services are semantically annotated using RDF (Resource Description Framework) schema.

This work is done in the Business Informatics I - Very Large Business Applications (VLBA) Department at the Carl von Ossietzky University of Oldenburg.

**Fig. 7.3:** The SESOA Development Framework GUI

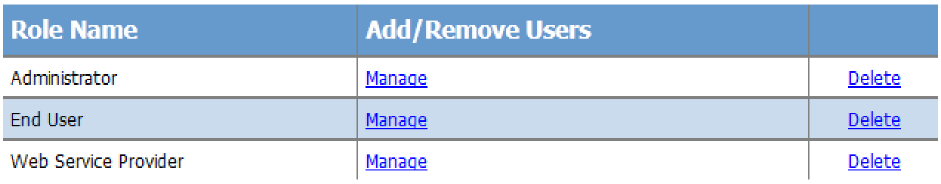
This application has been implemented as a Web application using ASP.NET 4.0 and all its functions have been realized using C# 4.0, Visual Basic.NET, WCF 4.0, WF 4.0, and SemWeb1.0.7. Microsoft SQL Server 2008 R2 has been selected to realize the SESOA databases. A list of the menu items with their webpages that host the system functionalities delivered by this application is detailed in Table 7.4.

**Tab. 7.4:** Items of the SESOA Development Framework GUI

| Root Item | Child Item | Description |
|---|---|---|
| **Home** | - | The SESOA development framework welcome page |
| **Semantic Service Repository** | Discover Repository | Show all assemblages and their services in the service repository |
| | Discover Assemblages | Show the available assemblages in the repository |
| | Discover Services | Show only the Web Services in the repository |
| | Add Assemblage | Add a new assemblage to the repository |

| | | |
|---|---|---|
| | Add Service Information | Add a new service as a member of one of the assemblages in the repository |
| | Show Assemblage Service Relations | Show RDF annotation between a selected Web Service and the assemblage to which it belongs |
| | Show All Assemblage Service Relations | Show all RDF relations between repository's assemblages and Web Services |
| | Delete Assemblage | Delete permanently an assemblage with its services from the repository |
| | Delete Service | Delete a service from the repository |
| **Validation Services** | RDF Validation | To validate the system's RDF relations |
| | Primitive Data type Validation | Apply validation on a primitive data type level |
| | Complex Validation | Apply validation on a complex data type |
| **Business Case** | - | The main site of the accompanying business case |
| **About** | - | About information |

These items are made available to the system users based on their rights. As mentioned before, there are three main types of users in the system: the administrator, the Web Service provider, and the end user. To realize that, ASP.NET Web Site Administration Tool has been used to create three roles as shown in Figure 7.4 in which users can be added or removed to permit or deny access on the items listed in Table 7.4.



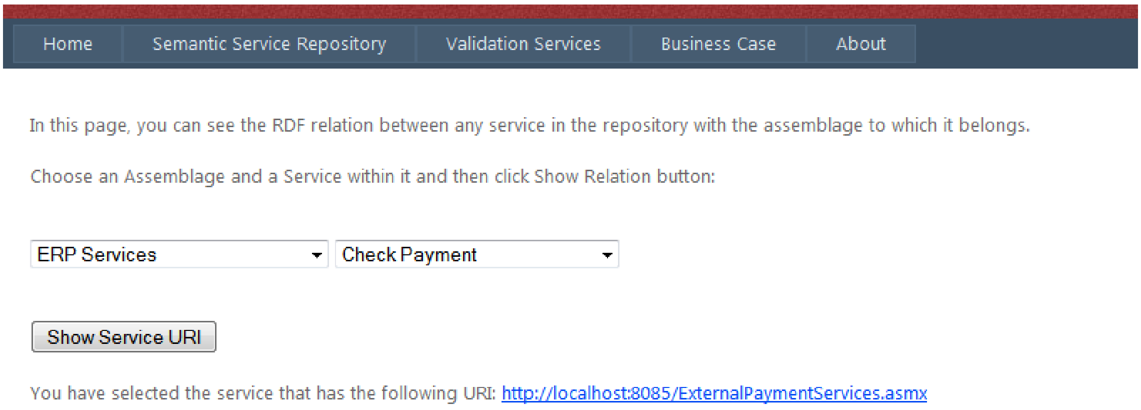
| Role Name | Add/Remove Users | |
|---|---|---|
| Administrator | Manage | Delete |
| End User | Manage | Delete |
| Web Service Provider | Manage | Delete |

**Fig. 7.4:** System's Main Roles

System administrators have full access to all items listed in Table 7.4. Web Service providers don't have access to the following items: add assemblage, delete assemblage, and delete service. As for the end users, they can only see the home, business case, and about items. By default, any created user is placed in the end user role. Only system administrator can assign the Web Service provider role to new users following specific criteria. The following sections explain each item individually.

### 7.2.1.1 Discover Repository

This item shows all the available assemblages in the service repository. Upon choosing an assemblage, all its members (i.e. the actual Web Services) can be seen. When selecting a Web Service, a button called “Show Service URI” can be clicked to show the address of this service. This is shown in Figure 7.5.



In this page, you can see the RDF relation between any service in the repository with the assemblage to which it belongs.

Choose an Assemblage and a Service within it and then click Show Relation button:

ERP Services

Check Payment

You have selected the service that has the following URI: http://localhost:8085/ExternalPaymentServices.asmx

**Fig. 7.5:** Discover Repository

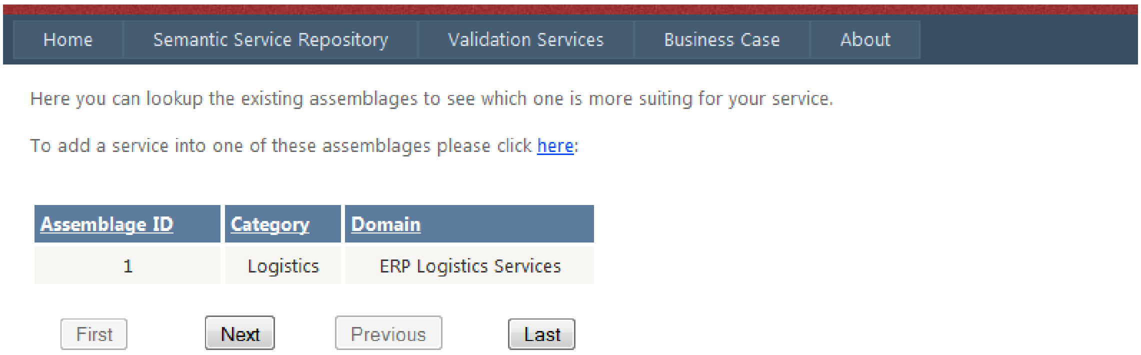
An example for that is as shown above, the “ERP Services” assemblage is selected from a dropdown list. All the registered services within this assemblage can be seen at the right dropdown list. In this example, the “Check Payment” service had been selected. Clicking on the “Show Service URI” is showing the address of the selected service and in this example it is: “http://localhost:8085/ExternalPaymentServices.asmx”.

### 7.2.1.2 Discover Assemblages

This semantic service repository’s functionality is implemented to list all the assemblages within the repository.

As shown in Figure 7.6, service providers can see list of assemblages available in the repository including “ID”, “Category”, and “Domain” attributes.

A navigation toolbar is also provided to browse all the available assemblages. Sorting is enabled as well and it can be applied on all of the assemblage’s attributes.



Here you can lookup the existing assemblages to see which one is more suiting for your service.

To add a service into one of these assemblages please click here:

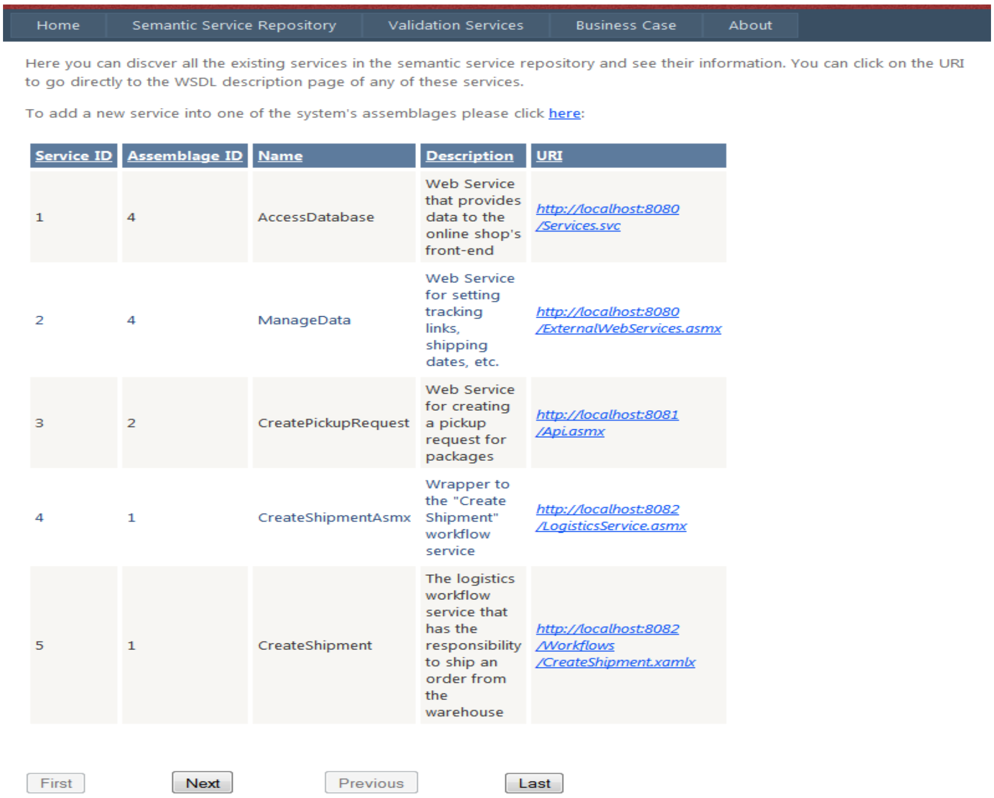
| Assemblage ID | Category | Domain |
|---|---|---|
| 1 | Logistics | ERP Logistics Services |

**Fig. 7.6:** Discover Repository’s Assemblages

The site of discovering assemblages is made available to the users who have the Web Service provider role in order to choose which assemblage suits at most their Web Services. System administrator is the only user who can add assemblages. In the case that service providers don’t find any assemblage suits their services, they can contact the system administrator to create it.

#### 7.2.1.3 Discover Services

This site provides similar functionality like the one provided in the previous section. What is retrieved in this site (webpage) are the actual Web Services that are registered in the assemblages and published in the semantic service repository.



Here you can discver all the existing services in the semantic service repository and see their information. You can click on the URI to go directly to the WSDL description page of any of these services.

To add a new service into one of the system's assemblages please click here:

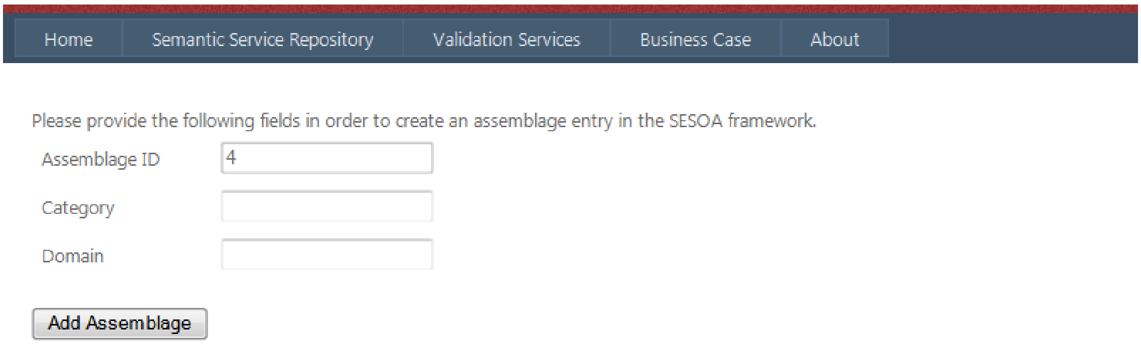
| Service ID | Assemblage ID | Name | Description | URI |
|---|---|---|---|---|
| 1 | 4 | AccessDatabase | Web Service that provides data to the online shop's front-end | http://localhost:8080/Services.svc |
| 2 | 4 | ManageData | Web Service for setting tracking links, shipping dates, etc. | http://localhost:8080/ExternalWebServices.asmx |
| 3 | 2 | CreatePickupRequest | Web Service for creating a pickup request for packages | http://localhost:8081/Api.asmx |
| 4 | 1 | CreateShipmentAsmx | Wrapper to the "Create Shipment" workflow service | http://localhost:8082/LogisticsService.asmx |
| 5 | 1 | CreateShipment | The logistics workflow service that has the responsibility to ship an order from the warehouse | http://localhost:8082/Workflows/CreateShipment.xamlx |

**Fig. 7.7:** Discover Repository’s Registered Web Services

As shown in Figure 7.7, “Service ID”, “Assemblage ID”, “Name”, “Description”, and “Service URI” are the retrieved information. A Web Service can be a member of one or more assemblages and this is done at the process of adding a service to the system (see following section). Navigation toolbar is also provided in this site to browse all available and registered Web Services. Sorting the system’s services is enabled as well and it can be applied to all of the service’s attributes.

### 7.2.1.4 Add an Assemblage

In this site, new assemblages can be added. The information required for adding a new assemblage includes: “Assemblage ID”, “Category”, and “Domain”. Figure 7.8 shows how to add a new assemblage.



Please provide the following fields in order to create an assemblage entry in the SESOA framework.

Assemblage ID: 4

Category:

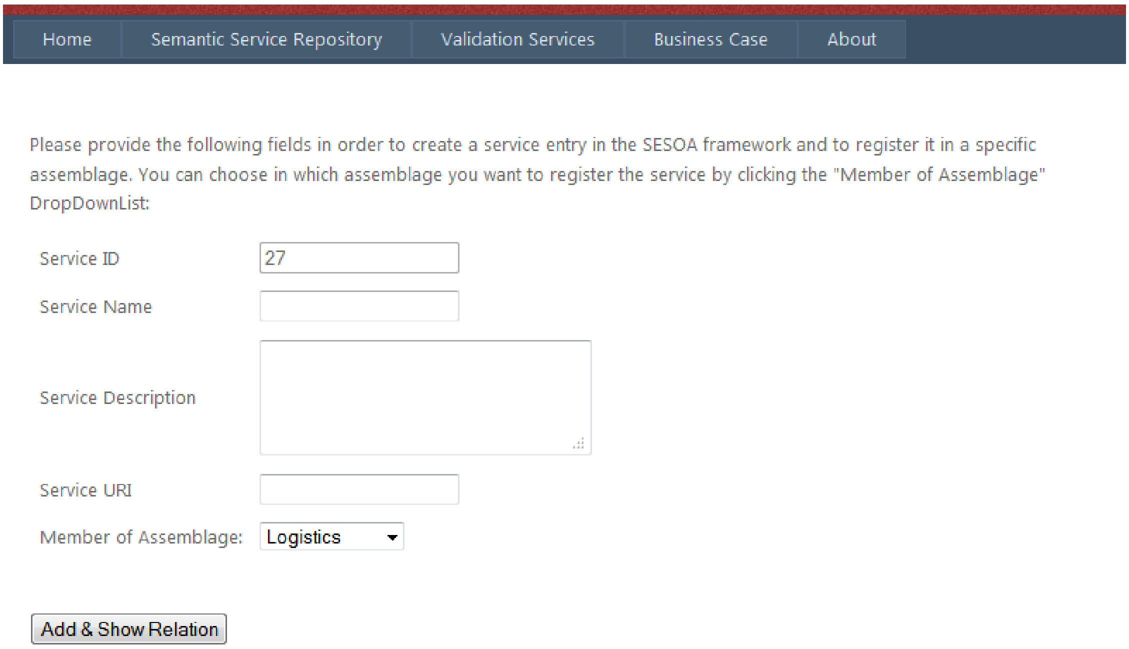
Domain:

**Fig. 7.8:** Add a New Assemblage

One issue has to be taken into consideration while adding new assemblages is that only the system administrator can add new assemblages into the system. If a service provider doesn’t find an assemblage suiting its needs, it can contact the system administrator to add new assemblage meeting its requirements.

### 7.2.1.5 Add Service Information

In this site, new Web Services can be added to the system. This means that the services are registered in the system’s assemblages and advertised in the semantic service repository. The information required for adding a new Web Service includes: “Service ID”, “Service Name”, “Service Description”, “Service URI”, and “Member of Assemblage”. The last attribute indicate the assemblage in which the service is member of, i.e. registered. To register a service with more than an assemblage, the adding process has to be repeated. Figure 7.9 shows the page in which new Web Services can be added to the system. System administrators and Web Service providers are the users who can add new Web Services to the system.



Please provide the following fields in order to create a service entry in the SESOA framework and to register it in a specific assemblage. You can choose in which assemblage you want to register the service by clicking the "Member of Assemblage" DropDownList:

Service ID: 27

Service Name:

Service Description:

Service URI:

Member of Assemblage: Logistics

**Fig. 7.9:** Add a New Web Service

Upon clicking the "Add & Show Relation" button, the Web Service is registered in the chosen assemblage, an RDF statement between the service and the assemblage is created, the service is stored in the RDF database as an entity with ID and value and these two attributes are linked with their counterparts in the assemblage, and finally this relation is published at the semantic service repository. Listing 7.1 shows this automatic created relation between the newly added service and the assemblage in which this service is registered. This relation can be shown at any time upon request.

**Listing 7.1:** Assemblage - Service Semantic-annotated Relation

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:assemblage="http://asbl.wi-ol.de/sesoa"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:sesoa="http://asbl.wi-ol.de/sesoa/">
    <rdf:Description rdf:about="http://asbl.wi-
     ol.de/sesoa/assemblage/Shipping">
        <sesoa:hasMember rdf:resource="http://asbl.wi
         ol.de/sesoa/services/GetShippingType" />
    </rdf:Description>
</rdf:RDF>
```

The above listing shows an example of a new service "GetShippingType" (object) that is related with the assemblage "Shipping" (subject) with the "hasMember" relation (predicate). This subject-object-predicate RDF statement represents the semantic-annotated relationship between the assemblage and its member, i.e. the actual Web Service. Table 7.5 lists the triple of this RDF statement.

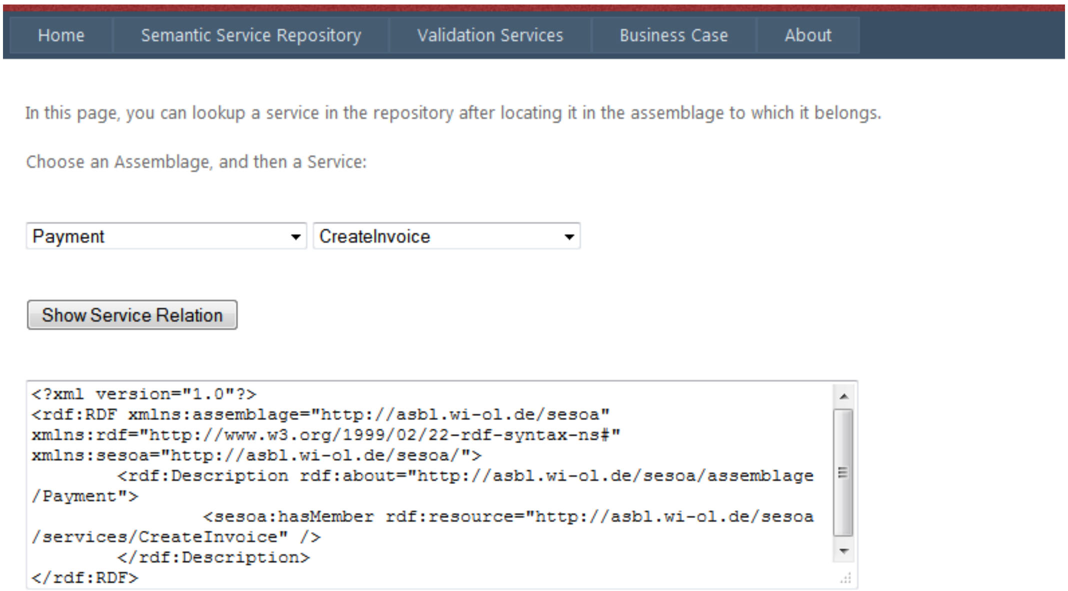
**Tab. 7.5:** RDF Statement Triple Example

| **Entity** | **Value** |
|---|---|
| **Subject** | http://asbl.wi-ol.de/sesoa/assemblage/Shipping |
| **Predicate** | http://asbl.wi-ol.de/sesoa/hasMember |
| **Object** | http://asbl.wi-ol.de/sesoa/services/GetShippingType |

As mentioned before in last chapter, all entity names of the subjects within the system's RDF statements are prefixed with: "http://asbl.wi-ol.de/sesoa/assemblage/", all entity names of predicates are prefixed with: "http://asbl.wi-ol.de/sesoa/" and all entity names of the objects are prefixed with: "http://asbl.wi-ol.de/sesoa/services/".

### 7.2.1.6 Show Assemblage Service Relation

This site enables the process of displaying the semantic-annotated relation of an assemblage with one of its members (a Web Service). As shown in Figure 7.10, a service can be chosen in an assemblage to show the semantic relation among them.



In this page, you can lookup a service in the repository after locating it in the assemblage to which it belongs.

Choose an Assemblage, and then a Service:

Payment ▾ CreateInvoice ▾

```
<?xml version="1.0"?>
<rdf:RDF xmlns:assemblage="http://asbl.wi-ol.de/sesoa"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:sesoa="http://asbl.wi-ol.de/sesoa/">
        <rdf:Description rdf:about="http://asbl.wi-ol.de/sesoa/assemblage
/Payment">
                <sesoa:hasMember rdf:resource="http://asbl.wi-ol.de/sesoa
/services/CreateInvoice" />
        </rdf:Description>
</rdf:RDF>
```

**Fig. 7.10:** Single Assemblage - Web Service Relation

After choosing the assemblage from the left dropdown list, all the services registered within this assemblage can be located and only one can be chosen from the right dropdown list. Upon choosing an assemblage and a registered service within it and clicking on the "Show Service Relation" button, the RDF statement that links between the assemblage and service entities is displayed. In the example shown in Figure 7.10, the assemblage "Payment" is having the service "CreateInvoice" as a member.

#### 7.2.1.7 Show All Assemblages Services Relations

This site enables displaying the relations among all system's assemblages with their members (the services). SESOA prototype offers two views for showing the semantic relations: the simplified and the full views. Listing 7.2 shows a subset of the simplified view of the overall system's RDF annotations. This view represents merely the traditional subject-predicate-object RDF statement. The example depicted in the listing below shows that an assemblage called "Logistics" (subject) has eight registered services (objects) as members (predicates) within it.

**Listing 7.2:** Subset of the Simplified RDF View

```
<?xml version="1.0"?>
<rdf:Description rdf:about="http://asbl.wi-ol.de/sesoa/assemblage/Logistics">
      <assemblage:hasMember rdf:resource="http://asbl.wi-Ol.de/sesoa/services
       /CreateShipment">
      </assemblage:hasMember>
      <assemblage:hasMember rdf:resource="http://asbl.wi-ol.de/sesoa/services
       /CreateShipmentAsmx">
      </assemblage:hasMember>
      <assemblage:hasMember rdf:resource="http://asbl.wi-ol.de/sesoa/services
       /GetDateAvailability">
      </assemblage:hasMember>
      <assemblage:hasMember rdf:resource="http://asbl.wi-ol.de/sesoa/services
       /GetItemAmount">
      </assemblage:hasMember>
      <assemblage:hasMember rdf:resource="http://asbl.wi-ol.de/sesoa/services
       /GetItems">
      </assemblage:hasMember>
      <assemblage:hasMember rdf:resource="http://asbl.wi-ol.de/sesoa/services
       /GetLongTermAmount">
      </assemblage:hasMember>
      <assemblage:hasMember rdf:resource="http://asbl.wi-ol.de/sesoa/services
       /GetOpenPurchases">
      </assemblage:hasMember>
      <assemblage:hasMember rdf:resource="http://asbl.wi-ol.de/sesoa/services
       /PurchaseItems"/>
      </rdf:Description>
```

While the simplified view shows neither the other attributes of the assemblages nor the other attributes of the services, the full view shows all attributes. Listing 7.3 shows a subset of the full view of the overall system's RDF annotations. It shows the same example depicted before in Listing 7.2. However, the example below shows all the attributes for both the "logistics" assemblage and its services. The assemblage in the below example has the ID "1", the category "Logistics", and the domain "ERP Logistics Services". It has eight services registered within it.

**Listing 7.3:** Subset of the Full RDF View

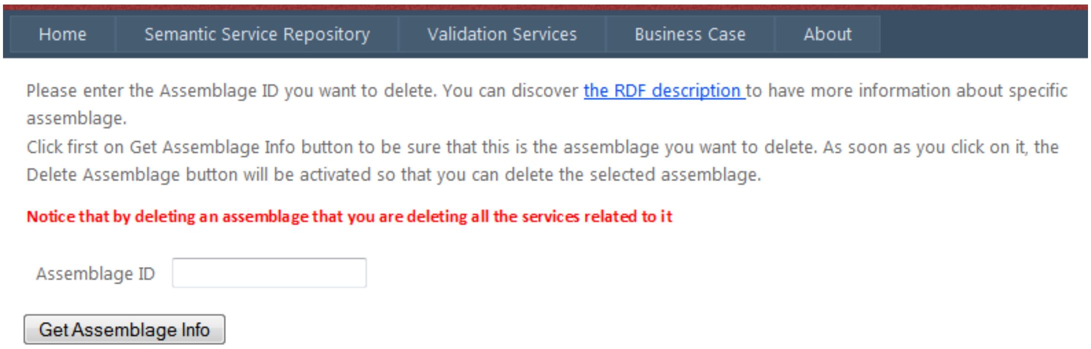
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:assemblage="http://sesoa.wi-ol.de/" xmlns:service="http://sesoa.wi-
ol.de/services/" xmlns:sesoa="http://asbl.wi-ol.de/sesoa/"
xmlns:services="http://asbl.wi-ol.de/sesoa/services/">
    <rdf:Description rdf:about="http://asbl.wi-ol.de/sesoa/assemblage/1">
        <sesoa:hasCategory>Logistics</sesoa:hasCategory>
        <sesoa:hasDomain>ERP Logistics Services</sesoa:hasDomain>
        <sesoa:hasMember>
            <rdf:Description rdf:about="http://asbl.wi-ol.de/sesoa/services
             /CreateShipmentAsmx">
                <services:hasID>5</services:hasID>
                <services:hasName>CreateShipment</services:hasName>
                <services:hasDescription> The logistics workflow service that
                 has the responsibility to ship an order from the warehouse
                </services:hasDescription>
                <services:hasURI>
                 http://localhost:8082/Workflows/CreateShipment.xamlx
                </services:hasURI>
            </rdf:Description>
        </sesoa:hasMember>

        ...

        <sesoa:hasMember>
            <rdf:Description rdf:about="http://asbl.wi-ol.de/sesoa/services
             /PurchaseItems">
                <services:hasID>11</services:hasID>
                <services:hasName>PurchaseItems</services:hasName>
                <services:hasDescription>Workflow service for placing and
                 processing a company order to refill the stock
                </services:hasDescription>
                <services:hasURI>
                 http://localhost:8082/Workflows/PurchaseItems.xamlx
                </services:hasURI>
            </rdf:Description>
        </sesoa:hasMember>
    </rdf:Description>
```

In the listing above, all the attributes for the eight registered services are shown. For example the first service shown in the above listing has the ID “5”, the name “CreateShipment”, the description “The logistics workflow service that has the responsibility to ship an order from the warehouse”, and finally it has the URI “http://localhost:8082/Workflows/CreateShipment.xamlx”. The listing does not show all the eight services registered in the “Logistics” assemblage, rather just the first and the last registered services.

#### 7.2.1.8 Delete an Assemblage

This functionality is mainly designed for the system administrator to delete the assemblages that are no more needed. Figure 7.11 shows the page from which an assemblage can be deleted.

First of all the administrator needs to check which assemblage is going to be deleted and gets its information to be sure that this assemblage is the correct one. When the administrator enters a value in the “Assemblage ID” field in the page shown in the below figure and clicked on the “Get Assemblage Info” button, all attributes of this assemblage together with its Web Services are retrieved. The administrator can first discover the system’s assemblages to retrieve the ID of the assemblage that is going to be deleted (see Section 7.2.1.2).



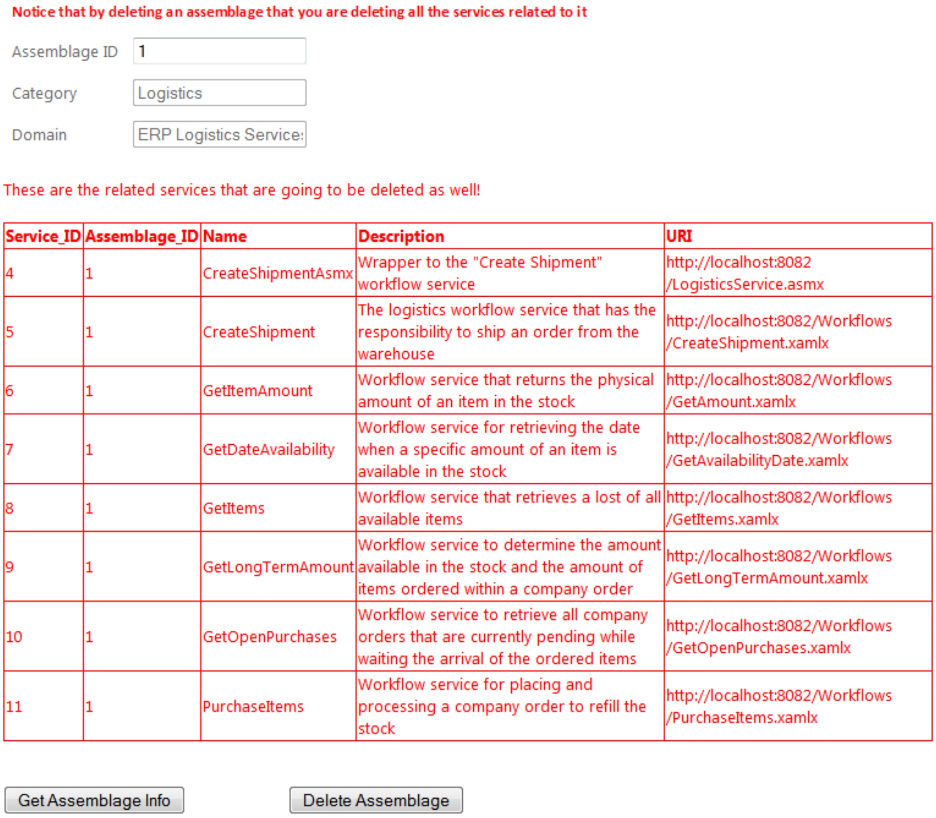
Please enter the Assemblage ID you want to delete. You can discover the RDF description to have more information about specific assemblage.
Click first on Get Assemblage Info button to be sure that this is the assemblage you want to delete. As soon as you click on it, the Delete Assemblage button will be activated so that you can delete the selected assemblage.

Notice that by deleting an assemblage that you are deleting all the services related to it

Assemblage ID

**Fig. 7.11: Get Assemblage’s Information**

Since each assemblage has a set of registered Web Services as members within it, deleting an assemblage means that all the members of this assemblage are going to be deleted as well. An example can be seen in Figure 7.12. The administrator types “1” in the “Assemblage ID” field and clicks on the “Get Assemblage Info” button.



Notice that by deleting an assemblage that you are deleting all the services related to it

Assemblage ID 1

Category Logistics

Domain ERP Logistics Service:

These are the related services that are going to be deleted as well!

| Service_ID | Assemblage_ID | Name | Description | URI |
|---|---|---|---|---|
| 4 | 1 | CreateShipmentAsmx | Wrapper to the "Create Shipment" workflow service | http://localhost:8082 /LogisticsService.asmx |
| 5 | 1 | CreateShipment | The logistics workflow service that has the responsibility to ship an order from the warehouse | http://localhost:8082/Workflows /CreateShipment.xamlx |
| 6 | 1 | GetItemAmount | Workflow service that returns the physical amount of an item in the stock | http://localhost:8082/Workflows /GetAmount.xamlx |
| 7 | 1 | GetDateAvailability | Workflow service for retrieving the date when a specific amount of an item is available in the stock | http://localhost:8082/Workflows /GetAvailabilityDate.xamlx |
| 8 | 1 | GetItems | Workflow service that retrieves a lost of all available items | http://localhost:8082/Workflows /GetItems.xamlx |
| 9 | 1 | GetLongTermAmount | Workflow service to determine the amount available in the stock and the amount of items ordered within a company order | http://localhost:8082/Workflows /GetLongTermAmount.xamlx |
| 10 | 1 | GetOpenPurchases | Workflow service to retrieve all company orders that are currently pending while waiting the arrival of the ordered items | http://localhost:8082/Workflows /GetOpenPurchases.xamlx |
| 11 | 1 | PurchaseItems | Workflow service for placing and processing a company order to refill the stock | http://localhost:8082/Workflows /PurchaseItems.xamlx |

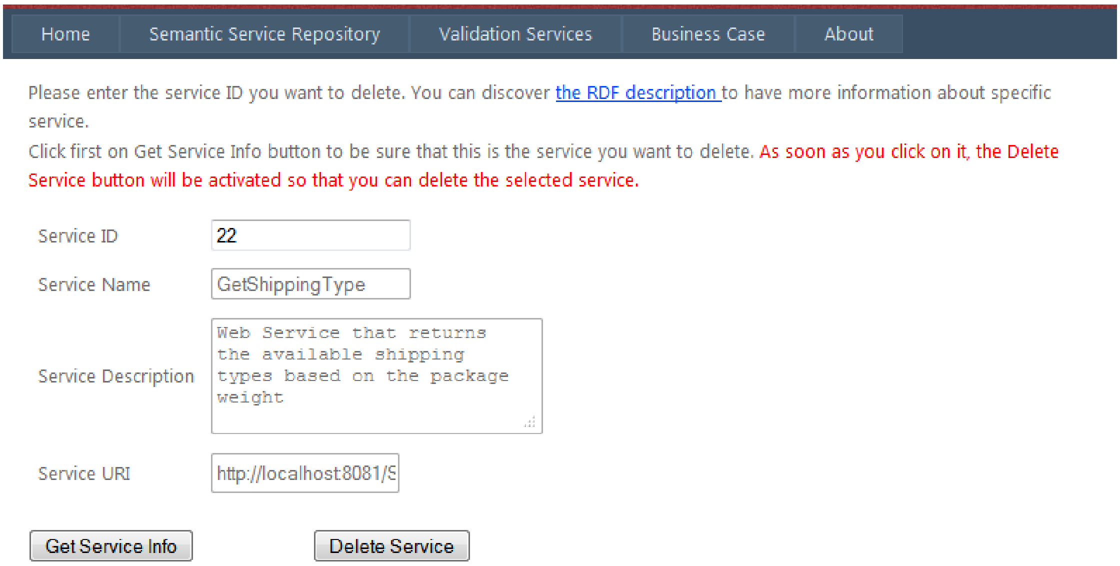
**Fig. 7.12: Delete an Assemblage**

Then it is shown that this assemblage has the category “Logistics” and the domain “ERP Logistics Services”. It is also shown that this assemblage has eight registered Web Services as members. All the attributes of these services are also retrieved and shown in red color to enable better decision from the administrator to be taken. If the deletion decision is final, the administrator clicks then on the “Delete Assemblage” button. Then the assemblage and its services are permanently deleted from the system.

One last issue to mention here is that it is recommended that the assemblages are deleted just when all the registered services within it can’t be reused anymore. Therefore the lifecycle of an assemblage is considered relatively long in comparison with a Web Service’s lifecycle.

### 7.2.1.9 Delete a Service

This site enables the system administrator to delete the system’s registered Web Services. When a Web Service is no more needed and not used anymore, the site shown in Figure 7.13 provides the service deletion functionality. As it is in the assemblage site, the service that is going to be deleted is retrieved based on its ID. System administrators can always discover the registered services in the system (see Section 7.2.1.3) to ensure their deletion decision. For example if the system administrator types “22” in the “Service ID” textbox and clicks on the “Get Service Info” button, all the other attributes of the service are retrieved including the “Service Name”, “Service Description”, and finally “Service URI”.



Please enter the service ID you want to delete. You can discover the RDF description to have more information about specific service.
Click first on Get Service Info button to be sure that this is the service you want to delete. As soon as you click on it, the Delete Service button will be activated so that you can delete the selected service.

Service ID: 22

Service Name: GetShippingType

Service Description: Web Service that returns the available shipping types based on the package weight

Service URI: http://localhost:8081/S

**Fig. 7.13: Delete a Web Service**

Upon retrieving the service information, the authorized user can click on the “Delete Service” button to delete the service permanently from the system.

### 7.2.2 Validation Services

This site of the SESOA's Web application provides a set of validation services. These validation services as shown in Figure 7.14 are used to validate data on a functional level based mainly on data type (primitive and complex data types). RDF validation service is also provided to validate RDF statements.

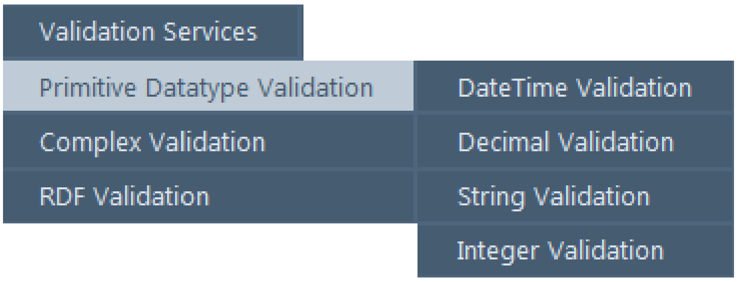


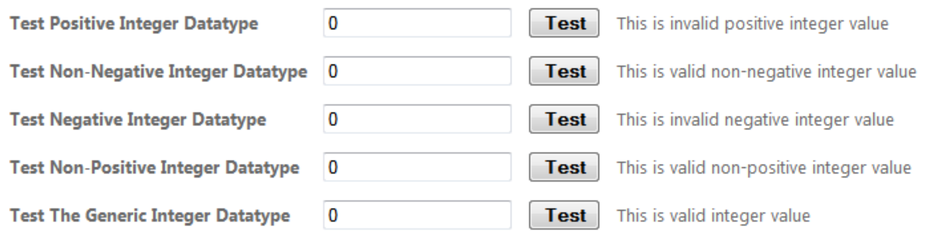
**Fig. 7.14:** The Validation Services

Data type validation services are made available to the service providers so that they can use them as a kind of supporting services to build their own Web Services on top of them. Therefore, their data types can be unified because they use the same validation services. As for the RDF validation service, it is used by the system administrators and the service providers to validate the automatically created RDF relations that link the assemblages and their registered services. The following subsections explain in details all types of validation services.

#### 7.2.2.1 Primitive Validation Services

***Integer Validation Test Service:*** This Web Service checks the validity of the integer data type. In this work, the integer data type is split into four sub-data types namely: positive integer, non-negative integer, negative integer and non-positive integer. These sub-data types are defined as follows:

- Positive integer includes all positive integer values excluding zero.
- Non-negative integer is the generalization of the positive integer data type. It includes all positive integer values plus zero.
- Negative integer includes all negative integer values excluding zero
- Non-positive integer is the generalization of negative integer data type. It includes all negative integer values plus zero.



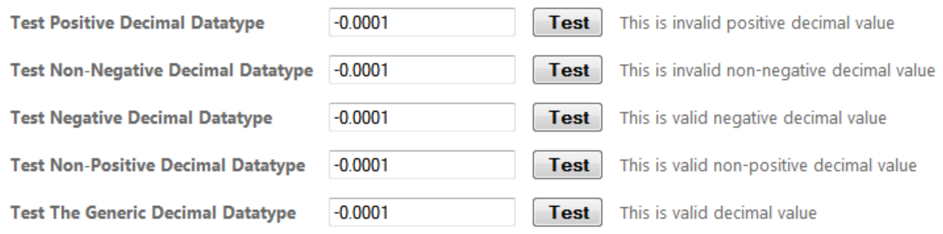
| | | | |
|---|---|---|---|
| Test Positive Integer Datatype | 0 | Test | This is invalid positive integer value |
| Test Non-Negative Integer Datatype | 0 | Test | This is valid non-negative integer value |
| Test Negative Integer Datatype | 0 | Test | This is invalid negative integer value |
| Test Non-Positive Integer Datatype | 0 | Test | This is valid non-positive integer value |
| Test The Generic Integer Datatype | 0 | Test | This is valid integer value |

**Fig. 7.15:** The Interface of the Integer Validation Test Web Service

As depicted in Figure 7.15, the interface of the integer validation test service provides four types of tests to validate the above mentioned four integer sub-data types. In addition to these tests, a generic integer data type validation test is provided.

***Decimal Validation Test Service**:* This Web Service checks the validity of the decimal data type. Like integer, the decimal data type in this work is split into four sub-data types namely: positive decimal, non-negative decimal, negative decimal and non-positive decimal. These sub-data types are defined as follows:

- Positive decimal includes all positive decimal values excluding zero.
- Non-negative decimal is the generalization of the positive decimal data type. It includes all positive decimal values plus zero.
- Negative decimal includes all negative decimal values excluding zero
- Non-positive decimal is the generalization of negative decimal data type. It includes all negative decimal values plus zero.



| | | | |
|---|---|---|---|
| Test Positive Decimal Datatype | -0.0001 | Test | This is invalid positive decimal value |
| Test Non-Negative Decimal Datatype | -0.0001 | Test | This is invalid non-negative decimal value |
| Test Negative Decimal Datatype | -0.0001 | Test | This is valid negative decimal value |
| Test Non-Positive Decimal Datatype | -0.0001 | Test | This is valid non-positive decimal value |
| Test The Generic Decimal Datatype | -0.0001 | Test | This is valid decimal value |

**Fig. 7.16:** The Interface of the Decimal Validation Test Web Service

As depicted in Figure 7.16, the interface of the decimal validation test service provides four types of tests to validate the above mentioned four decimal sub-data types. In addition to these tests, a generic decimal data type validation test is provided.

***String Validation Test Service**:* This Web Service checks the validity of the string data type. In this work, the string data type is split into four sub-data types namely: alpha, numeric, alpha numeric and non-alpha numeric. These four sub-data types are defined as follows:

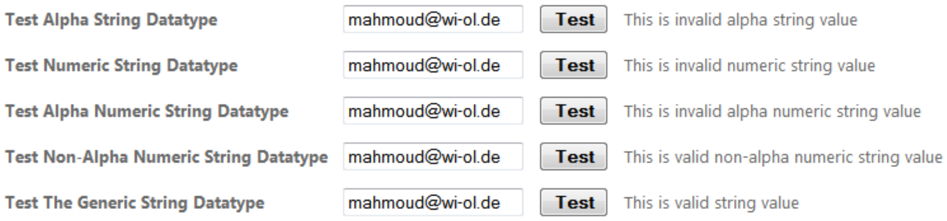
- Alpha string includes all string values that contain just alphabetical letters.

- Numeric string includes just numeric values (plain numbers with the minus “-” or the floating point “.” symbols but excluding any other special character).

- Alpha numeric string includes both alpha and numeric values (letters and numbers respectively) excluding any special character.

- Non-alpha numeric string includes special characters (rest of string characters).

As illustrated in Figure 7.17 below, the interface of the string validation test service provides four types of tests to validate the above mentioned four string sub-data types. In addition to these tests, a generic string data type validation test is provided by this interface.

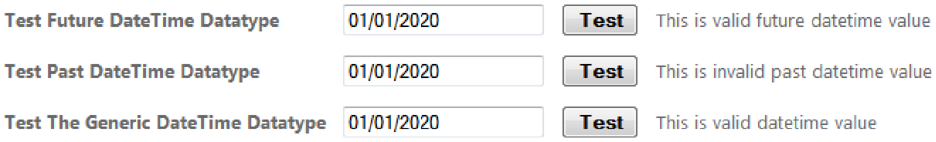


| | | | |
|---|---|---|---|
| Test Alpha String Datatype | mahmoud@wi-ol.de | Test | This is invalid alpha string value |
| Test Numeric String Datatype | mahmoud@wi-ol.de | Test | This is invalid numeric string value |
| Test Alpha Numeric String Datatype | mahmoud@wi-ol.de | Test | This is invalid alpha numeric string value |
| Test Non-Alpha Numeric String Datatype | mahmoud@wi-ol.de | Test | This is valid non-alpha numeric string value |
| Test The Generic String Datatype | mahmoud@wi-ol.de | Test | This is valid string value |

**Fig. 7.17:** The Interface of the String Validation Test Web Service

For example and as shown above, an email address is valid as a non-alpha numeric string data type and as a generic string data type.

***DateTime Validation Test Service:*** This Web Service checks the validity of the DateTime data type. In this work, the DateTime data type values are either valid or invalid and can be classified into future DateTime or past DateTime values.



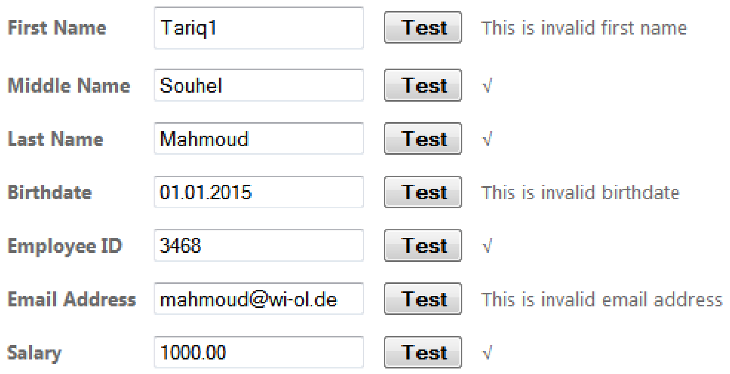
| | | | |
|---|---|---|---|
| Test Future DateTime Datatype | 01/01/2020 | Test | This is valid future datetime value |
| Test Past DateTime Datatype | 01/01/2020 | Test | This is invalid past datetime value |
| Test The Generic DateTime Datatype | 01/01/2020 | Test | This is valid datetime value |

**Fig. 7.18:** The Interface of the DateTime Validation Test Web Service

These two formats can be tested as shown above in Figure 7.18 together with testing the generic DateTime data type.

### 7.2.2.2 Complex Validation Services

This site shows an example of how to use the validation services explained in the previous section. It can be seen as a mixture where all the previous validation tests Web Services are included. For example and as shown below in Figure 7.19, the complex validation can have an interface that validates the employee’s data. These data include the employee’s first, middle, and last names besides basic information like birthdate, employee ID, email address and salary.



| | | | |
|---|---|---|---|
| First Name | Tariq1 | Test | This is invalid first name |
| Middle Name | Souhel | Test | √ |
| Last Name | Mahmoud | Test | √ |
| Birthdate | 01.01.2015 | Test | This is invalid birthdate |
| Employee ID | 3468 | Test | √ |
| Email Address | mahmoud@wi-ol.de | Test | This is invalid email address |
| Salary | 1000.00 | Test | √ |

**Fig. 7.19:** Complex Validation Test Example

What can be concluded from the above figure is that the values of each employee's first, middle and last names are from the alpha string data type. The employee's birthdate value has to belong to the past DateTime data type, the employee ID value belongs to the positive integer data type. Furthermore, the email address value belongs to the non-alpha numeric string data type (since it is validated against the "@" special character) and finally the salary value belongs to the positive decimal data type.

### 7.2.2.3 RDF Validation Service

In this work and to validate the generated RDF statements, the known W3C RDF validation service has been used. This service accepts URI or easily RDF/XML document as input and produces a 3-tuple representation or a graphic visualization as output. This service is based on the java-based Another RDF Parser (ARP). The version that has been used in this work is the 2-alpha-1 version. Table 7.6 shows the output of parsing the RDF part of the "Validation" assemblage.

**Tab. 7.6:** RDF Validation - Triple Representation

| Number | Subject[58] | Predicate[59] | Object[60] |
|---|---|---|---|
| **1** | Validation | hasMember | IntegerValidation |
| **2** | Validation | hasMember | DecimalValidation |
| **3** | Validation | hasMember | StringValidation |
| **4** | Validation | hasMember | DateTimeValidation |

Changing the output style of parsing the same part to a graphical visualization results in the graph depicted in Figure 7.20.

[58] The full entity name is: http://asbl.wi-ol.de/sesoa/assemblage/Validation

[59] The full entity name is: http://asbl.wi-ol.de/sesoa#hasMember

[60] The full entity name is prefixed by: http://asbl.wi-ol.de/sesoa/services/

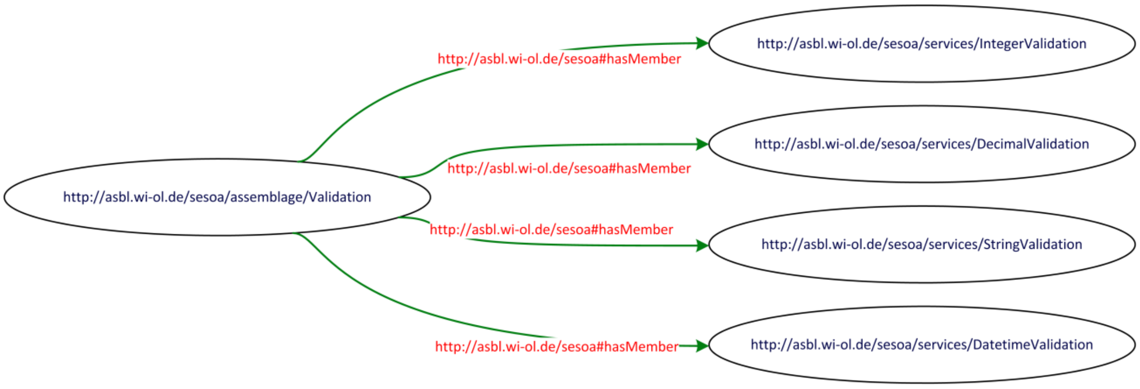

http://asbl.wi-ol.de/sesoa/assemblage/Validation
http://asbl.wi-ol.de/sesoa#hasMember
http://asbl.wi-ol.de/sesoa/services/IntegerValidation
http://asbl.wi-ol.de/sesoa#hasMember
http://asbl.wi-ol.de/sesoa/services/DecimalValidation
http://asbl.wi-ol.de/sesoa#hasMember
http://asbl.wi-ol.de/sesoa/services/StringValidation
http://asbl.wi-ol.de/sesoa#hasMember
http://asbl.wi-ol.de/sesoa/services/DatetimeValidation

**Fig. 7.20:** RDF Validation - Graph Representation

The triples and the graph both shows that the assemblage “Validation” represents the subject, the predicate is the “hasMember” relation, and the object is one of the data type validation services. Using the RDF validation service helps in detecting any error that might appear in the automatic generation of the assemblage-Web Service relations. Moreover having the triples and graph representations can help the system administrator in managing the system’s assemblages in a better way.

## 7.3 The Business Case Web Application

The architecture of the implemented business case has been already introduced in the Section 6.6. The main components of this architecture have been then depicted in Figure 7.2 in Section 7.1.2. In this section, the implementation details of these components are explained. The business case implementation was done by a group of students in “VLBA Seminar: SOA and Business Processes” at the department of Business Information Systems in Oldenburg University (Denker et al., 2011)[61]. This business case has implemented on top of the SESOA prototype. It is workflow-based Web application and its workflows have been implemented as WCF workflow services that can be discovered and invoked as Web Services. Most of activities that build the system’s workflows have been implemented as Web Services. These Web Services published in the SESOA semantic service repository as members of its assemblages can be called while the workflows are executed.

Before going deep into the implementation details, some words about the business case itself are good to get the idea behind it. This business case represents a fictional online shop for selling watches. This case has been implemented using several technologies including C#, ASP.NET, Visual Basic.NET, Windows Server 2008 R2, Microsoft SQL Server 2008 R2, and Microsoft Internet Information Service (IIS Version 7). By using IIS as can be seen in Table 7.7, different sites have been created on different ports to realize the business case’s functionalities.

[61] The handout of this seminar is internal document. It is available at the Business Information Systems I department at Oldenburg University.

**Tab. 7.7:** Business Case's Sites and their Ports

| Number | Site Name | Port |
|---|---|---|
| **1** | Website (GUI) | 80 |
| **2** | Shop | 8080 |
| **3** | Shipment | 8081 |
| **4** | Logistics | 8082 |
| **5** | Payment | 8084 |
| **6** | ItemImages | 8086 |
| **7** | Mail | 8087 |
| **8** | Coordination | 8090 |

The first site "Website" represents the main front end for this business case. "Shop" site represents the link with the system's database. The "Shipment" site links the business case to external shipment services and similarly the "Payment" site links it to external payment services. The "Logistics" site constitutes the backend of this fictional shop where the material planning together with the inventory and warehouse management are the main responsibilities. The "ItemImages" site is where the shop images are stored. The "Mail" site is responsible of contacting the shop's customers to notify them with the state of their orders. Finally, the "Coordination" site is the backbone site in which the main coordinating workflow is executed. This site put the parts from all the other sites together to finalize a customer order. Moreover, it is responsible of contacting the semantic service repository to call the Web Services involved in executing the other sites' workflows. This is done using a SPARQL query tool like Twinkle to query the RDF statements of the semantic service repository and then to give the control back to the running workflow. More information about this query tool is presented in the Section 7.3.2.

### 7.3.1 WF and WCF

This section explains briefly the main features of Microsoft Windows Workflow Foundation (WF) and Microsoft Widows Communication Foundation (WCF) technologies. WF is used in this business process as an in-process workflow engine for modeling and executing its workflows. Since all the components in the business case's architecture are implemented using Web Services, Workflow Foundation technology can be easily replaced by any alternative workflow engine, like IBM WebSphere Process Server[62] or even the W3C's SCXML[63].

[62] IBM WebSphere Process Server has similar properties in comparison to Microsoft Workflow Foundation. One of these properties allows exposing workflows as Web Services. This is the key feature that allows replacing Workflow Foundation used in this work with IBM WebSphere Process Server. More

Since all SESOA applications have been implemented using Microsoft technologies, WF 4.0 has been adopted as the main workflow engine to implement workflows in this business case. WF consists of the actual workflow engine besides an extension to the Visual Studio 2010 IDE to model workflows. This extension is a graphical editor through which all the activities that make up the workflow can be used using drag and drop technique. Alternatively, workflows can be modified by editing their underlying XAML[64] code, which is Microsoft XML-based notation used to describe WF workflows (Scott Allen, 2006, p. 25).

Implemented using WF, Workflows consist of a mixture of predefined and custom activities. A predefined activity allows for standard operations and flows like conditional branches, loops, or sends and receives. They all have the same “Activity” base class in common. This class defines all the necessary methods like for example “Execute” and “Cancel” that are required by the workflow engine. Custom activities are treated the same way as the predefined activities (Scott Allen, 2006, pp. 11–12). They are derived from the same “Activity” base class. An example for a custom activity is a code activity, which is derived from the “CodeActivity” class (which is in turn derived from the common “Activity” base class) and implements some logic using C# code.

As mentioned in (Scott Allen, 2006, pp. 17–19), WF workflow engine offers set of different services to support the controlling and execution of its workflows. These services are the runtime, scheduling, transaction, persistence and tracking services:

- *The Runtime Service* offers only basic functionality for executing a workflow. This service is extendable by other services that add new features to enhance the workflow execution.

- *The Scheduling Service* controls the runtime threads. The default scheduling runtime delivered with WF creates new threads for each workflow. Therefore, the workflows can be executed in parallel.

- *The Transaction Service* keeps the internal state of a workflow in synchronization with a database. By default, multiple activities of a single workflow instance can share the same transaction context.

- *The Persistence Service* allows saving the internal state of a workflow in a workflow instance store. This is useful for long-running workflows that can be persisted and restored from the workflow instance store whenever they are needed.

information about the features of IBM WebSphere Process Server is available online at: http://www-01.ibm.com/software/au/integration/wps/features/

[63] State Chart XML (SCXML): State Machine Notation for Control Abstraction represents a general-purpose event-based state machine language. More information about SCXML can be found online at: http://www.w3.org/TR/scxml/

[64] XAML stands for Extensible Application Markup Language.

- *The Tracking Service* allows monitoring and tracking of workflow instances. SQL tracking to store data in SQL Server databases is also enabled by WF 4.0.

After this brief explanation of the WF basic features, this section explicates in short Microsoft Windows Communication Foundation technology. WCF can be considered as successor to various older inter-process communication technologies like COM+[65], DCOM, and .NET Remoting.

In this implemented business case, most of the WF workflows' activities have been implemented as Web Services using WCF. As mentioned in (Sharp, 2010, p. 42), WCF represents the ideal platform to implement SOA and Web Services. Therefore, WCF is used in this work to provide the necessary tools for building Web Services. Last but not least, WCF achieves interoperability by supporting open standards like the W3C Web Service standards.

### 7.3.2 SPARQL Queries

To call a Web Service, the corresponding component[66] communicates with the semantic service repository to locate the desired assemblage in which the service is registered and then retrieves its endpoint. Selecting service depends usually on the user's preferences based on its Service Level Agreement (SLA)[67]. The system's workflows can discover the SESOA semantic service repository using any kind of RDF query language tools. In this work, SPARQL has been used as a query language and Twinkle 2.0 as a SPARQL query tool. Listing 7.4 shows how RDF queries have been used.

**Listing 7.4:** Generic SPARQL Query

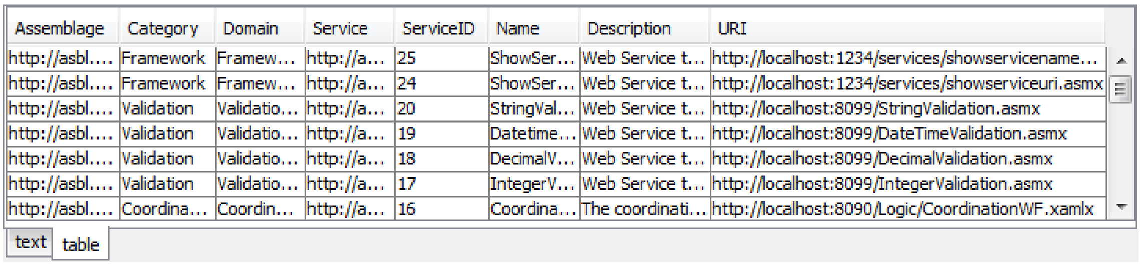
```
PREFIX sesoa:<http://asbl.wi-ol.de/sesoa/>
PREFIX WS:<http://asbl.wi-ol.de/sesoa/services/>
SELECT *
WHERE
{
?Assemblage sesoa:hasCategory ?Category.
?Assemblage sesoa:hasDomain ?Domain.
?Assemblage sesoa:hasMember ?Service.
?Service WS:hasID ?ServiceID.
?Service WS:hasName ?Name.
?Service WS:hasDescription ?Description.
?Service WS:hasURI ?URI.
}
```

[65] COM+ is the successor of Microsoft Component Object Model (COM) standard. DCOM is a Microsoft's proprietary technology used to ease software components' communication in distributed networks.

[66] All the system's workflows have to contact the SESOA semantic service repository to locate the services needed to execute their activities.

[67] SLA represents the service contract that helps the consumer in deciding whether to use the service or not. SLA normally defines service's availability, performance, price…

The listing above applies a general RDF query to the SESOA semantic service repository to retrieve a list of all its assemblages and their registered Web Services. Figure 7.21 shows the results of applying the query to the SESOA semantic service repository. As shown in the figure below, the result can be obtained either in text or table forms.



| Assemblage | Category | Domain | Service | ServiceID | Name | Description | URI |
|---|---|---|---|---|---|---|---|
| http://asbl.… | Framework | Framew… | http://a… | 25 | ShowSer… | Web Service t… | http://localhost:1234/services/showservicename… |
| http://asbl.… | Framework | Framew… | http://a… | 24 | ShowSer… | Web Service t… | http://localhost:1234/services/showserviceuri.asmx |
| http://asbl.… | Validation | Validatio… | http://a… | 20 | StringVal… | Web Service t… | http://localhost:8099/StringValidation.asmx |
| http://asbl.… | Validation | Validatio… | http://a… | 19 | Datetime… | Web Service t… | http://localhost:8099/DateTimeValidation.asmx |
| http://asbl.… | Validation | Validatio… | http://a… | 18 | DecimalV… | Web Service t… | http://localhost:8099/DecimalValidation.asmx |
| http://asbl.… | Validation | Validatio… | http://a… | 17 | IntegerV… | Web Service t… | http://localhost:8099/IntegerValidation.asmx |
| http://asbl.… | Coordina… | Coordin… | http://a… | 16 | Coordina… | The coordinati… | http://localhost:8090/Logic/CoordinationWF.xamlx |

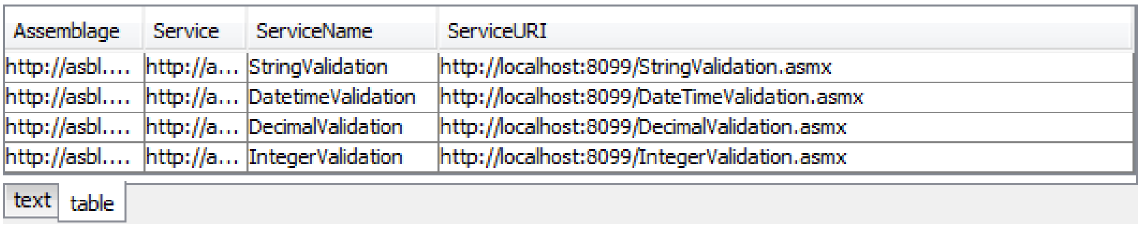
**Fig. 7.21:** Result of the Generic RDF Query

Another and more concrete example of applying SPARQL RDF queries to the SESOA semantic service repository is shown in the Listing 7.5. It is written to retrieve all the registered Web Services within the "Validation" assemblage.

**Listing 7.5:** Specific SPARQL Query

```
PREFIX sesoa:<http://asbl.wi-ol.de/sesoa/>
PREFIX WS:<http://asbl.wi-ol.de/sesoa/services/>
SELECT *
WHERE
{
?Assemblage sesoa:hasCategory "Validation".
?Assemblage sesoa:hasMember ?Service.
?Service WS:hasName ?Name.
?Service WS:hasURI ?URI.
}
```

Figure 7.22 shows the results of applying this query to the SESOA semantic service repository. As can be seen in this figure, there are four Web Services registered in the "Validation" assemblage.



| Assemblage | Service | ServiceName | ServiceURI |
|---|---|---|---|
| http://asbl.… | http://a… | StringValidation | http://localhost:8099/StringValidation.asmx |
| http://asbl.… | http://a… | DatetimeValidation | http://localhost:8099/DateTimeValidation.asmx |
| http://asbl.… | http://a… | DecimalValidation | http://localhost:8099/DecimalValidation.asmx |
| http://asbl.… | http://a… | IntegerValidation | http://localhost:8099/IntegerValidation.asmx |

**Fig. 7.22:** Result of a Specific RDF Query

Every assemblage, Web Service, or relation in the SESOA database system is stored as RDF entity composed of ID and value pair. For example the aforementioned "Validation" assemblage is stored in the SESOA database system using the value: http://asbl.wi-ol.de/sesoa/assemblage/Validation besides a unique ID. This applies to the Web Services and their relations as well. The IDs of the assemblages, relations, and services are used to create the subjects, predicates, and objects in the RDF statements respectively. Table 7.8 shows an example of an RDF statement stored in the SESOA database. The values in this table are shortened for presentation reasons and the full values can be seen as footnotes. The assemblage “Validation” (ID 48) relates using the “hasMember” relation (ID 4) to the “DecimalValidation” Web Service (ID 51). The IDs of the subjects, predicates, and objects forms an RDF triple that is stored in the system’s database. All the attributes of the assemblages, relations, and the Web Services are published in the semantic service repository and can be retrieved using SPARQL query tool like Twinkle.

**Tab. 7.8:** Entities Properties

| | Subject[68] | Predicate[69] | Object[70] |
|---|---|---|---|
| **ID** | 48 | 4 | 51 |
| **Value** | Validation | hasMember | DecimalValidation |

The following section gives some highlights to the implementation of the business case’s workflows and how these workflows used the abovementioned query tool to get the necessary information to call the Web Services published in the SESOA semantic service repository.

### 7.3.3 Implementation Details

The following subsections give some implementation details regarding the components of the business case and its accompanying online shop. The linkage to the SESOA Web application is also illustrated.

#### 7.3.3.1 The Website Component

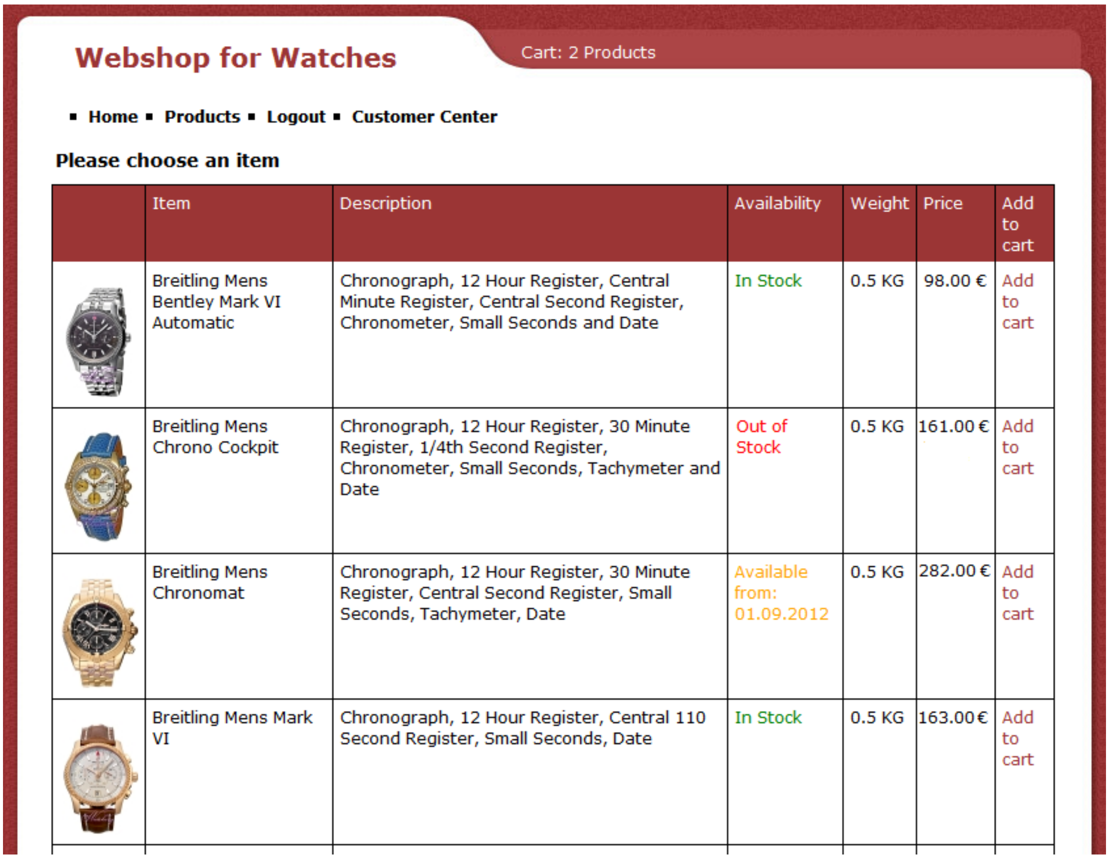
All of the functionalities assigned to the business case that are detailed at the requirement definitions phase (see Chapter 5) are implemented in the business case’s Web application. GUI elements of the fictional online shop can be seen in Figure 7.23. The site structure of this GUI is depicted in Figure 7.24. The GUI is used to handle purchases done by the customers. It gives them the possibility to browse the product catalogue and accomplish the ordering process. If the potential customers open the Web page, it offers them a default start page with a welcome message and an introduction of products offered in the Web shop. Using the navigation on the header, the customer is able to navi-

[68] The subject here represent an assemblage called: “Validation” that has the ID 48 and its actual value is: http://asbl.wi-ol.de/sesoa/assemblage/Validation

[69] The predicate here represent a relation called: “hasMember” between an assemblage and a registered Web Services. This relations has the ID 4 and its actual value is http://asbl.wi-ol.de/sesoa/hasMember

[70] The object here represent a Web Service called: “DecimalValidation” that has the ID 51 and its actual value is: http://asbl.wi-ol.de/sesoa/services/DecimalValidation

gate through the whole website. He/she has the possibility to show the product list besides logging into his/her account. If logged in, he/she is able to view the customer center in which the functionality to show the previous completed orders is provided.



Webshop for Watches

Cart: 2 Products

Please choose an item

| | Item | Description | Availability | Weight | Price | Add to cart |
|---|---|---|---|---|---|---|
| | Breitling Mens Bentley Mark VI Automatic | Chronograph, 12 Hour Register, Central Minute Register, Central Second Register, Chronometer, Small Seconds and Date | In Stock | 0.5 KG | 98.00 € | Add to cart |
| | Breitling Mens Chrono Cockpit | Chronograph, 12 Hour Register, 30 Minute Register, 1/4th Second Register, Chronometer, Small Seconds, Tachymeter and Date | Out of Stock | 0.5 KG | 161.00 € | Add to cart |
| | Breitling Mens Chronomat | Chronograph, 12 Hour Register, 30 Minute Register, Central Second Register, Small Seconds, Tachymeter, Date | Available from: 01.09.2012 | 0.5 KG | 282.00 € | Add to cart |
| | Breitling Mens Mark VI | Chronograph, 12 Hour Register, Central 110 Second Register, Small Seconds, Date | In Stock | 0.5 KG | 163.00€ | Add to cart |

**Fig. 7.23:** The Business Case Web Application

Independent from the status of login, all the website's visitors are able to fill the shopping cart by choosing items from the products page. The cart is accessible using the link placed in the header on the right side. If a user filled the cart, he/she is able to start the checkout process. The first step then is to prove if the user already logged in using an existing account. If not, the system asks the user to login or to create a new account. The next step is to let the user choose invoice and shipping addresses. The possibility to add more invoice or shipping addresses is also possible. Selecting a shipping service from a list of available shipping services is then provided. The system gives then the customer the estimated delivery time and fees for each selected shipping service. After that, the user should select one of the system's payment methods. These methods request input data from the customer like credit card or bank account data. Upon providing the system with the needed data, the final possibility to prove the order before making it final is given to the customer. By accepting the current terms and conditions of the Website with a final click on the proceed button, an order is placed in the system. The customer is then able at any time to see the history of his/her orders besides order status using the customer center. While executing the selling process, several updates will be applied to the order. The customer center gives the user the opportunity to get an overview about the updates of his order. The same updates are sent to the user email address as well.

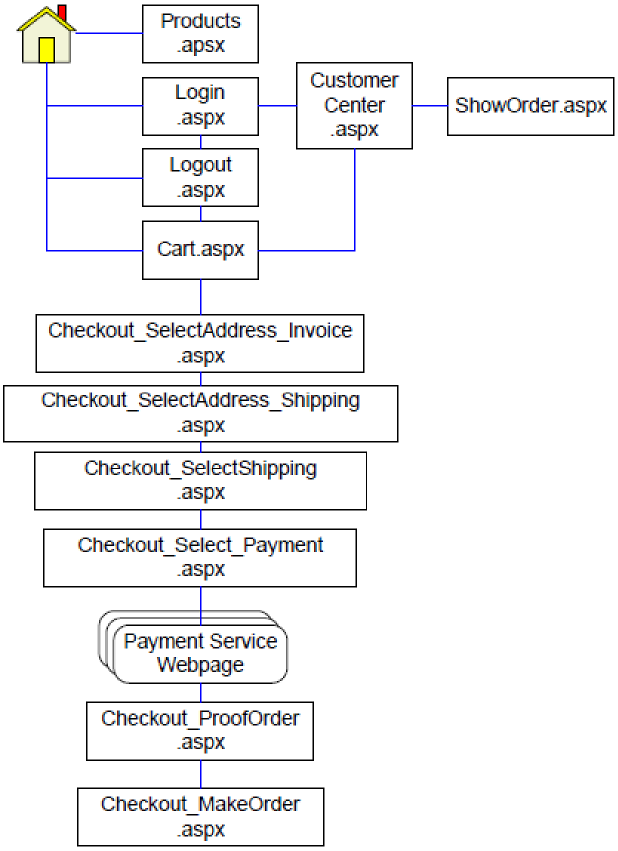
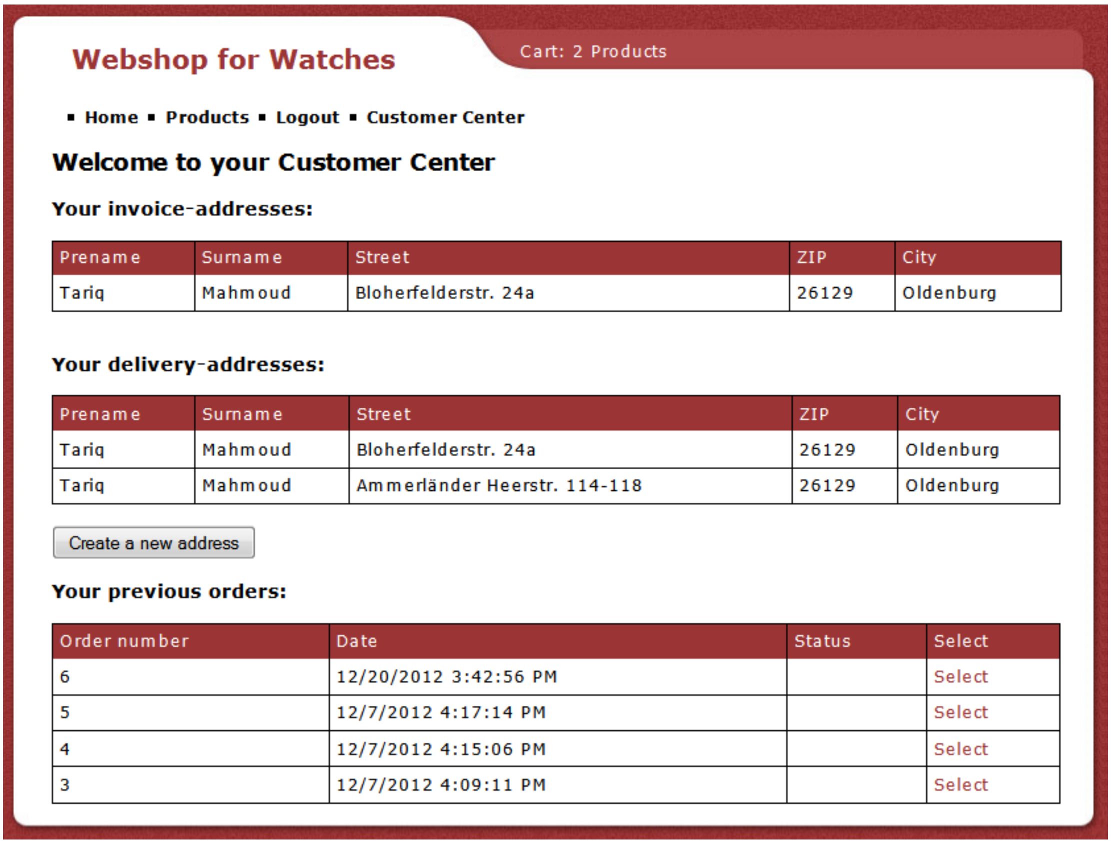

Products .apsx
Login .aspx
Logout .aspx
Cart.aspx
Customer Center .aspx
ShowOrder.aspx
Checkout_SelectAddress_Invoice .aspx
Checkout_SelectAddress_Shipping .aspx
Checkout_SelectShipping .aspx
Checkout_Select_Payment .aspx
Payment Service Webpage
Checkout_ProofOrder .aspx
Checkout_MakeOrder .aspx

**Fig. 7.24:** Business Case GUI Structure

The implementation of the abovementioned GUI has been realized using ASP.NET and VB.NET. One of the most interesting features in developing this GUI is that the Web front end has no direct database connection. To read and process data, it uses a Web Service imported from the shop component. Therefore, database changes do not affect the changes on the Web front end. As a consequence, the whole front end design can be easily substituted with another one that will use the same Web Service accessing the system’s database. The process of requesting any data starts from the browser of a potential customer. It actually starts with a user request, for example viewing the product list. The IIS handles the received HTTP request and forwards it to the called ASP.NET page. Fulfilling the user request, the ASP.NET page calls the implemented VB.NET class that sends a request to the Web Service that has a connection to the database to retrieve the product information. To realize the shopping cart functionality, .NET session objects are used. Using session objects makes it possible to transport user given inputs over the whole Web application’s pages.



Welcome to your Customer Center

Your invoice-addresses:

| Prename | Surname | Street | ZIP | City |
|---|---|---|---|---|
| Tariq | Mahmoud | Bloherfelderstr. 24a | 26129 | Oldenburg |

Your delivery-addresses:

| Prename | Surname | Street | ZIP | City |
|---|---|---|---|---|
| Tariq | Mahmoud | Bloherfelderstr. 24a | 26129 | Oldenburg |
| Tariq | Mahmoud | Ammerländer Heerstr. 114-118 | 26129 | Oldenburg |

Your previous orders:

| Order number | Date | Status | Select |
|---|---|---|---|
| 6 | 12/20/2012 3:42:56 PM | | Select |
| 5 | 12/7/2012 4:17:14 PM | | Select |
| 4 | 12/7/2012 4:15:06 PM | | Select |
| 3 | 12/7/2012 4:09:11 PM | | Select |

**Fig. 7.25:** The Customer Center

ASP.NET provides master pages and site navigation feature in which the whole layout is written in one master file that is included in the header of every page. Master pages and site navigation feature allows defining several content placeholders that can be filled by every webpage separately.

Every registered customer has the possibility to access his customer center that is directly shown to the user after a successful logging. It gives the user an overview of his/her entered addresses and shows him/her a list of previous orders as depicted in Figure 7.25.

### 7.3.3.2 The Shop Component

The shop component has two roles in the selling business process. On the one hand, it acts as a backend for the website component (the front end) and on the other hand it handles order proceeding after a customer has created a new order by communicating with the coordination component. This is the only component that has direct connection to the system’s database and is therefore responsible for all database transactions. It offers the other components a variety of Web Services that read from or write data into the database. This component is implemented in three phases:

1. Design a database model specific for the selling business process

2. Implementation of this database model and filling it with exemplary data

3. Implementation of all necessary Web Services needed for the communication with the other component's workflows

The designed database model is shown in Figure 7.26 as UML class diagram. The center of this model is an order. An order is connected to a list of items. Each item in an order has a specific price. Prices are always valid for a specific period of time. If the price for an item needs to be changed, a new entry to the price table has to be added and the validation period for the old price must be changed. Furthermore an order is connected to a customer. Each customer consists of several attributes like for example the login email address and password. A customer is also connected to one or more addresses. An address can be either a billing or a shipping address. The address table is connected to the shipping and payment tables. The shipping represents a shipment of an order and therefore has attributes like shipping date and tracking link. Shipping is connected to one or more shipping types that can be for example international shipment via DHL or national shipment via Hermes.

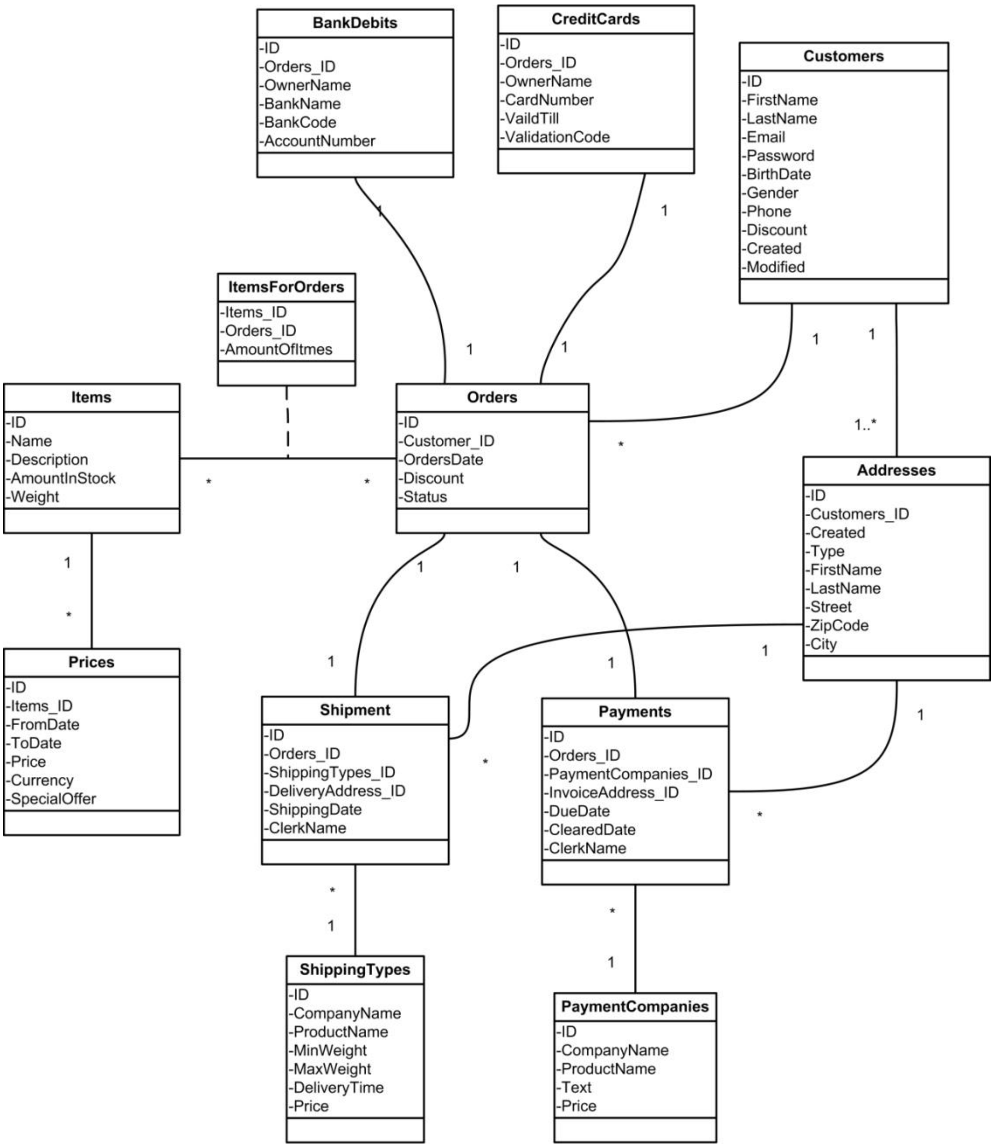


**Fig. 7.26:** The Database Model for the Selling Business Process

The payment table represents a payment for an order and therefore has attributes like payment date. Each payment is connected to one or more payment companies. A payment company offers payment methods like for example MasterCard or bank transfer. Depending on the chosen payment method, an order can be connected to credit card or bank transfer payments. If the customer has chosen to pay via MasterCard, American Express, or VISA, his credit card information will be stored in the credit card table. If the customer wants to pay via direct debit, the bank debit table will contain his/her account information. The database model shown above is implemented using Microsoft SQL Server 2008 R2 and its tables have been filled with exemplary values.

The last phase in the realization of this component was the implementation of its Web Services. This component consists of 37 Web Service operations encapsulated in three ASP.NET Web Services. Most of these operations are used for the communication with the website component. Each time the website needs specific information or wants to store specific information in the database, it calls an appropriate Web Service operation. Table 7.9 lists all operations implemented at the shop component.

**Tab. 7.9:** The Web Service Operations of the Shop Component

| Operation Name | Description |
|---|---|
| **GetItem** | Return an item object |
| **ListAllItems** | Returns list of all items |
| **AddItemToOrder** | Adds an item to the current shopping cart |
| **CreateCustomer** | Creates a new customer |
| **GetCustomer** | Returns a customer object |
| **AuthenticateCustomer** | Returns true value for a valid email/ password combination |
| **GetCustomerOrders** | List all orders for a given customer |
| **SendOrder** | Sends an order to the coordination component |
| **CreateOrder** | Creates a new (empty) order |
| **GetOrder** | Returns an order object |
| **GetCart** | Returns the current cart content |
| **GetOrdersCustomer** | Returns the customer info of a given order |
| **GetOrdersShipping** | Returns the shipping info of a given order |
| **GetOrdersPayment** | Returns the payment info of a given order |
| **GetOrdersPaymentCompany** | Returns the payment company of a given order |
| **GetOrdersShippingType** | Returns the shipping type of a given order |
| **GetOrdersBankdebit** | Returns the bank debit info of a given order |
| **GetOrdersCreditcard** | Returns the credit card info of a given order |
| **SetOrderPaymentToBankDebit** | Sets the payment method to bank debit |
| **SetOrderPaymentToMasterCard** | Sets the payment method to MasterCard |
| **SetOrderPaymentToVisa** | Sets the payment method to Visa card |

| | |
|---|---|
| **SetOrderPaymentToAmericanExpress** | Sets the payment method to American Express card |
| **SetOrderPaymentToBankTransfer** | Sets the payment method to bank transfer |
| **SetOrderShipping** | Sets the shipping for a given order |
| **GetAvailableShippingTypes** | List the available shipping types for a list of items |
| **GetShippingType** | Returns the shipping type object |
| **GetCurrentItemPrice** | Returns the current price for an item |
| **GetOrderItemPrice** | Returns the price of an item in a given order |
| **GetOrderPrice** | Returns the price for an order |
| **GetOrdersPaymentPrice** | Returns the payment price of an order |
| **GetOrdersShippingPrice** | Returns the shipping price of an order |
| **GetPaymentPrice** | Returns the current price of using a payment method |
| **GetShippingPrice** | Returns the current price of using a shipping type |
| **CreateCustomerAddress** | Creates a new address for a given customer |
| **ListCustomerAddresses** | Returns all addresses associated to a given customer |
| **GetShippingAddress** | Returns the shipping address of an order |
| **GetInvoiceAddress** | Returns the invoice address of an order |
| **GetAvailabilityDate** | Returns the date when an item will be available (or the current date for items already available in stock) |
| **GetAmount** | Returns the amount of a specific item available in stock |
| **SetShippingTrackingLink** | Sets the order’s tracking link |
| **SetShippingDate** | Sets the order’s shipping date |
| **SetPaymentDate** | Sets the order’s payment date |

An example of such operations is the “ListAllItems” Web Service operation that reads from the database. It reads all information from all products in the database including their current price. It is invoked when the customer wants to see all offered products in the shop. Another Web Service operation example of writing data in the system’s database is the “CreateCustomer”. When a customer fills in the registration form his/her information, this operation is invoked to add a new entry to the customer table. Upon creating a new order, the website invokes the “SendOrder” Web Service operation. It reads all information from the order table and sends them to the coordination component. While the coordination component processes an order, it calls three other Web Service operations. These operations were implemented to set the order’s payment date, shipping date, and tracking link so that the customer is able to track his/her order.

#### 7.3.3.3 The Logistics Component

This implementation of the logistics component handles mainly all interactions with the stock of the online shop using workflows. All workflows in this component are implemented as workflow services and published in the SESOA semantic service repository. Therefore, these workflows can be discovered and invoked as Web Services. These workflows are registered in the “Logistics” assemblage in the SESOA semantic service repository. RDF queries can be executed on this repository using “Twinkle” tool based on the assemblage name, the service (workflow service) name, or the assemblage-service relation to retrieve the workflows’ information. Table 7.10 lists the implemented workflows in the logistics component indicating the workflow name and a short description.

**Tab. 7.10:** Implemented Workflow Services in the Logistics Component

| Name | Description |
|---|---|
| **CreateShipment** | The main logistics workflow for shipment process |
| **GetAmount** | Returns the amount of a specific item available in the stock |
| **GetAvailabilityDate** | Returns the date when an item will be available (or the current date for items available in the stock) |
| **GetLongTermAmount** | Returns the amount of an item available in the stock plus the amount of the company orders for this item |
| **GetOpenPurchases** | Returns the company orders that are currently pending waiting for the ordered items to arrive to the stock |
| **PurchaseItems** | Creates a new company order to replenish the stock |

The main functions implemented in this component include: adding and removing items from the stock, marking items as reserved, preparing items for shipping, and placing item orders to replenish the stock. The logistics solution is split into three projects: the logistics main project, the logistics client project, and the logistics policy project.

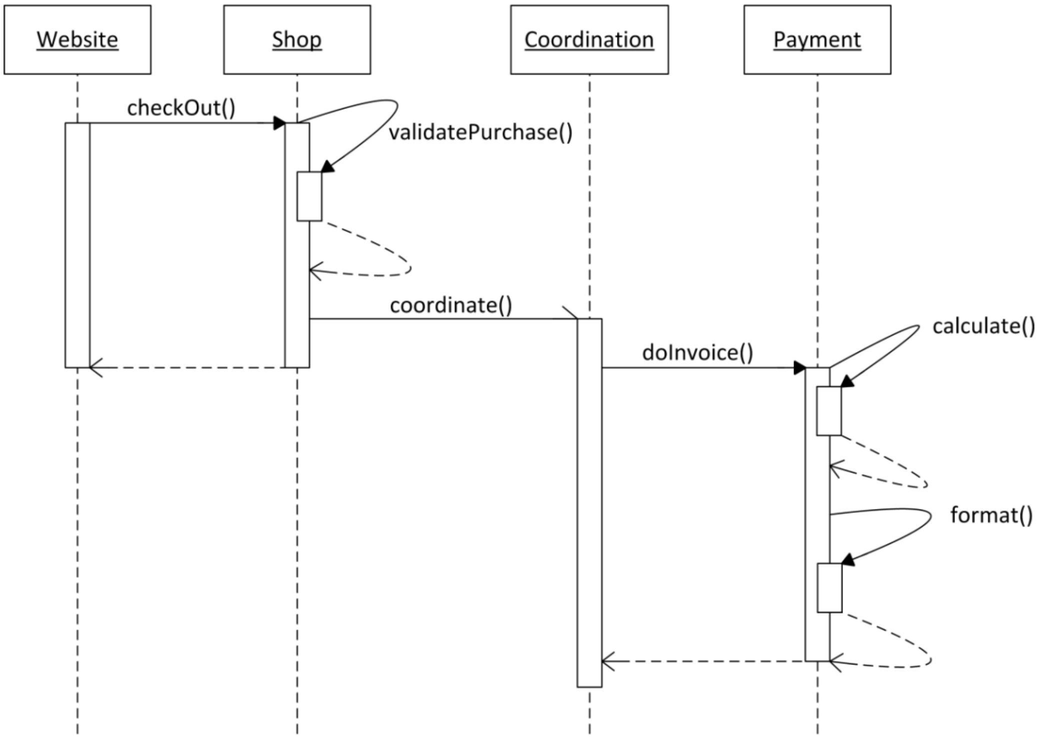
- *The Logistics Main project* represents the main logic of the entire logistics solution. It contains workflows and services that prove the availability of an item in stock and determine the estimated delivery date of that item if it is out of stock. The biggest two workflows: “CreateShipment” and “PurchaseItems” are implemented in this project. The “PurchaseItems” workflow is just used internally (by the “CreateShipment” workflow and the logistics client project). “CreateShipment” workflow is wrapped to a W3C-compliant standard Web Service to be used internally in this project and externally by the other system’s components.

- *The Logistics Client Project* is implemented mainly to handle the online shop’s logistics actions. For instance, it can be used to request the amount of an available item in the stock. Furthermore, all the shop’s backbone functions are implemented in this project. Examples of these functions include placing and processing item orders to replenish the stock.

- *The Logistics Policy Project* represents a rule engine that decides whether the stock has to be replenished or not. If it needs replenishment, item order needs to be placed automatically. This project has been implemented using a predefined activity called the "PolicyActivity". It represents set of rules in which a rule merely represents a constraint and an action to handle this constraint. The rules (and the actions to handle these rules) to check the stock status have been created in this project using this predefined activity.

#### 7.3.3.4 The Payment Component

The payment component is internal billing system in this business case that issues invoices out of given data. In order to issue an invoice, the billing Web Service receives a data set contains customer, product, shipping and payment details. Based on the data in this data set, the total is calculated considering the price summation of all ordered items, a possible discount added to the payment and shipping prices. The shipping, payment, and discount are order-dependent details. Issuing an invoice is finished after the inclusion of the order-dependent details. The sequence diagram in Figure 7.27 below illustrates the data flow in the payment component.



Source: Figure 24 in **(Denker et al., 2011, p. 40)**

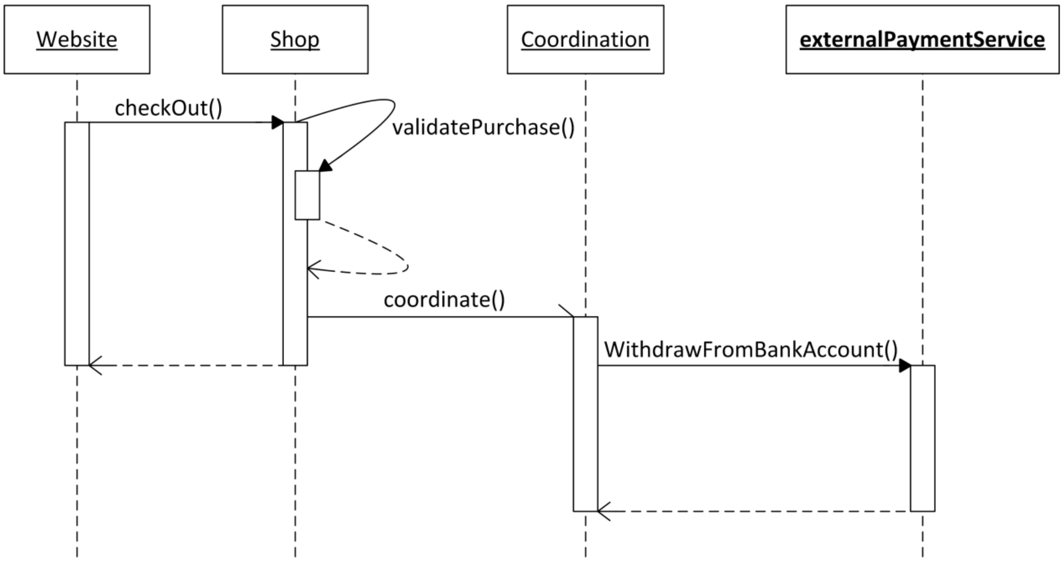
**Fig. 7.27:** Dataflow in the Payment Component

After receiving order-dependent data, the coordination workflow in the coordination component finalizes the data set and sends it to the payment component. The invoice is then calculated and formatted successively. As a result the coordination workflow receives back the invoice formatted by the payment component as object. This object is further processed by the coordination workflow and the invoice including this object is sent as email body to the customer via the mail component. Table 7.11 lists all the Web Service's operations implemented at the payment component.

**Tab. 7.11:** The Web Service Operations of the Payment Component

| Operation Name | Description |
|---|---|
| **DoInvoice** | Creates invoice out of given data set |
| **GetPriceForItem** | Returns items price out of given data set and for a given item ID. It is used by the "DoInvoice" operation |
| **BanktransferReceived** | Checks whether a transfer to the shop's bank account was received or not |
| **CheckSchufa** | Checks whether customer's bank account is solvent or not |
| **CheckMasterCard** | Checks whether customer's MasterCard is solvent or not |
| **CheckVisaCard** | Checks whether customer's Visa card is solvent or not |
| **CheckAmericanExpress** | Checks whether customer's American Express card is solvent or not |
| **WithdrawFromCreditCard** | Used to withdraw money from a specific customer's credit card. |
| **WithdrawFromBankAccount** | Used to withdraw money from a specific customer's bank account |

The Web Services operations that simulate the real financial institutions were implemented as dummy external services. The business case Web application includes the Visa, MasterCard, American Express, and bank debit services. The credit card operation enables the shop to check whether a customer's credit card is solvent. It enables the shop to withdraw money from a customer's credit card. The bank debit operation enables the shop to directly withdraw money from a customer's bank account (direct debit). It checks whether a bank transfer is received by a customer (advance payment) and checks whether the bank account is solvent. The process of withdrawing money from a customer's bank account is depicted in Figure 7.28.



Source: Figure 26 in **(Denker et al., 2011, p. 42)**

**Fig. 7.28:** Payment using an External Web Service

As shown above, the coordination workflow from the coordination component calls the external payment Web Service operation to make the withdrawing process from a customer's bank account. The coordination workflow passes the customer's name, account and bank numbers, the withdrawal amount and a reference string as input parameters to process the payment.

#### 7.3.3.5 The Shipment Component

The shipping component simulates the services provided by the external shipping companies like DHL Express or Hermes. Its main responsibility is to inform these external shipping companies to pick up ordered packages and deliver them directly to the customers' delivery addresses. This component provides only one Web Service called: "CreateShipment". This service implements shipment functions. It expects a customer's first name, surname, and delivery address besides shipping type (DHL or Hermes) as input parameters for its operation. Based on these input parameters, the service generates a GUID[71] shipping code that identifies the pick-up number for a specific order. For each order, the service returns a shipping code and a tracking link back to the customer.

#### 7.3.3.6 The Mail Component

The mail component provides a Web Service to send notifications via emails to the registered customers' email addresses. These notifications are related to customers' orders.

[71] GUID or the Global Unique Identifier is used in implementing the shipping Web Service to make sure that each shipping code is totally unique and never repeated.

The implemented Web Service is called “SendMail” and it expects three input parameters: the recipient’s email address, the subject and the body of the message. This Web Service gets the values of the input parameters from the coordination component to constitute the emails that will be sent to the customers. The implemented version of this Web Service sends the message as text only. Other formats could be easy added when desired. Due to security considerations, this component is only allowed to be called from the other business case’s internal components. What is also needed to make this service functioning properly is to set the internal system’s email configuration. The configuration consists of the from email address, the email’s password needed to send an email via SMTP[72], the SMTP server, its port besides the option to use an SSL[73] encrypted connection. Table 7.12 below shows the configuration used in implementing the mail component.

**Tab. 7.12:** The Configurations for the Mail Component

| Variable | Value |
|---|---|
| **From Email Address** | vlbaws@googlemail.com |
| **Password** | ******** |
| **SMTP Outgoing Mail** | smtp.googlemail.com (use authentication) |
| **Port for TLS/STARTTLS** | 587 |
| **Enable SSL** | True |

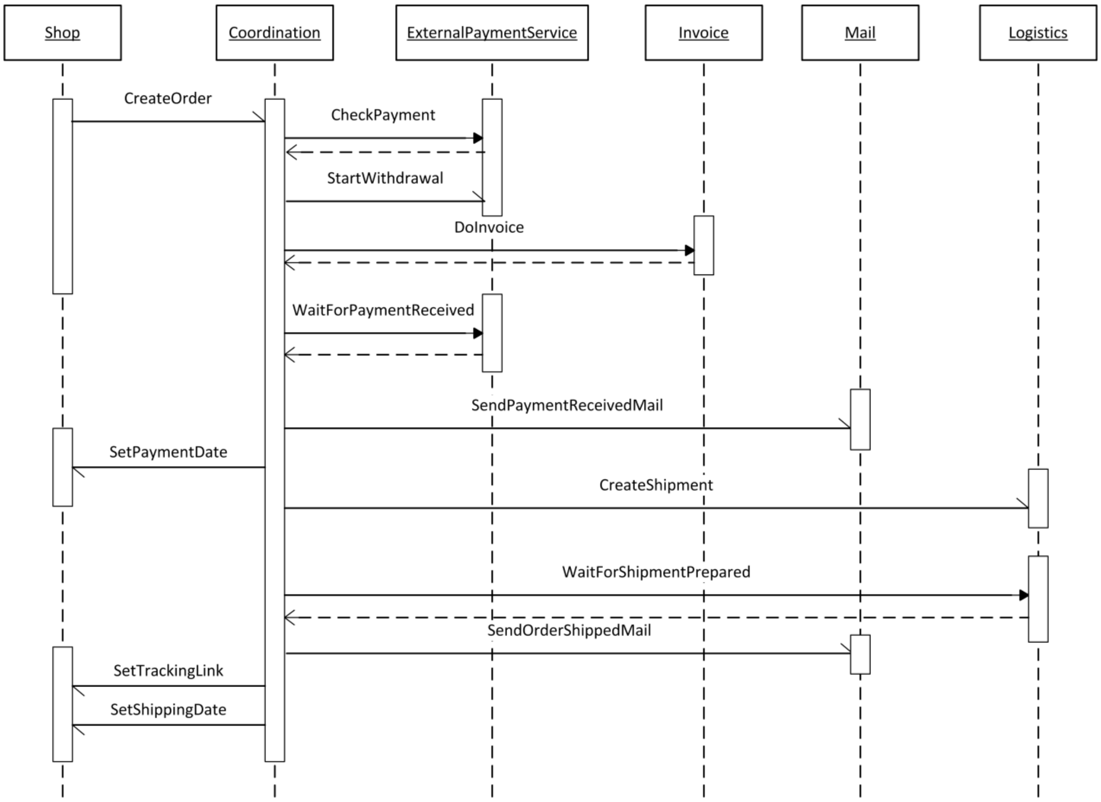
One last thing to be mentioned is that the aforementioned configurations are hardcoded within the implementation of the mail component. However, it could be easily stored and loaded from any kind of configuration files like “.INI” or “.XML” files, or even from a database.

### 7.3.3.7 The Coordination Component

The coordination component is the central component in this business case. The coordination component includes a long-running workflow called “Coordination”. This workflow actually coordinates the payment and shipment processes, returns the status updates to the shop interfaces and notifies the customers about updates using the mail component’s functions. The main interconnections among this component and the other business case’s components are shown in Figure 7.29.

[72] SMTP or the Simple Mail Transport Protocol is the protocol used for transferring the online shop emails via the Internet.

[73] SSL or Secure Socket Layer and its successor, TLS or Transport Layer Security are security protocols to secure the communication over the Internet. In this work, the SMTP configuration enables the use of TLS and SSL cryptographic protocols.



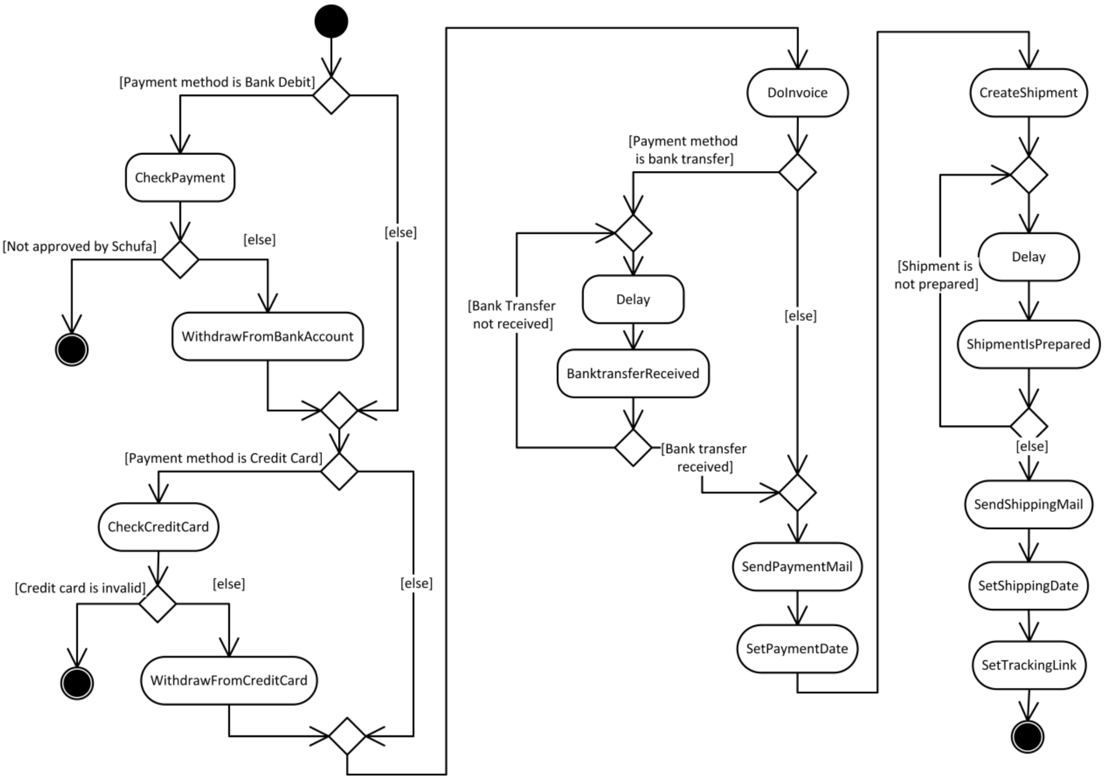
Source: Figure 5 in **(Denker et al., 2011, p. 17)**

**Fig. 7.29:** Component Interactions in the Selling Business Process

The shop component sends asynchronous "CreateOrder" message to the coordination component. This latter communicates with an external payment Web Service to check the validity of the customer's payment data. If this checking is successful, the coordination component asks the invoice component to issue the invoice and waits for acknowledgment from the payment service. The coordination component then asks the mail component to notify the customer that the payment was successful and creates a pickup request to be sent to the logistics component. Upon successful product shipment, the coordination component asks the mail component to notify the customer that the shipping process started providing the tracking link and the shipping date at which the ordered product has been shipped.

The "Coordination" workflow and the other system's workflows have a connection to the SESOA semantic service repository using the "Twinkle" query tool. They firstly have to know the assemblage in which the desired Web Service is registered. This is easily done by discovering the assemblages in the SESOA main Web application GUI (see Section 7.2.1.2). After that, "Twinkle" tool is used as described before in Section 7.3.2 to apply queries to the SESOA repository's URL (in which the RDF statements are published) to retrieve the Web Services' information. The "Coordination" workflow has been implemented as a workflow service using WF 4.0. This means that the workflow itself can be discovered and invoked as a Web Service. Moreover, it can be registered in an assemblage and published in the SESOA semantic service repository.

Most of the activities in the system’s workflows have been realized as Web Services as well. This method of implementing this business case using a mixture of workflow services and Web Services has been decided to take as much benefit from the SESOA prototypes as possible[74].



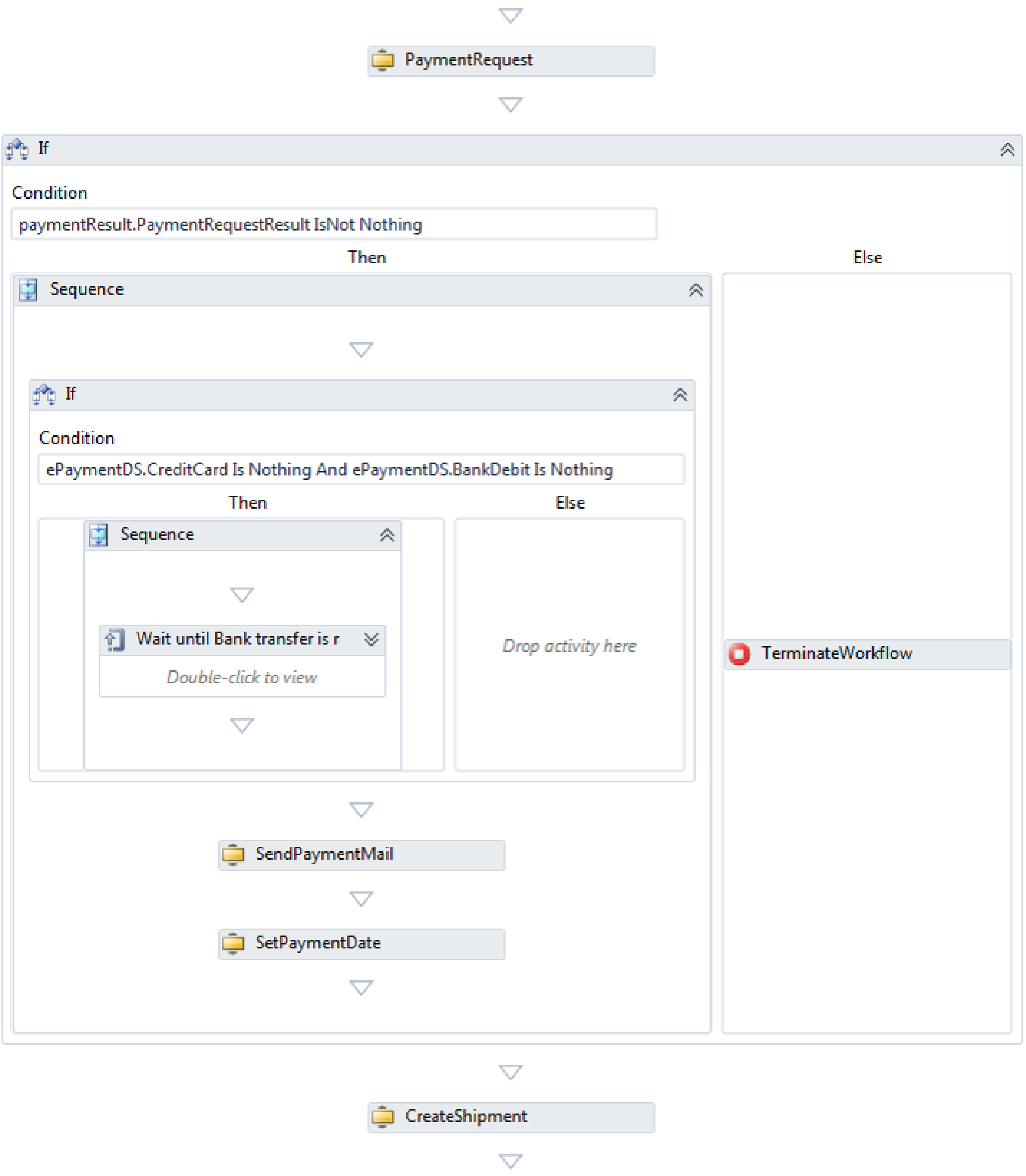
Source: Figure 6 in **(Denker et al., 2011, p. 18)**

**Fig. 7.30:** Coordination Workflow

The activities of the “Coordination” workflow are depicted in Figure 7.30. This workflow starts by checking the payment details provided by the customer. The already implemented prototype enables either bank transfer or credit card payment. If the payment data provided by the customer for one of these two methods are wrong or invalid, the workflow stops executing. Otherwise and based on the payment method, the invoice will be created and sent to the customer. Upon payment, the customer will be notified that the payment process was successful. Finally, the workflow initiates the shipment process and notifies the customer as soon as this process is finished.

Upon placing an order, the first action to be done before executing the “Coordination” workflow is collecting the order’s different business objects: address, shipping type, bank debit, credit card, customer info, item, payment type, payment service, price, purchase order, and shipping.

[74] The most important advantage from this decision is to achieve the highest degree of system reusability.



Condition
paymentResult.PaymentRequestResult IsNot Nothing

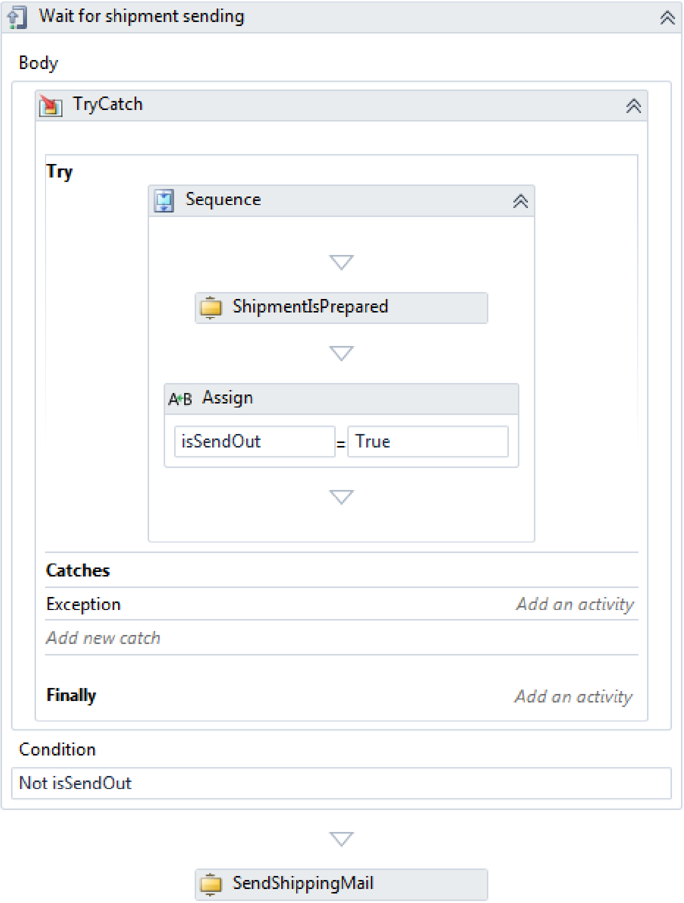
Condition
ePaymentDS.CreditCard Is Nothing And ePaymentDS.BankDebit Is Nothing

Source: Figure 18 in **(Denker et al., 2011, p. 34)**

**Fig. 7.31:** Part of the Coordination Workflow - Payment Notification

Figure 7.31 depicts a part of the WF “Coordination” workflow that is responsible for the payment notification. Before that, the first step in this workflow is to check the payment method of the current order needed to collect the money from the customer in order to pick up the order for shipment. Afterwards the “PaymentRequest” custom activity is invoked (see the figure above). This activity is imported from the payment component and returns a payment object. Based on this object the workflow is able to notify the customer about the receipt of the money. Customer notification is done through the invocation of the “SendPaymentMail” custom activity. This activity communicates with the mail component and sends an email to the customer including the payment details. The “SetPaymentDate” custom activity imported from the shop component is then invoked to update the payment date in the database. After executing this activity, the customer is able to see the payment information in the customer center GUI of the website.

Subsequently, the next activity to be invoked in this workflow is the “CreateShipment” activity imported from the logistics component.



isSendOut = True

Condition

Not isSendOut

Source: Figure 19 in **(Denker et al., 2011, p. 35)**

**Fig. 7.32:** Part of the Coordination Workflow - Shipment Notification

Figure 7.32 shows basically how the WF “Coordination” workflow waits for the response of the “CreateShipment” custom activity. Within a while loop, the “ShipmentIsPrepared” activity imported from the logistics component is invoked. This activity checks the current state of the order shipment and it gets called repetitively until it returns that the package has been sent. Afterwards, the workflow continues executing by sending a shipping notification to the customer using the “SendShippingMail” activity imported from the mail component. This step represents the end of the coordination workflow and thus the end of the selling business process.

## 7.4 Evaluation

Based on the research methods presented in Chapter 4, this section gives the final discussion about the SESOA concept evaluation in different business domains. The last two phases in that research process are the evaluation and communication. This section gives an overview of how SESOA concept has been applied on an industrial scale.

Moreover, this section also illustrates how the concept has been communicated with other researches as part of the evaluation process.

Besides the implemented business case (see Section 7.3), SESOA approach has been applied to Corporate Environmental Business Information Systems (CEMIS) and on-demand business intelligence research domains. Furthermore, a workshop had been conducted in the company "CeWeColor[75] AG & Co." to potentially apply the SESOA prototypical implementation in one of its organizational units. However, before evaluating the approach in the aforementioned business domains, set of questions need to be answered:

- What requirements are already designed, covered and implemented?
- What artifacts in the system have particular importance?
- How does the prototype change from its previous versions?
- Which elements of other partial subsystems are affected by changes on a particular artifact?
- What percentage of completion for the implementation can be adopted based on outcomes-objectives relations?

The following subsections try to answer these questions and depict how SESOA approach can be evaluated in different business domains.

### 7.4.1 Corporate Environmental Management Information Systems

This section illustrates the applicability of the SESOA approach to CEMIS research field from both practical and scientific points of view. The concept is applied to CEMIS project called: "IT-for-Green". It is also communicated to another project called: "Organizations' Environmental Performance Indicators" and to collaborative CEMIS in general. The following three subsections introduce the projects and the ideas behind collaborative CEMIS and shows how SESOA concept is applied to or can be applied to them.

#### 7.4.1.1 IT-for-Green

All the details described here in this subsection are derived from the publications: (Rapp et al., 2011; Marx Gómez et al., 2011; Gräuler et al., 2012, 2013). IT-for-Green (Next

[75] http://www.cewecolor.de/de/home.html

Generation CEMIS for Environmental, Energy and Resource Management)[76] is a funded by the European Regional Development Fund (grant number W/A III 80119242) in the period from April 2011 till October 2014.

This project is considered related to the CEMIS research domain. It aims at establishing open source software modules that are able to deal with current stakeholder demands. Moreover, these modules have to be flexible and adaptable for any of these stakeholders' future demands. The core approach behind this project is developing a Web-based software using service-oriented platform that handles the generation process of structured reports besides supporting the required company-specific business processes in this software. This project represents actually one of the ideal places to prove the applicability of SESOA approach.

The resulted system from this project will integrate three modules that map any product life cycle from input (energy efficiency measurement), going on with transformation processes (production and green logistics), ranging up to the output side (company communication and sustainability reporting) (Rapp et al., 2011, pp. 574–579; Marx Gómez et al., 2011, p. 19). A previously accomplished feasibility study has already illustrated a list of requirements to achieve this (cf. Teuteberg & Marx Gómez, 2010).

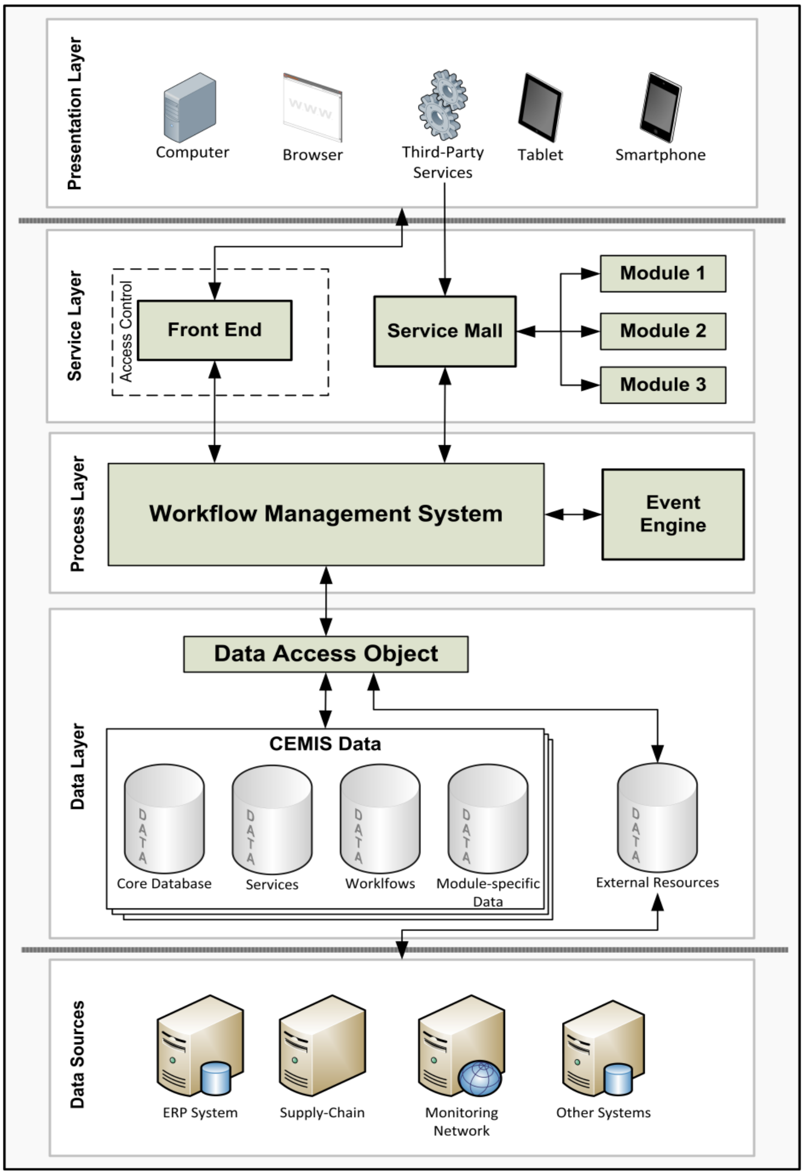
Similar to SESOA layered architecture illustrated in Chapter 6 (Section 6.1.1) the counterpart initial architecture of CEMIS next generation that will result from the IT-for-Green project has been designed. This architecture is depicted in Figure 7.33.

This figure shows that the CEMIS layered architecture is composed of five layers. These layers are: presentation, service, process, data and data sources layers. The presentation and data sources layers do not belong directly to CEMIS architecture and therefore they are detached by hatched lines in the above figure. Information flows and function calls are indicated by the directed lines. The presentation layer primarily represents the system's GUI by which users can login to the system via different devices (computers, tablets, smart phones…). User authentication is required while logging in to the system. This authentication is part of the access control located in the service layer. Based on user's rights, a list of workflows will be provided. These workflows are entirely managed by the workflow management system (the process layer). Some details about CEMIS workflows are explained in (Bremer, Mahmoud, & Rapp, 2012).

Similar to SESOA, many workflows' activities are implemented using Web Service technology. The service mall in Figure 7.33 belongs to the service layer and represents a traditional Web Service directory in which Web Services are published and discovered by service consumers. The step of executing workflows is initiated by the user. Afterwards, the workflow management system discovers the service mall to fetch a list of Web Services required to execute the workflow. These Web Services are provided ei-

[76] Information about the IT-for-Green project is available online at: http://www.it-for-green.eu

ther externally by authorized service providers or internally by the system's modules (located in the service layer).



Source: Figure 3 in **(Gräuler et al., 2013)**

**Fig. 7.33:** CEMIS Next Generation Layered Architecture

As can be seen in the above figure, CEMIS next generation has three modules:

- Module 1 or the "Green IT": This module captures and automatically measures energy consumption of different ICT[77] realized by data centers;

[77] ICT stands for Information and Communication Technologies.

- Module 2 or the “Green Logistics and Sustainable Product Development”: This module provides functions for automatic determination of $CO_2$-emissions along the supply chain;

- Module 3 or “Sustainability Reporting and Communication”: This module sets up a Web-based reporting solution using the SOA paradigm.

Authorized users in each of the abovementioned modules design and implement different domain-related Web Services and applications in a straightforward manner. This means that, each module has embedded domain-specific knowledge that is managed by the domain experts (Gräuler et al., 2013).

The workflow management system uses the services developed by the abovementioned modules. As can be seen in Figure 7.33, the workflow management system is connected to the so-called: “Event Engine”. The event engine represents the system’s monitoring tool by which different indicators can be calculated to capture the status of different tasks of the system’s modules. Since this engine is not thematically related to SESOA concept, it will not be further explained. However, further details about the event engine are explained in (Rapp & Bremer, 2012).

Via data access object, the workflow management system is connected to the CEMIS database system. This database system composed of the CEMIS databases namely: the core database, module-specific, services, and workflows. Moreover, the system can be also connected and integrated to other external resources like ERP, SCM… via a data access object too.

SESOA reference architecture had been investigated to choose the proper components that can be transferred to the CEMIS next generation architecture. After defining the general requirements for the IT-for-Green project, it has been decided to adapt some of the components of the SESOA reference architecture’s to design the project’s service-oriented infrastructure. These components include: the consumer system, the provider system and the service repository (the green service mall). Moreover, a workflow management system will be applied to the CEMIS next generation. The difference here from SESOA is that an open source workflow system will be applied to the CEMIS next generation instead of the Microsoft Windows Workflow Foundation utilized in SESOA. It had been decided also that all the business processes included in the resulted software from the project will be implemented as workflows with environmental context. These workflows are designed to include activities implemented as Web Services. These Web Services will be published in the system’s repository that is called “Service Mall”.

As a conclusion, SESOA components had been seen as the basis to create the project’s service-oriented infrastructure that integrates its three modules. This resulted system from the IT-for-Green project enables its users based on their rights and roles to access its green service mall to discover and invoke its Web Services. The user management has been considered as a clear requirement in designing the system’s database schema.

The semantic part of the SESOA reference architecture is not seen as part of the prototypes resulted from the project. However, the RDF semantic enrichment of the resulted software from the IT-for-Green project can constitute an outlook for a potential extension of the system.

#### 7.4.1.2 OEPI

The details described here in this subsection are derived from the publication: (Meyerholt, Mahmoud, & Marx Gómez, 2011) that tried to find the link between SESOA concept and the European financed OEPI project. This section will firstly introduce the OEPI project and then gives a brief explication of how SESOA concept can be applied to it.

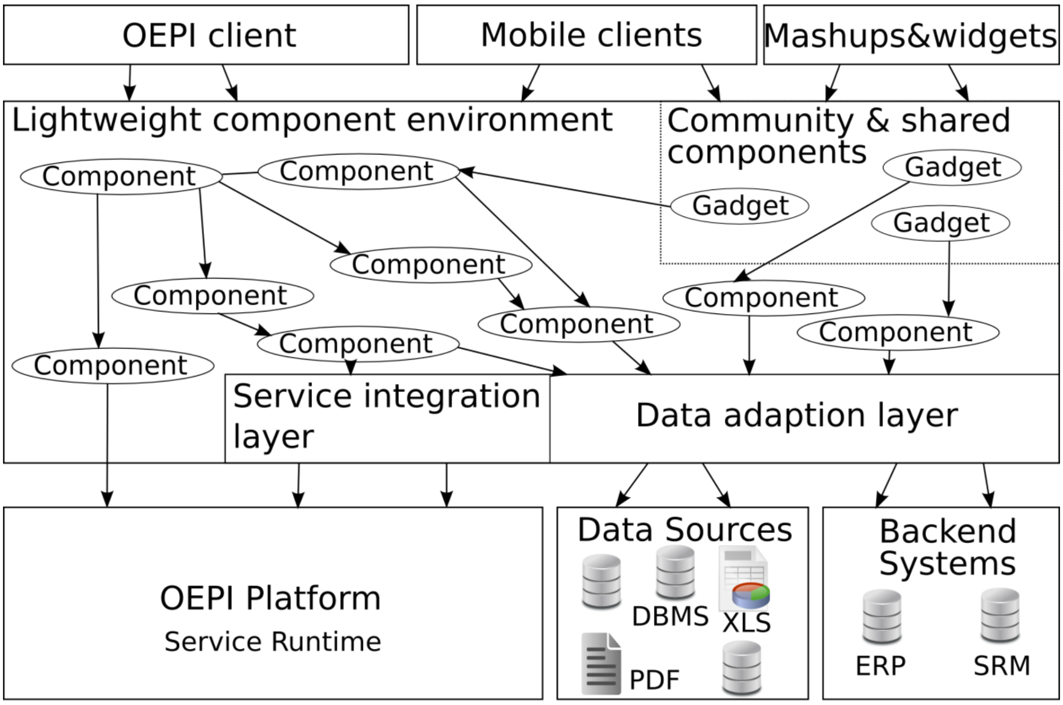
Since most sustainability reports are handled in a one shot process to produce an annual report for example, the environmental information managed by a CEMIS stays normally unused in the daily business tasks improvement (e.g. production or the procurement process of businesses). To overcome this shortcoming, the EU funded the research project “Solution and Services Engineering for Measuring, Monitoring and Management of Organizations’ Environmental Performance Indicators” (OEPI) project[78]. This project aims to improve the traditional CEMIS by e.g. reintegrating Environmental Performance Indicators (EPIs) into enterprise systems. Doing that improves directly the environmental impact of the product design or the procurement process by improving the overall environmental footprint of a company (Meyerholt et al., 2010).

OEPI architecture relies upon the SOA approach and is composed of many separate components and subsystems. This architecture is shown in Figure 7.34. As can be seen from the figure below, there are three different types of clients that utilize different types of components. These clients are ranging from the OEPI client application to mobile clients to mashups & widgets and the components are exposed as Web Services. They are either directly embedded into the OEPI platform which serves as the Service Runtime or they access the OEPI platform through a service integration layer. What is important to observe here is the data adaption layer. This layer is responsible of accessing different types of data (mainly EPIs) that are coming from various data sources. Therefore, the data adaption layer represents a bridge that can be utilized by OEPI’s components to arbitrary access underlying data sources.

The exchanged data are mainly EPI-based data. However, it can also be in structure forms like for example hierarchies, stakeholder information or even documents. It has been suggested in this project to create a mediator to deal with this diverseness of information in terms of data access and interpretation. This mediator serves as the main component in the data adaption layer (see Figure 7.35). The overall design in this pro-

[78] More details about OEPI project are available online at: http://www.oepi-project.eu/

ject is largely influenced by the structure of an ontology used for describing the EPIs main aspects.

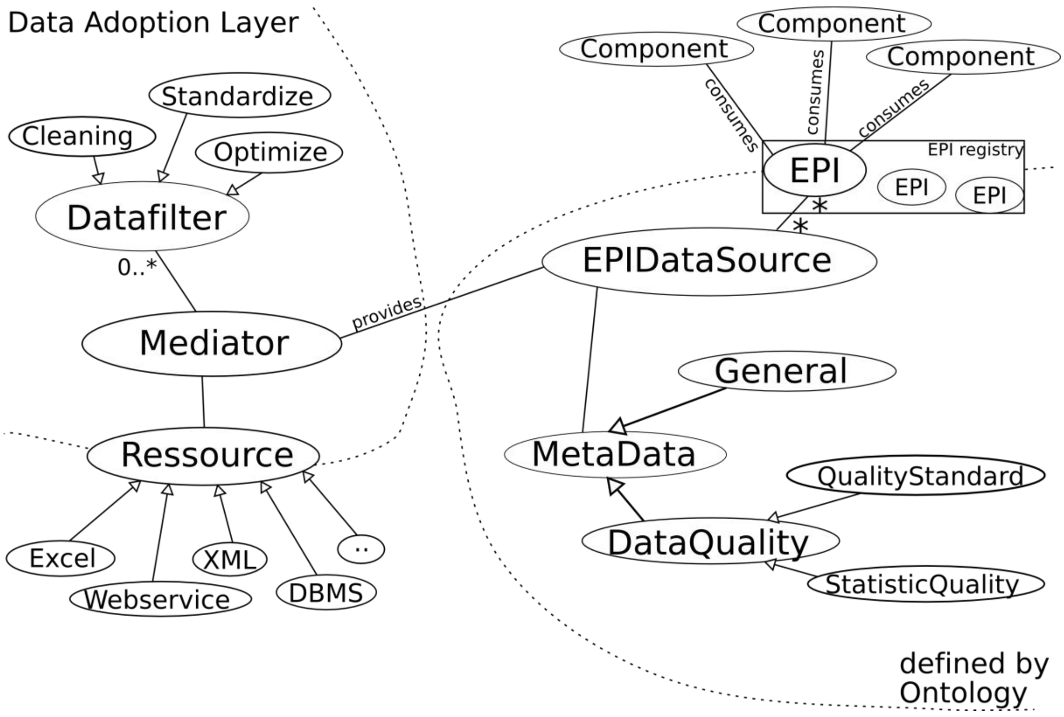


Source: Figure 1 in **(Meyerholt et al., 2011, p. 303)**

**Fig. 7.34:** Overview of OEPI's Architecture

The figure below depicts the data application layer in the project. OEPI components are represented in the upper right corner. These components just request to consume a specific EPI (and its associated data structures) from an EPI registry. The registry looks up the desired information needed to create a specific request to the data adaption layer that provides the EPI in the backend. Moreover, this registry may provide a thin layer of caching in order to reduce the traffic needed to retrieve the EPI again from its sources. It is advisable to distinguish EPIs using a unique RDF Identifier. This RDF tuple should be attached to the requested EPI in the EPI registry to have unique mapping (Meyerholt et al., 2011, p. 304). This is another place where SESOA concept can be used.

As can be seen in Figure 7.35, the EPIs, its data sources, and the accompanying metadata will be described by the OEPI's core ontology. Using the EPI's related RDF tuple, the mediator component is able to lookup how to access the specific EPI data by having its own resource repository. The mediator then decides how to handle that specific type of resource. The access to the data itself must be handled in a plug-in manner for each type of resource. For example, such a plug-in may provide the mediator with extracted data values of: a document (e.g. PDF file), an Excel spread sheet, databases in general, or Web Service provided data.



Source: Figure 2 in **(Meyerholt et al., 2011, p. 304)**

**Fig. 7.35:** The OEPI’s Data Adaption Layer

Another approach had been exploited in the data adaption layer is the application of a data filter chain to the extracted data. A variety of data filters can be applied to the EPI values. These filters may include:

- Data Cleaning: This filter is used to remove erroneous or unneeded data;
- Data Standardizing: This filter is used to recalculate the gathered values to have for example a common unit or to standardize the percentage values to values between from 0 to 1;
- Data Optimization: This filter is used to complete tasks like for example sorting data or reordering them to improve the performance in the later processes.

Since there is no common agreement in handling environmental performance indicators across available software systems, OEPI project leverages achievements of the field of Semantic Web technologies to overcome this shortcoming. This is exactly the place where SESOA reference architecture can be seen and considered as a potential architecture for delivering semantics to handle such a shortcoming.

As mentioned before in this section, most of the EPIs in the OEPI project are realized as Web Services. From the service provider’s perspective, these services can be registered in desired assemblages in SESOA framework to constitute RDF statements. This means that the relations between the EPIs services and the SESOA assemblages will be semantically annotated. On the other side and from a consumer’s perspective, the objects included in the Web Service request will be stored in semantic dictionaries that are linked with the assemblages using similar RDF statements as the one used by assemblages-EPIs relations. The matchmaking process between the request objects and the service capabilities is done using the RDF annotations between the "semantic dictionary - assemblage" and the "assemblage - Web Service" relations. The result will make the proper data source available to the consumer if these sources are available.

One last thing to be mentioned here in this section is that the evaluation of SESOA in OEPI project is considered more as research-oriented evaluation. The details explained in this section were communicated to the scientific community as a peer-reviewed paper in the 5th international symposium on Information Technologies in Environmental Engineering ITEE 2011[79] that took place in Poznan, Poland from 6-8 of July 2011.

#### 7.4.1.3 Collaborative CEMIS

All the details described here in this subsection are derived from these two publications: (Allam, Mahmoud, & Marx Gómez, 2011; Allam, Mahmoud, Marx Gómez, et al., 2011). The main goal behind the collaborative CEMIS is firstly to overcome the shortcoming of having rare use of CEMIS in the German-speaking region and secondly to strengthen the efforts of having collaborative work among companies to gain the benefits of industrial symbiosis. Being based on Web Services, collaborative CEMIS provides powerful integration of environmental applications and data. Instead of being connected to several dissimilar environmental applications, Web Services-enabled collaborative CEMIS solution provides a medium by which these heterogeneous applications can communicate.

Before explaining how SESOA can be evaluated on collaborative CEMIS, brief introduction about this latter is quite necessary. A collaborative CEMIS concept is exemplary developed to control the usage of hazardous materials. Based on the German law[80] on chemical substances, enterprises that deal with hazardous materials have to provide preventive measures and up-to-date information about it. To achieve that, a register of hazardous materials is required. The important data needed are:

- The storage place and quantity
- The utilization place
- Instruction for use[81]

[79] More information about this symposium is available online at: http://www.itee2011.put.poznan.pl/

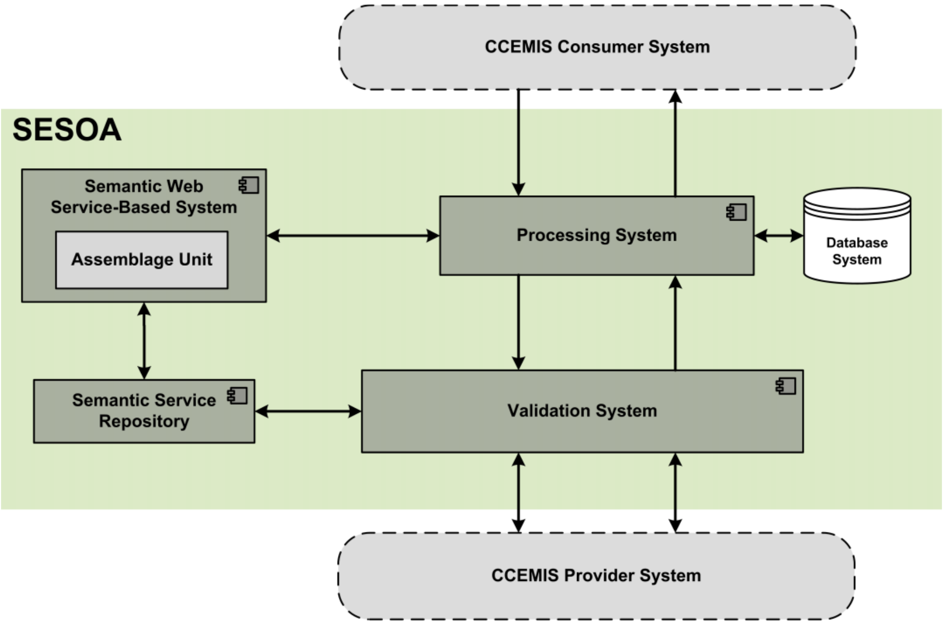
[80] Further details can be found in the German Hazardous Material Law, Paragraph 14-3-B.

[81] The Instruction for use is normally delivered as a document. This document should be checked at least once every year.

- Risk assessment[82]
- The danger classification for human, animals, and environment.

Every company should normally regulate the storage and utilization information. All other data are normally company independent. Therefore, a network of companies can give an effort to manage this. To make the idea more clear assume that three companies work with some hazardous materials. Instead of the situation that each company updates its data alone, applying a central data storage approach helps to reduce the efforts to manage the data that must be updated yearly. Moreover, it gives these companies better opportunities to exchange information (Allam, Mahmoud, Marx Gómez, et al., 2011).

One possible way to realize the collaborative CEMIS concept is to realize most of its functionalities as Web Services and orchestrate these services in form of workflows. For this purpose SESOA sounds a good candidate. Figure 7.36 represents the realization of Collaborative CEMIS (CCEMIS) using SESOA system. By relying on the workflow system (processing system), the market best practices can be used by storing workflows in the database system. The workflow system is linked to the collaborative CEMIS consumer and provider systems. Consumer requests' can be annotated as RDF objects.



Source: Figure 1 in **(Allam, Mahmoud, & Marx Gómez, 2011, p. 186)**

**Fig. 7.36:** Collaborative CEMIS Realization using SESOA

The services supplied by the collaborative CEMIS providers are registered in desired assemblages in the assemblage unit. These assemblages are annotated as RDF subjects. The assemblage unit links these subjects and objects to compose RDF statements. These

[82] Risk assessment is normally delivered in form a work sheet (document). It should be also checked at least once every year.

statements are then published in the SESOA semantic service repository and made available to all providers.

Using SESOA concept in collaborative CEMIS can enable the sharing of bigger amount of data taking into consideration that the identities of the collaborative CEMIS players must be hided if requested. In this way, these players can hide their identity or reveal it depending on their policies and the sensibility of their data values.

The evaluation of SESOA in collaborative CEMIS is also considered more as research-oriented evaluation. The details explained in this section were communicated to the scientific community as two peer-reviewed papers. The first paper: (Allam, Mahmoud, & Marx Gómez, 2011) was presented in the 5th international symposium on Information Technologies in Environmental Engineering ITEE 2011 that took place in Poznan, Poland from 6-8 of July 2011. And the second paper: (Allam, Mahmoud, Marx Gómez, et al., 2011) was presented in the EnviroInfo 2011[83] conference that took place in Ispra, Italy from 5-7 of October 2011.

### 7.4.2 On-Demand Business Intelligence

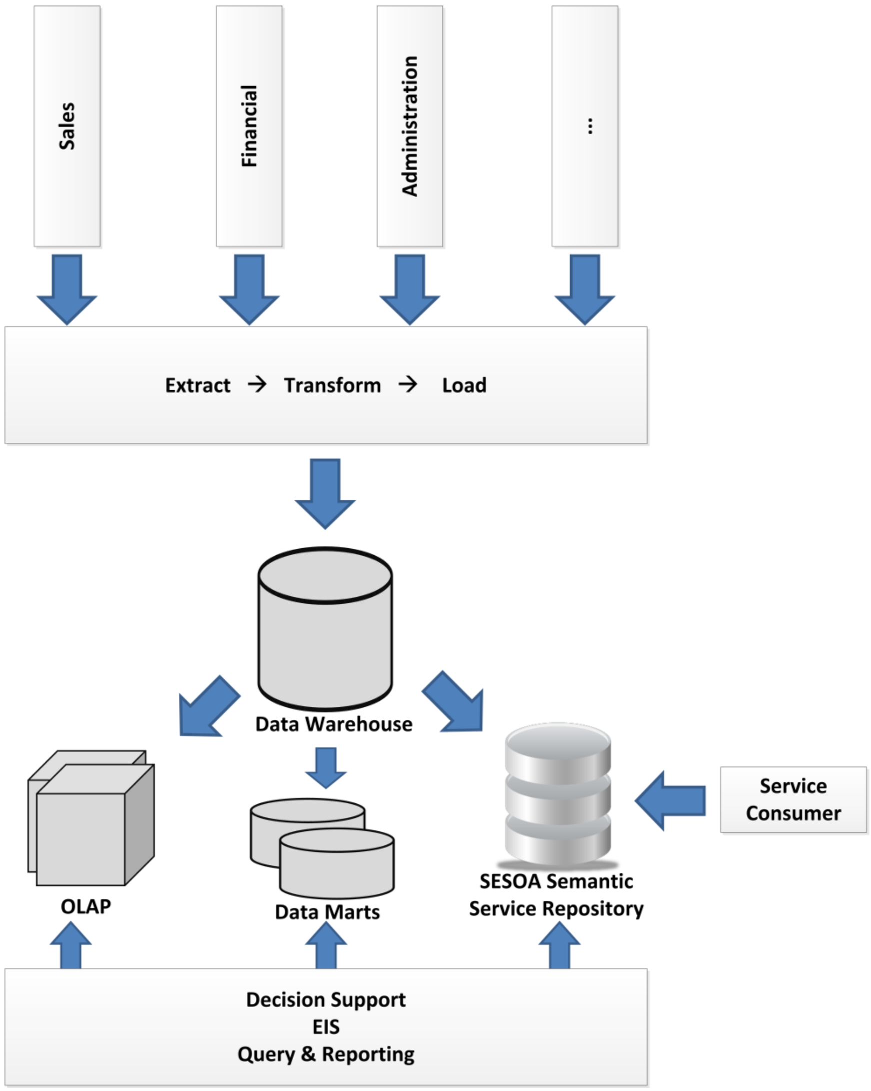
The details on how SESOA can be evaluated in the on-demand business intelligence systems are derived from the publication: (Mahmoud, Marx Gómez, et al., 2012). Before showing how SESOA can be evaluated in business intelligence domain, some words about this latter are necessary. The current value of Business Intelligence (BI) for companies is generally affected by two main improvements namely: the improvement of process management and the improvement of operational processes.

BI systems are generally considered as an extension of a data warehouse that transfers data from an operative system using online transaction processing (OLTP) towards a decision support and reporting system via online analytical processing (OLAP). The main idea behind BI constitutes a generic concept and a strategy that aims at supporting the following fields: technology, users, and expert knowledge (cf. Marx Gómez, Rautenstrauch, & Cissek, 2009, p. 13).

This section presents a linkage between BI to service-oriented architectures in general and SESOA specifically. BI applications can be considered as service consumers to the services published in the SESOA semantic service repository. Therefore they can discover, select and invoke all services supplied by external providers published in this repository. In this way, SESOA can fill the gap between SOA and BI concepts to enable the delivery of the “on-demand” data as services in real time. These services can be supplied from companies of any size. This can open the BI market to include SMEs

[83] Information about this conference is available at: http://www.ec-gis.org/Workshops/EnviroInfo2011/

besides huge enterprises to be potential resources of such services. The BI architecture can be enhanced using SESOA semantic service repository as depicted in Figure 7.37.



Source: Figure 2 in **(Mahmoud, Marx Gómez, et al., 2012)**

**Fig. 7.37:** On-Demand BI enhanced by SESOA

By integrating SESOA semantic service repository in BI architecture the main potential data processing can be facilitated using the available Web Services published in this repository. Services can be discovered in this repository and eventually invoked whenever there is a need for a resource not available internally. Moreover, Web Services published in this repository are annotated with RDF statements so that the proper data can be retrieved at the right time to the right user. As described many times in this work, the semantic annotation is applied to the relations between services and assemblages to which they belong. Using SPARQL as an RDF query language, the BI application can query the repository's RDF statements to find the proper service based on the domain to which it belongs (business domain). As a result, this will enhance the BI decision support process by enabling another service classification level (the semantic allocation of services) besides the conventional service discovery techniques.

Data in BI systems are not just stored. Rather, the data must be collected, administered, filtered, analyzed, and controlled. This whole data management chain requires high cost of ownership in terms of hardware (storage sizes, network, etc.), software, and last but not least human resources. For many businesses (especially start-ups and medium-sized), these costs represent a real challenge. By introducing the BI-enhanced SESOA system, SMEs can make use of data that can be retrieved from multiple external resources without the traditional BI limitation of using just their in-house data.

Semantic service-enabled BI infrastructure will lead both SOA and BI concepts to a new era. IT-for-Green project (see Section 7.4.1.1) represents a perfect place for the realization of the BI-enhanced by SESOA concept. This project links analyzing and reporting of environmental information via Web Services. One potential added value from employing on-demand BI concept in this project is to administer the environmental sustainability reporting using BI functionalities. This will enable the processes of analyzing and generating EPIs to be placed in sustainability reports. Such reports are normally based on social, economic and ecologic aspects. The calculation of EPIs can be mainly done with the help of the published Web Services in the SESOA semantic service repository. All external service providers can offer their Web Services to be registered in this repository’s assemblages. This will enhance the usability of these services by enabling other applications (in this case, the BI ones) to use it in a collaborative manner. Examples of such services include: energy efficiency measurement, carbon footprint measurement, eco balancing creation services…

To sum up, the main outcomes that can be harvested from merging BI and SESOA concepts can be seen as follows:

- Opening the BI market to include SMEs as potential “on-demand” data sources so that they don’t only use their in-house data, rather any kind of available data supplied by external providers.

- Supporting the semantic annotation of BI-related Web Services’ relations as it is applied in the SESOA prototype.

- Enriching BI ad-hoc reports with the “on-demand” data. These reports can also be provided as Web Services published in the semantic service repository. This will reasonably enhance the reusability of these services.

Last but not least, the evaluation of SESOA in BI domain is considered research-oriented evaluation as well. The vision mentioned in this section was communicated to the scientific community as a peer-reviewed publication: (Mahmoud, Marx Gómez, et al., 2012). This paper was presented in the 20th European Conference on Information

Systems, ECIS 2012[84] conference that took place in Barcelona, Spain from 10-13 of June 2012.

### 7.4.3 CeWeColor AG & Co.

Another possible evaluation of the SESOA concept was the potential application of the resulted prototype in the “CeWeColor[85] AG & Co.” company. The CeWeColor AG & Co. OHG is a company located in Oldenburg, Germany. Its main revenue is with the production of digital images, CeWe Photo Books, personalized gifts, analog film development and online printing. This company is the Europe's largest photo finisher. The company has 12 photofinishing operations and over 2000 employees.

Since the company has already different applied information systems, it has been decided to conduct a workshop in the company to discuss the possibilities of applying the SESOA prototypical implementations in one of its organizational units. Managers and software developers were invited to this workshop. It was decided to have different key staff from the company to have more deep discussion of the potential use of SESOA resulted prototype and how it can be applied in the company.

The invited key persons to the workshop were namely as follows:

- Dr. Joachim Marz: The leader of IT department
- Dr. Peter Hartz: The leader IT Infrastructure department
- Mr. Herbert Nase: From the software development department
- Mr. Manfred Neugebauer: The leader of software development department
- Mr. Josef Tapken: from the production system department

The workshop was divided into two parts: the theoretical and the practical parts. In the theoretical part, the SESOA reference architecture had been presented together with short presentation of the used background information including SOA, Web Services, and semantic annotations. Moreover, the main harvested outcomes behind SESOA had been listed. After a short break, the practical part of the workshop took place. In this part, the prototypical implementations resulted from this work had been demonstrated. That included the main SESOA Web application with all its features including how to manage assemblages and Web Services and how to administer the semantic service repository. The implemented selling business process had been also demonstrated with more insights of how it had been built on top of the SESOA concept.

[84] More information about ECIS 2012 conference is available online at: http://www.ecis2012.es/

[85] http://www.cewecolor.de/de/home.html

The discussion round took place after the presentation sessions of the abovementioned two parts. The comments from the invited key staff were as follows:

- The company is having on its future plans a possibility to adapt new solutions like the one presented in this workshop. However, due to time and budget limitations, this couldn't be done directly.

- There is a clear requirement to introduce semantics to the internal company's data. However, they have just one central data repository and all the semantics have to be applied to the data stored in this repository.

- The SESOA concept is good related to the major set of products the company provides. This means that the possibility of applying SESOA concept to the internal departments is fairly needed.

- There was also a discussion of how such concept can be applied on a larger business model than the one presented in the business case. This means how to make revenue of applying SESOA concept in the company. The answer was that the company can develop its services to make more profits on an international scale based on the Web Service technology.

- It has been agreed upon that the introduction of such new concept needs to be accompanied with proper training of the main personnel who will interact with its prototype.

Finally, this workshop had the conclusion that the best place where SESOA concept can be applied is the customer care service department. This means that the entire customer's complains and opinions will be grouped based on a specific internal classification in the SESOA assemblages. Different automatic responses will be also developed in form of Web Services to be registered in these assemblages. This is side by side with the automatic semantic annotation of the assemblages and Web Services relations. In this way, the semantic service repository can be used also in this department. What needs to be developed is a proper interface to this repository so that the requests of the customers can be directly routed to the proper assemblages and then to the exact services to provide the proper responses.

Other communication channels have been initiated with other companies inside and outside Germany that are seen as potential collaborates to validate the prototype resulted from this work. The requirements of such companies have to be taken into consideration to adapt the SESOA prototype to make it applicable within their departments.

## 7.5 Summary

In this chapter, the main SESOA implementation details had been presented. The main evaluation and design criteria together with the configurations of the adopted technologies were then listed. This chapter then gave detailed insights about the resulted SESOA Web applications. Firstly, the SESOA main Web application had been demonstrated with details of how to manage the assemblages, the Web Services, the semantic RDF-enablement of their relations, and the semantic service repository. Set of validation services had been implemented also in this Web application. The other explained Web application in this chapter was the one implemented to realize the accompanying business case. It explicated the main aspects of the ERP's selling business process and how this business process had been implemented on top of the SESOA concept.

The last section of this chapter was dedicated to the evaluation aspects of this work. Five evaluation possibilities had been discussed. Three of these evaluation possibilities were in the CEMIS research domain.

A potential application of the SESOA concept is to the business intelligence domain. The details of how such evaluation can be done were explained at the end of this chapter. The last section of this chapter was related to the practical evaluation of the SESOA approach on an industrial scale. It has been explained that a workshop had been conducted at CeWeColor AG & Co for this purpose.

The next chapter is the final chapter in this thesis. It sums up the main contribution that can be harvested from this work and try to give an outlook of possible future directions.

# 8 Conclusion and Outlook

This chapter is the final chapter of this work. It summarizes all the ideas, concepts and approaches presented all over this dissertation. In addition, it provides the summary of contributions and opens the direction for the future work directions that can be derived from this research.

The first section of this chapter summarizes the conducted research with its major contributions. The last section of this chapter gives some highlights on the potential future directions like the introduction of a security pattern and a proposal of Web Service recommendation system.

## 8.1 Research Summary

The research conducted in this work explains all the related aspects of the lightweight semantic-enabled enterprise service-oriented architecture. SESOA deals with semantically-annotated Web Services in an enterprise context and shows how these services can be applied to different business environments.

One of the main purposes of SESOA is to group Web Services in assemblages based on the actual domains to which they are related (business domains). The main outcomes that can be harvested from using SESOA are:

- Open the market to include SMEs
- The high reusability of its components where each component can be seen as a standalone reusable module
- Provision of a second classification level of Web Services based on the assemblage-Web Services relations
- The lightweight semantic annotation of Web Services relations using RDFS statements
- The creation of workflow host services
- Web Service validation and evaluation
- The advertisement for new Web Services

This set of contributions has been explained in details throughout this dissertation. Therefore, they have been listed here without lengthy details.

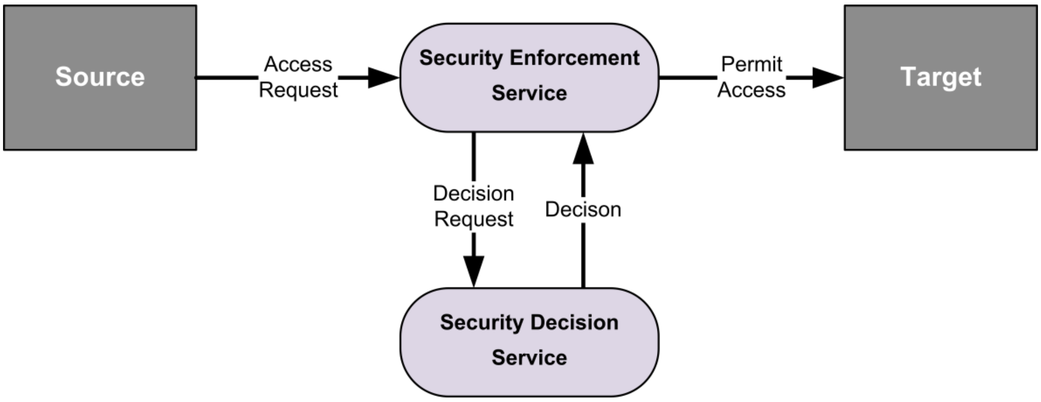
## 8.2 Future Work Directions

This section tries to give some ideas for future works that can be derived from the concept of SESOA. The main domain that is recommended to be researched in extending this work is security. Moreover, a Web Service recommendation system for business applications sounds also as an interesting research idea. These two potential research entry points are discussed in the next two sections. The last section of this chapter is dedicated to give a short overview of the conducted researches that were directly or indirectly based on this work and wrap up this dissertation.

### 8.2.1 Security Pattern

Managing security in SESOA is still an open point. At the moment, SESOA resulted artifacts enable just the embedded WS-Security extensions (Lawrence et al., 2006). However when individual business processes intend to implement and include their own security decision services, this will result in the duplication of information and administration among applications in the sense of identities and permissions.

One possible approach to unify the security support in SESOA is the usage of security as a service model. It supports applications and services that require security functions and don’t have them internally. International Standards Organization (ISO) had adopted this model (ISO 10181) and implemented it through a Policy Decision Point (PDP) and a Policy Enforcement Point (PEP). The equivalent SOA security as a service model is depicted in Figure 8.1. As can be seen in the figure below, the Security Decision Service (SDS) and the Security Enforcement Service (SES) are the counterparts of PDP and PEP respectively. SDS is responsible of the decision at access control level for allowing or denying access to specific resources. SES on the other hand is responsible of decision enforcement. Such decision allows or denies access in reaction to the access control decision requested by the decision service (cf. Williams, 2009, pp. 7–8).



Source: Figure 5 in **(Williams, 2009, p. 7)**

**Fig. 8.1:** Security as a Service

The separation of enforcement and decision has the benefit of allowing different enforcement points to reuse the same decision point functionality. This will in turn support the reuse of SESOA components accordingly. Applying security as a service model to SESOA will move the decision service to its semantic service repository so that all business processes that will use the enforcement service can call the decision service and take benefit of its security functions (cf. Williams, 2009, p. 8).

By the extension of single decision-enforcement approach for SESOA, the SDS can be seen as a service-based PDP that is placed in the semantic service repository to be invoked as implementation-independent service. Such service will be offered only as a single service that can be replicated and distributed where it is needed. Such suggestion is helpful to make SDS providing the PDP functionalities for all of the service enforcement points that will be used in different business processes across enterprises.

When the SESOA semantic service repository will offer the SDS to the processing system and the other components, the resulted security pattern will be similar to the one depicted in Figure 8.2.

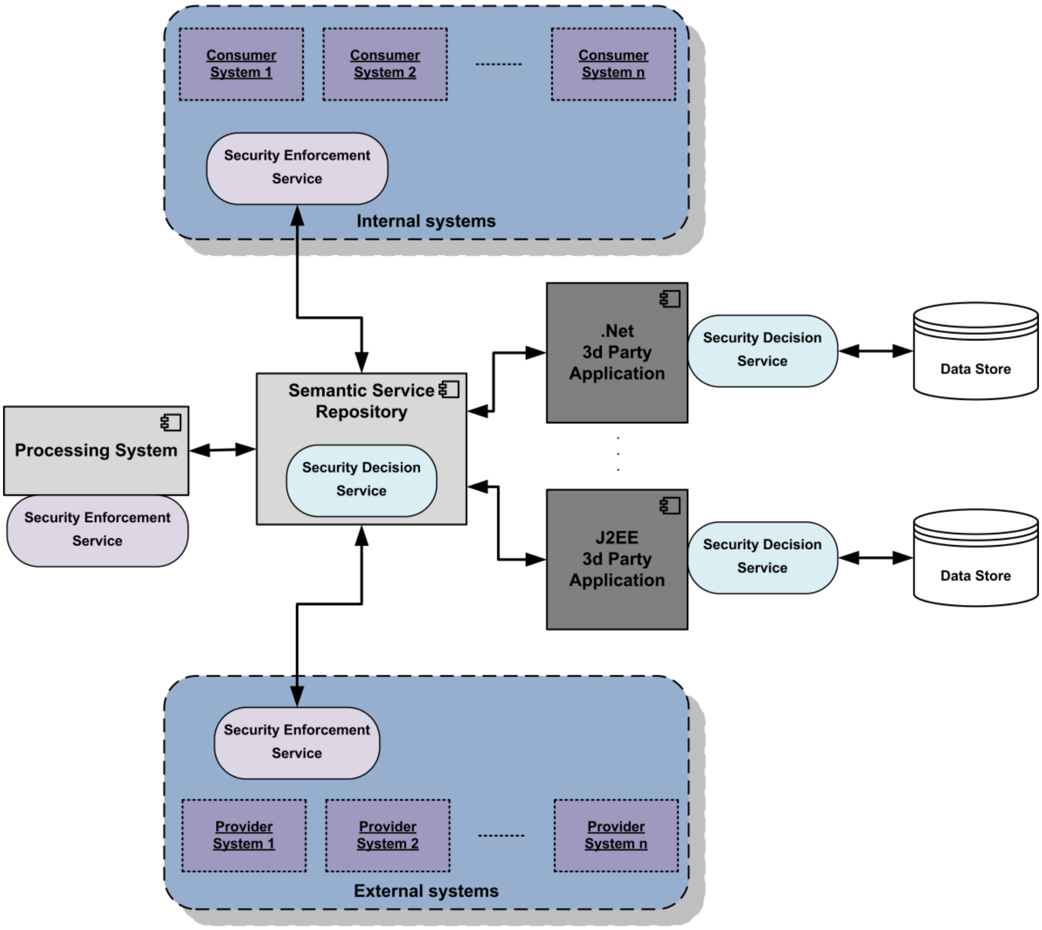

Consumer System 1
Consumer System 2
..........
Consumer System n
Security Enforcement Service
Internal systems
.Net 3d Party Application
Security Decision Service
Data Store
Semantic Service Repository
Security Decision Service
Processing System
Security Enforcement Service
J2EE 3d Party Application
Security Decision Service
Data Store
Security Enforcement Service
Provider System 1
Provider System 2
..........
Provider System n
External systems

**Fig. 8.2:** SESOA Potential Security Pattern

In this pattern, the security administration will be centralized at the semantic service repository level where all enforcement services will request for the SDS security functions. This decision to go to a centralized security pattern will be taken internally within the enterprise. If the provision of services in an automated manner will serve the business needs of the enterprise, then the SDS service can be configured to be called automatically from the enforcement services. On the other hand, enterprises can still initiate the security decision making via employees or contractors if it finds it suiting more to its business needs.

The automated scenario is as the one depicted in the figure above where the task of invoking the SDS is done from the SES via the processing system. Therefore, it is the business process designer's decision to include the security service at design time. Furthermore, if the semantic service repository will be globalized and can be accessed via external systems, each business application that wants to invoke such SDS can send a request via its SES to be permitted or denied based on some specific criteria that have to be defined if such approach sees the light.

### 8.2.2 Web Services Recommendation System

Another possible future research entry point that can result from the work done in this dissertation is developing a recommendation system for Web Services. Since the semantic service repository can be made available internally in an enterprise or globally, it is a very attracting idea to build a recommendation system for it.

This includes firstly building a catalogue of criteria needed to provide recommendations. This depends on where SESOA will be applied. However, a generic recommendation system is always favorable. It can be applied firstly to an open environment and then adjusted to enterprises based on their requirements.

The design of such recommendation system has to be done after analyzing all of the implemented business processes so that the algorithms needed to achieve recommendations can be initially trained. Moreover, it is always good practice to consult the business process designers to have a holistic view of the business needs that will form the requirements for building such system.

## 8.3 Wrap Up

Other possible future direction behind SESOA concept is the utilization of the Task Parallel Library (TPL) along the prototypical implementation to scale up the system to have a better performance on multi-core processors. From a semantic perspective, only RDF statements have been used in the implementation of the system's artifacts and other future work can enhance the semantic annotation to support higher level ontologies like e.g. OWL.

Many researches had been already conducted based on this work like (von der Dovenmühle, 2009; Hasan, 2010; Denker et al., 2011; Neemann, 2012). The bachelor work of (von der Dovenmühle, 2009) had the goal of applying validation to SESOA services. The master work of (Hasan, 2010) dealt with applying a machine-to-machine evaluation protocol to the services supplied in SESOA. The seminar work of (Denker et al., 2011) had one major goal of realizing the accompanying selling business process that had been built on top of SESOA concept. The master work of (Neemann, 2012) focused on the comparison between object-oriented business process management, automatic building blocks, workflows, and service-oriented architecture approaches. The result from this work was to point out the main differences between these approaches and to give a recommendation on which approach is considered the “best” in different business contexts. Furthermore, there are still ideas for two ongoing researches related to this research. They will end in the mid of 2013.

To sum up this dissertation, the main focus was to present SESOA concept with an accompanying business case built on top of it. SESOA is a SOA-based architecture that manages Web Services and semantically annotates their relations. Generally spoken, SESOA enables the realization of business processes as Web Service-based workflows to take as much benefit from the market’s best practices as possible. Moreover, one of the main purposes behind SESOA is to group Web Services in assemblages based on their actual business domain. All the details of the overall framework development had been presented together with the implementation of an accompanying business case.

# References

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M. T., Sheth, A., & Verma, K. (2005). Web Service Semantics - WSDL-S. *W3C*, *7*.

Alesso, H. P., & Smith, C. F. (2005). *Developing Semantic Web Services*. Natick, Massachusetts: A K Peters, Ltd.

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction*. USA: Oxford University Press.

Allam, N., Mahmoud, T., & Marx Gómez, J. (2011). Web Service-enabled Collaborative Corporate Environmental Management Information Systems. In P. Golinska, M. Fertsch, & J. Marx Gómez (Eds.), *Information Technologies in Environmental Engineering: New Trends and Challenges* (pp. 179–188). Presented at the ITEE 2011, Poznan, Poland: Springer (Heidelberg).

Allam, N., Mahmoud, T., Marx Gómez, J., & Junker, H. (2011). A Central Collaborative CEMIS. In *EnviroInfo 2011 - Innovations in Sharing Environmental Observations and Information* (pp. 683–691). Presented at the EnviroInfo Ispra 2011 - 25th International Conference Environmental Informatics, Ispra, Italy: Shaker Verlag.

Alwadain, A., Rosemann, M., Fielt, E., & Korthaus, A. (2011). Enterprise Architecture and the Integration of Service-Oriented Architecture. In *Proceedings of 15th Pacific Asia Conference on Information Systems (PACIS 2011).*

Antoniou, G., & Van Harmelen, F. (2004). *A Semantic Web Primer*. The MIT Press.

Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S., Takacsi-Nagy, P., & others. (2002). Web Service Choreography Interface (WSCI) 1.0. *World Wide Web Consortium (W3C)*. Retrieved September 2, 2012, from http://www.w3.org/TR/wsci/

Armario, J. M., Ruiz, D. M., & Armario, E. M. (2008). Market Orientation and Internationalization in Small and Medium-Sized Enterprises. *Journal of Small Business Management*, *46*(4), 485–511.

Arsanjani, A. (2004). Service-Oriented Modeling and Architecture. *IBM developer works*.

Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice* (2nd ed.). Addison-Wesley Professional.

Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., & others. (2005a). Semantic Web Services Framework (SWSF) Overview.

Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., & others. (2005b). Semantic Web Services Language (SWSL). *W3C Member Submission*, *9*.

Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., & others. (2005c). Semantic Web Services Ontology (SWSO). *W3C Member Submission*.

Bean, J. (2009). *SOA and Web Services Interface Design: Principles, techniques, and standards*. Morgan Kaufmann.

Bell, M. (2008). *Service-Oriented Modeling: Service Analysis, Design, and Architecture*. Wiley.

Berners-Lee, T. (1998). Semantic Web Road Map. *World Wide Web Consortium (W3C)*. Retrieved August 22, 2012, from http://www.w3.org/DesignIssues/Semantic.html

Berners-Lee, T. (2006). developerWorks interviews: Tim Berners-Lee. Retrieved from http://www.ibm.com/developerworks/podcast/dwi/cm-int082206txt.html

Berners-Lee, T., & Cailliau, R. (1990). WorldWideWeb: Proposal for a HyperText Project. *World Wide Web Consortium (W3C)*. Retrieved August 27, 2012, from http://www.w3.org/Proposal.html

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). Uniform Resource Identifier (URI): Generic Syntax. *RFC 3986, IETF*. Retrieved September 2, 2012, from http://merlot.tools.ietf.org/html/rfc3986

Berners-Lee, T., & Hendler, J. (2001). Scientific Publishing on the Semantic Web. *Nature*, *410*, 1023–1024.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American Magazine*.

Bieberstein, N., Bose, S., Fiammante, M., Jones, K., & Shah, R. (2005). *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap*. IBM Press.

Bolton, F. (2002). *Pure CORBA*. SAMS publishing.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., & Orchard, D. (2004). Web Services Architecture. *World Wide Web Consortium (W3C)*. Retrieved August 22, 2012, from http://www.w3.org/TR/ws-arch/

Bratt, S. (2006). *Emerging Web Technologies to Watch*. W3C (MIT, ERCIM, Keio). Retrieved from http://www.w3.org/2006/Talks/1023-sb-W3CTechSemWeb/

Braude, E. J. (2004). *Software Design: From Programming to Architecture*. J. Wiley.

Brehm, N. (2009). *Föderierte ERP-Systeme auf Basis von Web-Services* (PhD Thesis). Carl von Ossietzky University of Oldenburg, Oldenburg, Germany.

Brehm, N., Lübke, D., & Marx Gómez, J. (2007). Federated Enterprise Resource Planning Systems. In P. Saha (Ed.), *Handbook of Enterprise Systems Architecture in Practice* (pp. 290–305). London, UK: IGI Global.

Brehm, N., Mahmoud, T., Memari, A., & Marx Gómez, J. (2008). Towards Intelligent Discovery of Enterprise Architecture Services (IDEAS). *Journal of Enterprise Architecture*, *4*(3), 26–37.

Brehm, N., & Marx Gómez, J. (2005). Standardization Approach for Federated ERP Systems based on Web Services. In *Proceedings of the First International Workshop on Engineering Service Compositions (WESC'05)* (pp. 101–109). Amsterdam, Netherlands: IBM Research.

Brehm, N., & Marx Gómez, J. (2007). Web Service-Based Specification and Implementation of Functional Components in Federated ERP-Systems. In *Proceedings of the 10th International Conference on Business Information Systems* (Vol. 4439, pp. 133–146). Presented at the BIS 2007, Poznan, Poland: Springer-Verlag Berlin Heidelberg.

Brehm, N., & Marx Gómez, J. (2010). Secure Service Rating in Federated Software Systems Based on SOA. In C. A. Gutiérrez, E. Fernández-Medina, & M. Piattini (Eds.), *Web Services Security Development and Architecture: Theoretical and Practical Issues* (pp. 83–98). IGI Global.

Brehm, N., Marx Gómez, J., & Rautenstrauch, C. (2006). An ERP Solution Based on Web Services and Peer-to-Peer Networks for Small and Medium Enterprises. *International Journal of Information Systems and Change Management (IJISCM)*, *1*(1), 99–111.

Bremer, J., Mahmoud, T., & Rapp, B. (2012). Implementing CEMIS Workflows with State Chart XML. In H. K. Arndt, G. Knetsch, & W. Pillmann (Eds.), *EnviroInfo 2012 - Part 2: Open Data and Industrial Ecological Management* (pp. 749–757). Presented at the 26th International Conference on Informatics for Environmental Protection, Dessau, Germany: Shaker Verlag.

Brickley, D., & Guha, R. V. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. *W3C Recommendation*. Retrieved July 27, 2011, from http://www.w3.org/tr/rdf-schema/

Brodie, M. (1992). The Promise of Distributed Computing and the Challenges of Legacy Systems. In *Advanced Database Systems* (Vol. 618, pp. 1–28). Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-55693-1_29

Bussler, C. (2003a). *B2B Integration: Concepts and Architecture* (1 Edition.). Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Bussler, C. (2003b). The Role of Semantic Web Technology in Enterprise Application Integration. *IEEE Data Engineering*, *26*(4), 62–68.

Cabral, L., Domingue, J., Motta, E., Payne, T., & Hakimpour, F. (2004). Approaches to Semantic Web Services: an Overview and Comparisons. In *The Semantic Web: Research and Applications: First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004: proceedings* (Vol. 1, pp. 225–239).

Cardoso, J., Hepp, M., & Lytras, M. D. (2007). *The Semantic Web: Real World Applications from Industry* (1st ed.). Springer.

Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., & Wilkinson, K. (2004). Jena: Implementing the Semantic Web Recommendations. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters* (pp. 74–83). New York, NY, USA: ACM.

Casati, F., & Shan, M. C. (2001). Models and Languages for Describing and Discovering E-Services. In *SIGMOD Conference 2001 Proceedings* (p. 626). Presented at the International ACM SIGMOD RECORD Conference on Management of Data, Santa Barbara, California, USA.

Celino, I., de Medeiros, A. K. A., Zeissler, G., Oppitz, M., Facca, F., & Zoeller, S. (2007). Semantic Business Process Analysis. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management* (Vol. 251). Presented at the SBPM 2007, Innsbruck, Austria. Retrieved from http://ceur-ws.org/Vol-251/paper6.pdf

Channabasavaiah, K., Tuggle, E., & Holley, K. (2003). Migrating to a Service-Oriented Architecture. *IBM DeveloperWorks*.

Chappell, D. A. (2004). *Enterprise Service Bus*. O'Reilly Media, Inc.

Cheesman, J., & Ntinolazos, G. (2004). The SOA Reference Model. *CBDI Journal*.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., & others. (2001). *Web Services Description Language (WSDL) 1.1*. W3C.

Cimpian, E., & Mocan, A. (2005). WSMX Process Mediation Based on Choreographies (pp. 130–143). Presented at the 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management, Nancy, France.

Clement, L., Hately, A., von Riegen, C., & Rogers, T. (2004). UDDI Version 3.0. 2. *UDDI Spec Technical Committee Draft*. Retrieved August 9, 2011, from http://uddi.org/pubs/uddi-v3.0.2-20041019.htm

Clements, P., Bachmann, F., Bass, L., Garlan, D., Merson, P., Ivers, J., Little, R., Nord, R., & Stafford, J. (2010). *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional.

Collins, M. (2009). *Beginning WF: Windows Workflow in. NET 4.0*. Apress.

Colombo, M., Di Nitto, E., Di Penta, M., Distante, D., & Zuccalà, M. (2005). Speaking a Common Language: A Conceptual Model for Describing Service-Oriented Systems. In B. Benatallah, F. Casati, & P. Traverso (Eds.), *Service-Oriented Computing - ICSOC 2005* (Vol. 3826, pp. 48–60). Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/11596141_5

Conrad, S., & Turowski, K. (2001). Temporal OCL Meeting Specification Demands for Business Components. In T. Halpin & K. Siau (Eds.), *Unified Modeling Language: Systems Analysis, Design and Development Issues* (pp. 151–165). IGI Global.

Cooley, R., Mobasher, B., & Srivastava, J. (1997). Web Mining: Information and Pattern Discovery on the World Wide Web. In *Tools with Artificial Intelligence, 1997. Proceedings of the Ninth IEEE International Conference on* (pp. 558–567).

Cutter SOA Survey. (2008). Cutter Consortium, Benchmark Review Vol. 8, No. 1, January 2008, Graph 10, P. 25. Retrieved June 14, 2011, from http://www.cutter.com.mx/pdfs/cbr0801.pdf

Daemen, J., & Rijmen, V. (2002). *The Design of Rijndael: AES–The Advanced Encryption Standard*. Springer.

Daigneau, R. (2011). *Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services*. Addison-Wesley Professional.

Dan, A., Johnson, R. D., & Carrato, T. (2008). SOA Service Reuse by Design. In *Proceedings of the 2nd international workshop on Systems development in SOA environments* (pp. 25–28). New York, NY, USA: ACM. doi:10.1145/1370916.1370923

De Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Oren, E., & others. (2005, June 3). Web Service Modeling Ontology (WSMO). *W3C Member Submission*. Retrieved October 17, 2011, from http://www.w3.org/Submission/WSMO/

De Bruijn, J., Lausen, H., Polleres, A., & Fensel, D. (2006). The Web Service Modeling Language WSML: An Overview. In *Proceedings of the 3rd European Semantic Web Conference, ESWC 2006, June 11-14, 2006* (pp. 590–604). Budva, Montenegro: Springer-Verlag Berlin Heidelberg.

De Giorgio, T., Ripa, G., & Zuccalà, M. (2010). An Approach to Enable Replacement of SOAP Services and REST Services in Lightweight Processes. In *Proceedings of the 10th International Conference on Current Trends in Web Engineering* (pp. 338–346). Vienna, Austria: Springer-Verlag Berlin Heideberg.

Deeken, W., Eberhard, I., Grohmann,, M., Martens, A., Mesick, T., Peters, D., Schönbohm, T., Schwitschkowski, M., Stolarek, C., Thobe, A., Wang, Y., Wang,, Y., Wülpern, T., Xu, H., & Ziesenitz, A. (2007). *The Report of the "Förderierte ERP-Systeme auf der Basis von Web Services und P2P-Systemen" Project Group*. Oldenburg, Germany.

Denker, C., Gerken, C., Holtmann, R., Petersen, M., & Rummel, D. (2011). Handout of the VLBA Seminar: SOA and Business Processes.

Dietz, J. L. G. (2006). *Enterprise Ontology: Theory and Methodology*. Springer Verlag.

Domingue, J., Fensel, D., & Hendler, J. A. (2011). *Handbook of Semantic Web Technologies* (1st ed.). Springer-Verlag Berlin Heideberg.

Dreibelbis, A., Hechler, E., Milman, I., Oberhofer, M., van Run, P., & Wolfson, D. (2008). *Enterprise Master Data Management: An SOA Approach to Managing Core Information*. IBM Press.

Eastlake, D., & Jones, P. (2001). *US Secure Hash Algorithm 1 (SHA1)*. RFC 3174.

Eekels, J., & Roozenburg, N. F. M. (1991). A Methodological Comparison of the Structures of Scientific Research and Engineering Design: Their Similarities and Differences. *Design Studies, 12*(4), 197–203.

Elenius, D., Denker, G., Martin, D., Gilham, F., Khouri, J., Sadaati, S., & Senanayake, R. (2005). The OWL-S Editor - A Development Tool for Semantic Web Services. In *Proceedings of the Second European conference on The Semantic Web: research and Applications* (pp. 78–92). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/11431053_6

Erl, T. (2004). *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall PTR.

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall.

Erl, T. (2007). *SOA: Principles of Service Design*. Prentice Hall Press.

Erl, T. (2009). *SOA Design Patterns (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Prentice Hall PTR.

Farrell, J., & Lausen, H. (2007). Semantic Annotations for WSDL and XML Schema. *W3C Recommendation*. Retrieved September 13, 2012, from http://www.w3.org/TR/sawsdl/

Fellner, K. J., & Turowski, K. (2000). Identifying Business Components Using Conceptual Models. In *Information Resources Management Association International Conference* (pp. 161–165). Anchorage, Alaska, USA: IDEA Group Publishing.

Fensel, D., & Bussler, C. (2002). The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, *1*(2), 113–137.

Fensel, D., Facca, F. M., Simperl, E., & Toma, I. (2011). *Semantic Web Services*. Springer-Verlag Berlin Heideberg.

Fensel, D., Fischer, F., Kopeckỳ, J., Krummenacher, R., Lambert, D., & Vitvar, T. (2010). WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web. *W3C Member Submission*. Retrieved September 13, 2012, from http://www.w3.org/Submission/WSMO-Lite/

Fensel, D., Lausen, H., Polleres, A., Bruijn, J., Stollberg, M., Roman, D., & Domingue, J. (2007). *Enabling Semantic Web Services: The Web Service Modelling Ontology*. Springer-Verlag New York Inc.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (PhD Thesis). University Of California, Irvine, USA.

Fielding, R. T., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. Retrieved from http://tools.ietf.org/html/rfc2616

Frankel, D. S. (2003). BPM and MDA: The Rise of Model-Driven Enterprise Systems. *Business Process Trends*. Retrieved from http://www.businessprocesstrends.com/

Giachetti, R. E. (2010). *Design of Enterprise Systems: Theory, Architecture, and Methods*. CRC Press.

Gräuler, M., Teuteberg, F., Mahmoud, T., & Marx Gómez, J. (2012). Anforderungspriorisierung und Designempfehlungen für Betriebliche Umweltinformationssysteme der nächsten Generation – Ergebnisse einer explorativen Studie. In

*MKWI2012* (pp. 1531–1543). Presented at the Multikonferenz Wirtschaftsinformatik 2012, Braunschweig, Germany: GITO mbH Verlag Berlin.

Gräuler, M., Teuteberg, F., Mahmoud, T., & Marx Gómez, J. (2013). Requirements Prioritization and Design Considerations for the Next Generation of Corporate Environmental Management Information Systems - A Foundation for Innovation. *International Journal of Information Technologies and the Systems Approach (IJITSA) - Special Issue on "IT Goes Green: Systemic Approaches to IT Policy Making, Design, Evaluation and Management."*

Grimes, R., & Grimes, D. (1997). *Professional DCOM Programming*. Birmingham, UK: Wrox Press Ltd.

Grönroos, C. (2000). *Service Management and Marketing: A Customer Relationship Management Approach*. John Wiley & Sons Inc.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, *5*(2), 199–220.

Haas, H., & Brown, A. (2004). Web Services Glossary. *World Wide Web Consortium (W3C)*. Retrieved July 13, 2012, from http://www.w3.org/TR/ws-gloss/

Haerder, T., & Reuter, A. (1983). Principles of Transaction-Oriented Database Recovery. *ACM Computing Surveys*, *15*(4), 287–317. doi:http://doi.acm.org/10.1145/289.291

Haller, A., Cimpian, E., Mocan, A., Oren, E., & Bussler, C. (2005). WSMX - A Semantic Service-Oriented Architecture. In *Proceedings of the IEEE International Conference on Web Services* (pp. 321–328). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ICWS.2005.139

Haller, A., Gomez, J. M., & Bussler, C. (2005). Exposing Semantic Web Service Principles in SOA to Solve EAI Scenarios. In *Web Service Semantics: Towards Dynamic Business Integration Workshop. In conjunction with The 14th International World Wide Web Conference (WWW 2005)*.

Hammer, M., & Champy, J. (2003). *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperBusiness.

Handfield, R. B., Nichols, E. L., & Ernest, L. (1999). *Introduction to Supply Chain Management*. Prentice Hall Englewood Cliffs, NJ.

Handschuh, S., & Staab, S. (2003). *Annotation for the Semantic Web*. Amsterdam, Netherlands: IOS Press.

Hasan, B. (2010). *Konzeption und Umsetzung eines Sicherheitsprotokolls zur Bewertung von Web Services in SOA-basierten ERP-Systemen* (Master Thesis). Carl von Ossietzky University of Oldenburg, Oldenburg, Germany.

Heineman, G. T., & Councill, W. T. (2001). *Component-Based Software Engineering: Putting the Pieces Together* (Vol. 17). Addison-Wesley USA.

Herrmann, M., Dalferth, O., & Aslam, M. A. (2007). Applying Semantics (WSDL, WSDL-S, OWL) in Service Oriented Architectures (SOA). In *10th International Protégé Conference - July 15-18, 2007*. Budapest, Hungary.

Heuser, L., Alsdorf, C., & Woods, D. (2008). The Web-Based Service Industry-Infrastructure for Enterprise SOA 2.0, Potential Killer Applications-Semantic Service Discovery. In *International Research Forum*. Evolved Technologist Press.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–105.

Hitzler, P., Krötzsch, M., & Rudolph, S. (2009). *Foundations of Semantic Web Technologies* (1st ed.). Chapman & Hall - CRC Press.

Hohpe, G., & Woolf, B. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Hollingsworth, D. (1995). Workflow Management Coalition: The Workflow Reference Model. *Workflow Management Coalition*.

Hu, Y., Yang, Q. P., Sun, X., & Wei, P. (2008). Applying Semantic Web Services to Enterprise Web. In *The 6th International Conference on Manufacturing Research (ICMR08), Brunel university, UK, 9-11th September 2008. pp. 589-595.*

Inaganti, S., & Behara, G. K. (2007). Service Identification: BPM and SOA Handshake. *BPTrends*, *3*, 1–12.

Ionita, A. D., Florea, M., & Jelea, L. (2009). 4+1 Views for a Business Cooperation Framework Based on SOA. *IAENG International Journal of Computer Science*, *36*(4), 332–343.

J. F. Nunamaker, J., Chen, M., & Purdin, T. D. M. (1990). Systems Development in Information Systems Research. *Journal of Management Information Systems, Winter1990/91*, *7*(3), 89–106.

Järvinen, P. (2007). Action Research is Similar to Design Science. *Quality & Quantity*, *41*(1), 37–54.

Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., & others. (2007). Web Services Business Process Execution Language Version 2.0. *OASIS | Advancing open standards for the information society*. Retrieved September 2, 2012, from docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

Josuttis, N. (2007). *SOA in Practice* (1st ed.). O'Reilly.

Kaliski Jr, B. S., & Redwood City, C. (1991). An Overview of the PKCS standards. *RSA Data Security, Inc, 3*.

Karakostas, B., & Zorgios, Y. (2008). *Engineering Service Oriented Systems: A Model Driven Approach*. IGI Global.

Katzan, H. (2008). *A Manager's Guide to Service Science*. USA: iUniverse.

Katzan Jr, H. (2008). *Foundations of Service Science: A Pragmatic Approach*. iUniverse.

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., & Barreto, C. (2005). Web Services Choreography Description Language Version 1.0. *W3C Candidate Recommendation*. Retrieved September 2, 2012, from http://www.w3.org/TR/ws-cdl-10/

Keen, M., Acharya, A., Bishop, S., Hopkins, A., Milinski, S., Nott, C., Robinson, R., Adams, J., & Verschueren, P. (2004). *Patterns: Implementing an SOA Using an Enterprise Service Bus*. IBM RedBooks.

Kistasamy, C., Van Der Merwe, A., & De La Harpe, A. (2010). The Relationship between Service Oriented Architecture and Enterprise Architecture. In *14th IEEE*

*International Enterprise Distributed Object Computing Conference* (pp. 129–137). Presented at the EDOC 2010, Vitória, Brazil: IEEE Computer Society 2010.

Kjernsmo, K., & Passant, A. (2009). SPARQL New Features and Rationale. *World Wide Web Consortium (W3C)*. Retrieved August 28, 2012, from http://www.w3.org/TR/sparql-features/

Kopeckỳ, J., Gomadam, K., & Vitvar, T. (2008). hRESTS: An HTML Microformat for Describing RESTful Web Services. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (pp. 619–625). Sydney, Australia.

Kopeckỳ, J., & Vitvar, T. (2008). WSMO-Lite: Lowering the Semantic Web Services Barrier with Modular and Light-Weight Annotations. In *Proceedings of the 2008 IEEE International Conference on Semantic Computing* (pp. 238–244).

Kopeckỳ, J., Vitvar, T., Bournez, C., & Farrell, J. (2007). SAWSDL: Semantic Annotations for WSDL and XML Schema. *IEEE Internet Computing*, *11(6)*, 60–67.

Kopeckỳ, J., Vitvar, T., & Fensel, D. (2008). MicroWSMO: Semantic Description of RESTful Services. *WSMO Deliverable*. Retrieved June 14, 2011, from http://www.wsmo.org/TR/d38/v0.1

Krafzig, D., Banke, K., & Slama, D. (2005). *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall PTR.

Kreger, H. (2001). Web Services Conceptual Architecture (WSCA 1.0). IBM Software Group. Retrieved from www.csd.uoc.gr/~hy565/newpage/docs/pdfs/papers/wsca.pdf

Kruchten, P. (1995). The 4+1 View Model of Architecture. *IEEE Software*, *12*(6), 42–50.

Kuechler, B., & Vaishnavi, V. (2008). On Theory Development in Design Science Research: Anatomy of a Research Project. *European Journal of Information Systems*, *17*(5), 489–504.

Lafon, Y., & Mitra, N. (2007). SOAP Version 1.2 Part 0: Primer (Second Edition). *World Wide Web Consortium (W3C) Recommendation*. Retrieved from http://www.w3.org/TR/soap12-part0/

Lambert, D., & Benn, N. (2010). Deliverable 3.1 - Standardisation Activity Report. Service Web 3.0 FP7 Project. Retrieved from http://www.serviceweb30.eu/cms/index.php/resources/doc_view/100-d31-standardization-activity-report-m24ace7.pdf?tmpl=component&format=raw

Lamparter, S. (2007). *Policy-based Contracting in Semantic Web Service Markets*. University of Karlsruhe (TH), Karlsruhe, Germany.

Lawrence, K., Kaler, C., Nadalin, A., Monzillo, R., & Hallam-Baker, P. (2006). Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). *OASIS Standard Specification*. Retrieved September 3, 2012, from https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-soapmessagesecurity.pdf

Linthicum, D. S. (2000). *Enterprise Application Integration*. Addison-Wesley Longman Ltd. Essex, UK, UK.

Lytras, M. D., & García, R. (2008). Semantic Web Applications: A Framework for Industry and Business Exploitation – What Is Needed for the Adoption of the Semantic Web from the Market and Industry. *International Journal of Knowledge and Learning*, *4*(1), 93–108.

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., & Metz, R. (2006). Reference Model for Service Oriented Architecture 1.0. *OASIS | Advancing open standards for the information society*. Retrieved August 29, 2012, from docs.oasis-open.org/soa-rm/v1.0/soa-rm.doc

Magal, S. R., & Word, J. (2011). *Integrated Business Processes with ERP Systems*. John Wiley & Sons.

Mahmoud, T. (2009). Semantic SOA-Based Model to be applied in Business Environments. In *CENTERIS 2009 Conference of Enterprise Information Systems Proceedings* (pp. 473–482). Ofir, Portugal.

Mahmoud, T., & Marx Gómez, J. (2008a). Integration of Semantic Web Services Principles in SOA to Solve EAI and ERP Scenarios - Towards Semantic Service Oriented Architecture. In *Proceedings of the 3rd International Conference on Information & Communication Technologies: from Theory to Applications (ICTTA-2008)* (pp. 957–958). Damascus, Syria: IEEE.

Mahmoud, T., & Marx Gómez, J. (2008b). Semantic Web Services Process Mediation Using WSMX Concepts. In *Proceedings of the 20th International Conference on Systems Research, Informatics and Cybernetics (InterSymp-2008) - Engineering and Management of IT-Based Organizational Systems: A Systems Approach* (pp. 28–32). Baden Baden, Germany.

Mahmoud, T., & Marx Gómez, J. (2010). Applying Semantic SOA Based Model to Business Applications. In M. M. Cruz-Cunha & J. Varajao (Eds.), *Enterprise Information Systems Design, Implementation and Management: Organizational Applications* (pp. 1–20). IGI Global.

Mahmoud, T., Marx Gómez, J., Rezgui, A., Peters, D., & Solsbach, A. (2012). Enhanced BI Systems with On-Demand Data Based on Semantic-Enabled Enterprise SOA. In *Proceedings of the 20th European Conference on Information Systems, Paper 184*. Presented at the ECIS 2012, Barcelona, Spain. Retrieved from http://aisel.aisnet.org/ecis2012/184

Mahmoud, T., Marx Gómez, J., & von der Dovenmühle, T. (2011). Functional Components Specification in the Semantic SOA-based Model. In S. Smolnik, F. Teuteberg, & O. Thomas (Eds.), *Semantic Technologies for Business and Information Systems Engineering: Concepts and Applications* (pp. 277–291). IGI Global.

Mahmoud, T., Petersen, M., & Rummel, D. (2012). Business Process Integration within Lightweight Semantic-Enabled Enterprise Service-Oriented Architecture. In P. O. de Pablos, J. M. Cueva Lovelle, J. E. Labra Gayo, & R. Tennyson (Eds.), *E-Procurement Management for Successful Electronic Government Systems* (pp. 181–192). E-Procurement Management for Successful Electronic Government Systems.

Mahmoud, T., von der Dovenmühle, T., & Marx Gómez, J. (2009). Web Service Validation within Semantic SOA-Based Model. In D. Davcev & J. Marx Gómez (Eds.), *Proceedings of the ICT Innovations Conference 2009* (pp. 295–303). Presented at the ICT2009, Ohrid, Macedonia: Springer (Heidelberg).

Maleshkova, M., Kopeckỳ, J., & Pedrinaci, C. (2009). Adapting SAWSDL for Semantic Annotations of RESTful Services. Presented at the On the Move to Meaningful Internet Systems: OTM 2009 Workshops, Vilamoura, Portugal.

Manola, F., Miller, E., & McBride, B. (2004). RDF Primer. *W3C Recommendation*. Retrieved August 27, 2012, from http://www.w3.org/TR/rdf-primer/

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., & others. (2004). OWL-S: Semantic Markup for Web Services. *W3C Member Submission*. Retrieved September 13, 2012, from http://www.w3.org/Submission/OWL-S/

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., & others. (2005). Bringing Semantics to Web Services: The OWL-S Approach. In *Semantic Web Services and Web Process Composition: First International Workshop, SWSWPC 2004, San Diego, CA, USA, July 6, 2004* (pp. 26 – 42).

Martin, D., Paolucci, M., & Wagner, M. (2007). Toward Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspective. In *OWL-S: Experiences and Directions-Workshop at 4th European Semantic Web Conference (ESWC)*. Innsbruck, Austria.

Marx Gómez, J., Rapp, B., Solsbach, A., Mahmoud, T., Memari, A., & Bremer, J. (2011). Projekt IT-for-Green: Umwelt-, Energie- und Ressourcenmanagement mit BUIS der nächsten Generation. *Ökonomikuss Sommerausgabe 2011*, 18–20.

Marx Gómez, J., Rautenstrauch, C., & Cissek, P. (2009). *Einführung in Business Intelligence mit SAP NetWeaver 7.0*. Berlin, Heidelberg: Springer Verlag.

Maximilien, E. M., & Singh, M. P. (2004). Toward Autonomic Web Services Trust and Selection. In *ICSOC'04: Proceedings of the 2nd International Conference on Service Oriented Computing* (pp. 212–221). New York, NY, USA: ACM.

McGuinness, D. L., & Van Harmelen, F. (2004). OWL Web Ontology Language Overview. *W3C Recommendation*. Retrieved August 28, 2012, from http://www.w3.org/TR/owl-features/

McIlraith, S. A., Son, T. C., & Zeng, H. (2001). Semantic Web Services. *Intelligent Systems, IEEE, 16*(2), 46–53.

Medjahed, B. (2004). *Semantic Web Enabled Composition of Web Services* (PhD Thesis). Virginia Polytechnic Institute and State University, Falls Church, Virginia, USA.

Meyerholt, D., Mahmoud, T., & Marx Gómez, J. (2011). Administrating Environmental Performance Indicators Utilizing Lightweight Semantic Web Services. In P. Golinska, M. Fertsch, & J. Marx Gómez (Eds.), *EnviroInfo 2011 - Information Technologies in Environmental Engineering: New Trends and Challenges* (pp. 301–309). Presented at the ITEE 2011, Poznan, Poland: Springer (Heidelberg).

Meyerholt, D., Marx Gómez, J., Dada, A., Bremer, J., & Rapp, B. (2010). Bringing Sustainability to the Daily Business: The OEPI project. In *Proceedings of the Workshop "Environmental Information Systems and Services-Infrastructures and Platforms". CEUR workshop proceedings* (Vol. 679, pp. 1613–0073).

Monk, E. F., & Wagner, B. J. (2008). *Concepts in Enterprise Resource Planning* (3rd ed.). Course Technology.

Neemann, H. (2012). *Objektorientierte Softwareentwicklung im Kontext des Geschäftsprozessmanagements*. Carl von Ossietzky University of Oldenburg, Oldenburg, Germany.

Newcomer, E., & Lomow, G. (2004). *Understanding SOA with Web Services (Independent Technology Guides)*. Addison-Wesley Professional.

Nordsieck, F. (1934). *Grundlagen der Organisationslehre*. Nihon Shoseki.

Oberle, D. (2005). *Semantic Management of Middleware*. University of Karlsruhe (TH), Karlsruhe, Germany.

OMG. (2009). *OMG Unified Modeling Language$^{TM}$ (OMG UML), Superstructure - Version 2.2*. Object Management Group. Retrieved from www.omg.org/spec/UML/2.1.2/Superstructure/PDF/

Orfali, R., Harkey, D., & Edwards, J. (2007). *Client/Server Survival Guide, 3rd Edition*. Wiley India.

Pachghare, V. K. (2009). *Cryptography and Information Security*. Prentice-Hall Of India Pvt. Ltd.

Papazoglou, M. (2008). *Web Services: Principles and Technology*. Pearson - Prentice Hall.

Patel-Schneider, P. F., Hayes, P., & Horrocks, I. (2004). OWL Web Ontology Language Semantics and Abstract Syntax. *W3C recommendation*. Retrieved July 28, 2011, from http://www.w3.org/TR/owl-semantics/

Payne, T., & Lassila, O. (2004). Semantic Web Services. *Intelligent Systems, IEEE, 19*(4), 14–15.

Pedrinaci, C., Lambert, D., Maleshkova, M., Liu, D., Domingue, J., & Krummenacher, R. (2010). Adaptive Service Binding with Lightweight Semantic Web Services. In S. Dustdar & F. Li (Eds.), *Service Engineering: European Research Results* (pp. 233–260). SpringerWienNewYork.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. doi:10.2753/MIS0742-1222240302

Prud'Hommeaux, E., & Seaborne, A. (2008). SPARQL Query Language for RDF. *W3C Recommendation*. Retrieved August 9, 2011, from http://www.w3.org/TR/rdf-sparql-query/

Rapp, B., & Bremer, J. (2012). Design of an Event Engine for Next Generation CEMIS: A Use Case. In H. K. Arndt, G. Knetsch, & W. Pillmann (Eds.), *EnviroInfo 2012 - Part 2: Open Data and Industrial Ecological Management* (pp. 759–766). Presented at the 26th International Conference on Informatics for Environmental Protection, Dessau, Germany: Shaker Verlag.

Rapp, B., Solsbach, A., Mahmoud, T., Memari, A., & Bremer, J. (2011). IT-for-Green: Next Generation CEMIS for Environmental, Energy and Resource Management. In *EnviroInfo 2011 - Innovations in Sharing Environmental Observations and Information* (pp. 573–581). Presented at the EnviroInfo Ispra 2011 - 25th International Conference Environmental Informatics, Ispra, Italy: Shaker Verlag.

Richardson, L., & Ruby, S. (2007). *RESTful Web Services*. O'Reilly Media.

Robert Jacobs, F., & "Ted" Weston, J. (2007). Enterprise Resource Planning (ERP)—A Brief History. *Journal of Operations Management*, *25*(2), 357 – 363. doi:10.1016/j.jom.2006.11.005

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., & Fensel, D. (2005). Web Service Modeling Ontology. *Applied Ontology*, *1*(1), 77–106.

Rosen, M., Lublinsky, B., Smith, K. T., & Balcer, M. J. (2008). *Applied SOA: Service-Oriented Architecture and Design Strategies*. Wiley.

Roshen, W. (2009). *SOA-Based Enterprise Integration: A Step-by-Step Guide to Services-Based Application*. McGraw-Hill/Osborne.

Rossi, M., & Sein, M. K. (2003). Design Research Workshop: A Proactive Research Approach. Presented at the Twenty-Sixth Information Systems Research Seminar in Scandinavia, Information Systems Research in Scandinavia Association, Haikko, Finland.

Sabou, M. (2005). Learning Web Service Ontologies: Challenges, Achievements and Opportunities. In *Dagstuhl Seminar*.

Schäfer, R. (2001). Rules for Using Multi-Attribute Utility Theory for Estimating a User's Interests. In *Online-Proceedings des 9. GI-Workshops: ABIS-Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*. Presented at the ABIS-Workshop 2001, Dortmund, Germany.

Schmietendorf, A., Lezius, J., Dimitrov, E., Reitz, D., & Dumke, R. (2003). *Aktuelle Ansätze für Web Service basierte Integrationslösungen*. Magdeburg, Germany: Otto-von-Guericke-Universität.

Scott Allen, K. (2006). *Programming Windows Workflow Foundation: Practical WF Techniques and Examples Using XAML and C#* (1st ed.). Packt Publishing Ltd.

Seidlmeier, H. (2010). *Prozessmodellierung mit ARIS®: Eine beispielorientierte Einführung für Studium und Praxis.* (3rd ed.). Vieweg+ Teubner Verlag | Springer Fachmedien Wiesbaden GmbH.

Sharp, J. (2010). *Microsoft® Windows® Communication Foundation 4 Step by Step* (1st ed.). Microsoft Press.

Shishkov, B., van Sinderen, M., & Quartel, D. (2006). SOA-Driven Business-Software Alignment (pp. 86–94). Presented at the IEEE International Conference on e-Business Engineering (ICEBE 2006), Beijing, China.

Silver, M. S., Markus, M. L., & Beath, C. M. (1995). The Information Technology Interaction Model: A Foundation for the MBA Core Course. *MIS quarterly, 19*(3), 361–390.

Simon, H. A. (1996). *The Sciences of the Artificial* (Third.). Cambridge, MA: MIT Press.

Sivashanmugam, K., Verma, K., Sheth, A., & Miller, J. (2003). Adding Semantics to Web Services Standards. In *Proceedings of the International Conference on Web Services* (pp. 395–401).

Srinivasan, N., Paolucci, M., & Sycara, K. (2004). Adding OWL-S to UDDI, Implementation and Throughput. *proceeding of Semantic Web Service and Web Process Composition 2004*.

Stalk, G., Evans, P., & Sgulman, L. E. (1992). *Competing on Capabilities: The New Rules of Corporate Strategy*. Harvard Business Review.

Stallings, W. (2010). *Cryptography and Network Security: Principles and Practice* (Vol. 998). Prentice Hall.

Steiner, G. A. (1997). *Strategic Planning*. Simon & Schuster.

Studer, R., Grimm, S., & Abecker, A. (2007). *Semantic Web Services: Concepts, Technologies, and Applications*. Heidelberg, Germany: Springer Verlag.

Tauberer, J. (2010). SemWeb.NET. *SemWeb.NET: Semantic Web/RDF Library for C#/.NET*. Retrieved January 3, 2012, from http://razor.occams.info/code/semweb/

Teuteberg, F., & Marx Gómez, J. (2010). Green Computing & Sustainability - Status Quo und Herausforderungen für BUIS der nächsten Generation. *HMD - Praxis Wirtschaftsinform., 274*(47).

Tran, H., Zdun, U., & Dustdar, S. (2007). View-Based and Model-Driven Approach for Reducing the Development Complexity in Process-Driven SOA. In *Business Process and Services Computing: 1st International Working Conference on Business Process and Services Computing* (Vol. 116, pp. 105–124). Presented at the BPSC 2007, Leipzig, Germany.

Turner, M., Budgen, D., & Brereton, P. (2003). Turning Software into a Service. *Computer, 36*(10), 38–44.

Turowski, K. (2000). Establishing Standards for Business Components. In K. Jakobs (Ed.), *Information Technology Standards and Standardization: A Global Perspective* (pp. 131–151). Hershey: Idea Group Publishing.

Turowski, K. (2001). *Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme*. Otto-von-Guericke-Universität Magdeburg, Magdeburg, Germany.

Usländer, T. (2010). *Service-Oriented Design of Environmental Information Systems*. KIT Scientific Publishing.

Vaishnavi, V., & Kuechler, B. (2004). Design Science Research in Information Systems. *Association for Information Systems*. Retrieved July 4, 2012, from http://desrist.org/desrist

Van der Aalst, W. M. P. (2009). Challenges in Business Process Analysis. In J. Filipe, J. Cordeiro, & J. Cardoso (Eds.), *Enterprise Information Systems* (Vol. 12, pp. 27–42). Springer Berlin Heidelberg.

Vitvar, T., Kopecký, J., Viskova, J., & Fensel, D. (2008). WSMO-Lite Annotations for Web Services. In *Proceedings of the 5th European Semantic Web Conference on The Semantic Web: Research and Applications* (pp. 674–689). Tenerife, Canary Islands, Spain.

Vom Brocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., & Cleven, A. (2009). Reconstructing the Giant: On the Importance of Rigour in Documenting the Literature Search Process. In *Proceedings of the 17th European Conference on Information Systems* (pp. 1–13). Presented at the ECIS 2009, Verona, Italy.

Von der Dovenmühle, T. (2009). *Validierung adaptierter Web Services innerhalb eines Semantic SOA* (Bachelor Thesis). Carl von Ossietzky University of Oldenburg, Oldenburg, Germany.

Walls, J. G., Widmeyer, G. R., & El Sawy, O. A. (1992). Building an Information System Design Theory for Vigilant EIS. *Information Systems Research, 3*(1), 36–59.

Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, *26*(2), xiii–xxiii.

White, S. A. (2004). Introduction to BPMN. *IBM Cooperation*.

White, S. A., & Miers, D. (2008). *BPMN Modeling and Reference Guide*. Future Strategies Inc.

Williams, M. (2009). Technology Policy Level 1 - Security as a Service. Highways Agency. Retrieved from http://www.highways.gov.uk/business/documents/Technology_Policy_L1_Security_as_a_Service_2.doc

Xu, L., & Peng, B. (2005). An Intelligent Retrieval Framework in Semantic Web Based on Agents. In D. Li & B. Wang (Eds.), *Artificial Intelligence Applications and Innovations II*. Springer.

Zeginis, C., & Plexousakis, D. (2010). *Monitoring the QoS of Web Services using SLAs* (Technical Report No. 404).

Zimmermann, O., Koehler, J., & Leymann, F. (2006). The Role of Architectural Decisions in Model-Driven SOA Construction. In *International Conference on Object-Oriented Programming, Systems, Languages, and Applications, Best Practices and Methodologies in Service-Oriented Architectures*. Portland: ACM.

# Publications

## Journals

Gräuler, M., Teuteberg, F., **Mahmoud, T.**, & Marx Gómez, J. (2013). Requirements Prioritization and Design Considerations for the Next Generation of Corporate Environmental Management Information Systems - A Foundation for Innovation. International Journal of Information Technologies and the Systems Approach (IJITSA) - Special Issue on "IT Goes Green: Systemic Approaches to IT Policy Making, Design, Evaluation and Management."

Marx Gómez, Jorge, Rapp, B., Solsbach, A., **Mahmoud, T.**, Memari, A., & Bremer, J. (2011). Projekt IT-for-Green: Umwelt-, Energie- und Ressourcenmanagement mit BUIS der nächsten Generation. Ökonomikuss Sommerausgabe 2011, 18-20.

Brehm, N., **Mahmoud, T.**, Memari, A., & Marx Gómez, J. (2008). Towards Intelligent Discovery of Enterprise Architecture Services (IDEAS). Journal of Enterprise Architecture, 4(3), 26-37.

## Conferences

**Mahmoud, T.** (2013). CEMIS Next Generation Supported by Semantic Enterprise Service-Oriented Architecture. In F. Gargouri & W. Mahdi (Eds.), Proceedings of the 5th International Conference on Web and Information Technologies (pp. 21–30). Presented at the ICWIT'13, Hammamet, Tunisia.

Bremer, J., **Mahmoud, T.**, & Rapp, B. (2012). Implementing CEMIS Workflows with State Chart XML. In H. K. Arndt, G. Knetsch, & W. Pillmann (Eds.), EnviroInfo 2012 - Part 2: Open Data and Industrial Ecological Management (pp. 749–757). Presented at the 26th International Conference on Informatics for Environmental Protection, Dessau, Germany: Shaker Verlag.

**Mahmoud, T.**, Marx Gómez, J., Rezgui, A., Peters, D., & Solsbach, A. (2012). Enhanced BI Systems with On-Demand Data Based on Semantic-Enabled Enterprise SOA. Proceedings of the 20th European Conference on Information Systems, Paper 184. Presented at the ECIS 2012, Barcelona, Spain. Retrieved from http://aisel.aisnet.org/ecis2012/184

Gräuler, M., Teuteberg, F., **Mahmoud, T.**, & Marx Gómez, J. (2012). Anforderungspriorisierung und Designempfehlungen für Betriebliche Umweltinformationssysteme der nächsten Generation – Ergebnisse einer explorativen Studie. MKWI2012 (pp. 1531–1543). Presented at the Multikonferenz Wirtschaftsinformatik 2012, Braunschweig, Germany: GITO mbH Verlag Berlin.

Rapp, B., Solsbach, A., **Mahmoud, T.**, Memari, A., & Bremer, J. (2011). IT-for-Green: Next Generation CEMIS for Environmental, Energy and Resource Management. Innovations in Sharing Environmental Observations and Information (pp. 573–581). Presented at the EnviroInfo Ispra 2011 - 25th International Conference Environmental Informatics, Ispra, Italy: Shaker Verlag.

Allam, N., **Mahmoud, T.**, Marx Gómez, J., & Junker, H. (2011). A Central Collaborative CEMIS. Innovations in Sharing Environmental Observations and Information (pp. 683–691). Presented at the EnviroInfo Ispra 2011 - 25th International Conference Environmental Informatics, Ispra, Italy: Shaker Verlag.

Allam, N., **Mahmoud, T.**, & Marx Gómez, J. (2011). Web Service-enabled Collaborative Corporate Environmental Management Information Systems. Information Technologies in Environmental Engineering: New Trends and Challenges (pp. 179-188). Presented at the ITEE 2011, Poznan, Poland: Springer (Heidelberg).

Meyerholt, D., **Mahmoud, T.**, & Marx Gómez, J. (2011). Administrating Environmental Performance Indicators Utilizing Lightweight Semantic Web Services. Information Technologies in Environmental Engineering: New Trends and Challenges (pp. 301-309). Presented at the ITEE 2011, Poznan, Poland: Springer (Heidelberg).

von der Dovenmühle, T., **Mahmoud, T.**, & Marx Gómez, J. (2010). Energy Saving through User Scheduled Load Balancing within Service Oriented Architectures. Proceedings of the ISEE 2010 Conference, Advancing Sustainability in a Time of Crisis (p. 147). Presented at the ISEE 2010 Conference, Oldenburg & Bremen, Germany.

**Mahmoud, T.** (2009). Semantic SOA-Based Model to be applied in Business Environments. CENTERIS 2009 Conference of Enterprise Information Systems Proceedings (pp. 473-482). Ofir, Portugal.

**Mahmoud, T.**, von der Dovenmühle, T., & Marx Gómez, J. (2009). Web Service Validation within Semantic SOA-Based Model. Proceedings of the ICT Innovations Conference 2009 (pp. 295–303). Presented at the ICT2009, Ohrid, Macedonia: Springer (Heidelberg).

**Mahmoud, T.**, & Marx Gómez, J. (2008). Integration of Semantic Web Services Principles in SOA to Solve EAI and ERP Scenarios - Towards Semantic Service Oriented Architecture. Proceedings of the 3rd International Conference on Information & Communication Technologies: from Theory to Applications (ICTTA-2008) (pp. 957-958). Damascus, Syria: IEEE.

**Mahmoud, T.**, & Marx Gómez, J. (2008). Semantic Web Services Process Mediation Using WSMX Concepts. Proceedings of the 20th International Conference on Systems Research, Informatics and Cybernetics (InterSymp-2008) - Engineering and Management of IT-Based Organizational Systems: A Systems Approach (pp. 28-32). Baden Baden, Germany.

**Mahmoud, T.**, & Marx Gómez, J. (2008). Towards Semantic Federated Enterprise Resource Planning (SFERP) System. Discussion paper. Presented at the Modellierung betrieblicher Informationssysteme (MobIS 2008), Saarbrücken, Germany.

**Book Chapters**

**Mahmoud, T.**, Petersen, M., & Rummel, D. (2013). Business Process Integration within Lightweight Semantic-Enabled Enterprise Service-Oriented Architecture. In P. O. de Pablos, J. M. Cueva Lovelle, J. E. Labra Gayo, & R. Tennyson (Eds.), E-Procurement Management for Successful Electronic Government Systems (pp. 181–192). E-Procurement Management for Successful Electronic Government Systems.

**Mahmoud, T.**, Marx Gómez, J., & von der Dovenmühle, T. (2011). Functional Components Specification in the Semantic SOA-based Model. Semantic Technologies for Business and Information Systems Engineering: Concepts and Applications (pp. 277-291). IGI Global.

**Mahmoud, T.**, & Marx Gómez, J. (2010). Applying Semantic SOA Based Model to Business Applications. Enterprise Information Systems Design, Implementation and Management: Organizational Applications (pp. 1-20). IGI Global.

**Mahmoud, T.**, & Marx Gómez, J. (2009). Towards Process Mediation in Semantic Service Oriented Architecture. Handbook of Research on Social Dimensions of Semantic Technologies and Web Services (Vol. I). 780-801: IGI Global.