LAPPEENRANTA UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

MASTER'S THESIS

SERVICE DESCRIPTION CREATION EMPOWERMENT TO MOBILE END USERS

The topic of Master's Thesis was approved by the council of the Department of Information Technology on September 9th, 2009

Supervisors: Professor Jari Porras
Dr.Sc. (tech) Kari Heikkinen

Lappeenranta, April 30th, 2010

Minna Kunttu
Leirikatu 2 A 2
53600 Lappeenranta

Tel. +35840 763 0038
minna.kunttu@lut.fi

ABSTRACT

Lappeenranta University of Technology
Department of Information Technology
Kunttu, Minna

Service Description Creation Empowerment to Mobile End Users

Thesis for the Degree of Master of Science in Technology
2010
119 pages, 25 figures and 5 tables

Examiners: Professor Jari Porras
Dr.Sc. Kari Heikkinen

Keywords: service description, ontology, user generated services, Web Service

Increasing usage of Web Services has been result of efforts to automate Web Services discovery and interoperability. The Semantic Web Service descriptions create basis for automatic Web Service information management tasks such as discovery and interoperability. The discussion of opportunities enabled by service descriptions have arisen in recent years. The end user has been considered only as a consumer of services and information sharing occurred from one service provider to public in service distribution. The social networking has changed the nature of services. The end user cannot be seen anymore only as service consumer, because by enabling semantically rich environment and right tools, the end user will be in the future the producer of services. This study investigates the ways to provide for end users the empowerment to create service descriptions on mobile device. Special focus is given to the changed role of the end user in service creation. In addition, the Web Services technologies are presented as well as different Semantic Web Service description approaches are compared. The main focus in the study is to investigate tools and techniques to enable service description creation and semantic information management on mobile device.

TIIVISTELMÄ

Lappeenrannan Teknillinen Yliopisto
Tietotekniikan osasto
Kunttu, Minna

Palvelukuvauksien luomisen valtuutus mobiililaitteen käyttäjille

Diplomityö
2010
119 sivua, 25 kuvaa ja 5 taulukkoa

Tarkastajat: Professori Jari Porras
TkT Kari Heikkinen

Hakusanat: palvelukuvaus, ontologia, käyttäjien generoimat palvelut, Web-palvelu

Web-palveluiden kasvava käyttö on ollut seurausta pyrkimykselle automatisoida Web-palveluiden löydettävyyttä ja yhteistoiminnallisuutta. Automaattisen palveluiden toiminnallisuuden ja palveluiden löydettävyyden mahdollistaa kerätty informaatio, joka perustuu semanttisiin palvelukuvauksiin. Keskustelu palvelukuvauksien tuomista mahdollisuuksista on herännyt vasta viime vuosina. Loppukäyttäjä on nähty vain palveluita kuluttavana tekijänä ja informaation välitys on nähty yhdeltä sisällöntuottajalta yleisölle tapahtuvana jakeluna. Sosiaalisen verkottumisen kautta on tapahtunut muutos palveluiden luonteessa. Loppukäyttäjää ei enää pidä nähdä vain verkossa tarjottavia palveluita kuluttavana käyttäjänä, sillä mahdollistamalla oikeanlaisen ympäristön ja työkalut, käyttäjä on tulevaisuudessa se, joka tuottaa palveluita. Tämän tutkimus selvittää tapoja tarjota loppukäyttäjille valtuutuksen luoda palvelukuvauksia mobiililaitteella. Erityisesti huomiota kiinnitetään loppukäyttäjän muuttuneen roolin näkökulmaan palveluiden luomisessa. Lisäksi esitellään semanttisten Web-palveluiden tekniikoita ja erilaisia semanttisia Web-palvelukuvaus lähestymistapoja. Keskeinen osa tutkimusta on selvittää työkaluja ja tapoja jotka mahdollista palvelukuvauksen luomisen ja semanttisen informaation hallinnan mobiililaitteella.

ACKNOWLEDGEMENTS

This thesis is a result of my studies at the Lappeenranta University of Technology. The thesis has been done at the Department of Information Technology. I extend my special thanks to my supervisor Professor Jari Porras whose excellent guidance and knowledgeable comments were essential in the completion of this thesis. I would also like to thank Dr.Sc (Tech.) Kari Heikkinen for practical advices for this thesis work. I would like to express my gratitude to study coordinator Susanna Koponen of the Faculty of Technology Management for her support.

My sincere appreciation I give to my sisters and parents Mrs. Hilkka and Mr. Osmo Kunttu for their continued support, patience and financial support. I also wish to thank my friends Liliana and Matti for their continued support during my studies. Most thankful I am to Mubeen, Your moral support was essential to this thesis.

Minna Kunttu

TABLE OF CONTENTS

ABBREVIATIONS

API	Application Programming Interface
ASM	Abstract State Machine
DAML	DARPA Agent Markup Language
DL	Description Logic
DTD	Document Type Definitions
ESSI	European Semantic Systems Initiative
EUD	End-User Development
FLOWS	First-order Logic Ontology for Web Services
FOL	First-order Logic
FP7	European Seventh Framework Programme
GPRS	General Packet Radio Service
HTTP	Hypertext Transfer Protocol
IOPE	inputs, outputs, preconditions and effects
IoS	Internet of Services
KIF	Knowledge Interchange Format
MVC	Model-View-Controller
MOF	Meta-Object Facility
NAICS	North American Industry Classification System
NESSI	Networked European Software and Services Initiative
OIL	Ontology Inference Layer
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Services
P2P	peer-to-peer
PC	Personal Computer
PDA	Personal Digial Assistant
PDDL	Planning Domain Definition Language
PSL	Process Specification Language
QoS	Quality of Service
RDF	Resource Description Framework

RDFS	RDF Schema
ROWS	Rules Ontology for Web Services
RSS	Really Simple Syndication
SAWSDL	Semantic Annotations for WSDL
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	service oriented computing
SPARQL	RDF Query Language
SUIA	Service User Interface Annotation
SWSF	Semantic Web Services Framework
SWSL	Semantic Web Services Language
SWSO	Semantic Web Services Ontology
UDDI	Universal Description Discovery and Integration
UGC	User generated content
UGS	User generated services
UML	Unified Modeling Language
UNSPSC	United Nations Standard Products and Services Code
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
USI	User-Services Interaction
Wi-Fi	High-frequency wireless local area network
WSDL	Web Services Description Language
WSDL-S	Web Services Semantics for WSDL
WSML	Web Service Modeling Language
WSMO	Web Service Modelling Ontology
WSMX	Web Service Execution Environment
XML	eXtensible Markup Language
XSD	XML Schema
XSLT	eXtensible Stylesheet Language Transformations

1. INTRODUCTION

The Web Services are the current phenomena of the Internet world. The ability to create services fast and cost-efficiently, managing services have become crucial in open Web. Web Services bring a high level of interoperability between software applications and enable data sharing across heterogeneous environments. With the rising popularity of Web Services, the researches of Web Service technologies have concentrated on development of Web Service description standards, discovery and composition techniques. The Semantic Web have emerged with aim to enable machine reasoning about Web Service functionalities, leading the way to automate service discovery, invocation and composition with little or non-human involvement in these processes. Currently the standards utilized by Web Services support interoperability at the syntax level. There is a need to embed semantics into services in order to provide machine readability. The ontologies serve interpretations for Web content and ontologically annotated Web Services enable reasoning about their content in machine understandable way. Despite benefits of Semantic Web Services, description process of Web Services, specially adding semantics into Web Services, is time consuming and complex process and therefore remains largely as manual process by professionals. However, the insufficient involvement of users in the construction of ontologies can cause unsatisfying coverage in Web Service semantics [1]. Therefore the user involvement is needed because ontologies have to be based in real world.

The arrival of Web 2.0 applications has made it easy for everyone to use Internet for sharing wide variety of content. In the last several years the volume of user generated content (UGC) has grown rapidly with popularity of Web 2.0 applications. The Web 2.0 applications such as Facebook, Youtube and MySpace all allow users to post photos, share videos and multimedia content. While the content creation by common Internet users has grown and users have considered only as consumer of services and as end users of services, a major challenge is to step forward for rich service environment where to end users is given the ability to create their own services.

Pervasive devices, such as mobile phones and Personal Digital Assistants (PDAs) allow more and more nowadays execution of personal services. In particular, mobile service creation should not be limited to professional computer programmers, rather it should be equally easy for common Internet users to create services. There are numerous projects working on to enable service creation through mobile device by providing service platform such as Simple Mobile Services (SMS) [2], user-centric service creation and execution (OPUCE) [3], m:Ciudad [4] and Service Platform for Innovative Communication Environment (SPICE) [5]. Although these projects surround intensive researches about ontology creation and service semantics on the service platform, there is need to assist the user itself with service description creation. The main goal in the future is to provide tools to non-technical users for service creation. Therefore it is important to study how common Internet users are able to produce themselves ontology based service descriptions, in terms of future opportunities for innovation in service creation. This thesis explores ways to help the end user to create service descriptions by studying existing service description approaches and tools which are suitable to operate on mobile device.

1.1 Objective and scope of the thesis

The main focus of the thesis is to illustrate the role and importance of end user as creator and producer of services. In order to provide for user a semantically rich environment, where services can be automatically discovered, invocated, composed and reused, the thesis emphasis the importance of service descriptions. In the thesis Semantic Web Service description approaches are studied and existing tools compared for service descriptions creation. By comparison is examined how Semantic Web Service description approaches and service descriptions creation tools fit for mobile usage. The main objective is to examine possibilities for service description creation on mobile device and give implementation options for Mobile Semantic Assistant to facilitate the non-expertise user in machine-interpretable semantic service description creation. The scope of the thesis is on Semantic Web Service technologies and ontologies for service markup, focusing on Web Service Ontology (OWL-S) to describe the user generated service on mobile device.

1.2 Structure of the thesis

The users developed role and the ways to empower user driven service creation are discussed in chapter 2. The chapter 2 begins with the importance of user centricity in service creation leading into concept of future Internet of Services. The chapter 3 handles the Web Service technologies and the Semantic Web goal to make services interoperable by ontologies and schemas. Through semantics it is possible to make open and often chaotic Web knowledgeable. In chapter 4 are explored existing Web Service description approaches and the chapter 5 concentrates on Web Service Ontology (OWL-S). The next chapter examines the tools Web Service Ontology (OWL-S) creation and chapter 7 gives view to tools capable to work on mobile device. In the end of the thesis, in chapter 8 is presented the framework of Mobile Semantic Assistant and implementation possibilities for OWL files management on mobile device. The conclusion concludes summarized way the observations drawn of the studies in the thesis.

2. USER ROLE AND INVOLVEMENT IN WEB 2.0

User role and involvement has become important in emergent and future Internet trends. In the Internet user involvement appears by user's participation using various Web 2.0 technologies. The user role and the importance participation can be considered as driving factor of the vision of Internet of Services (IoS). The vision of Internet of Services is seen as a multitude of connected services, which are offered, bought, used, composed and reused in a worldwide network [6]. The focus of recent projects such as Services for All (S4All), Open Platform for User-centric service creation and execution (OPUCE), Service Platform for Innovative Communication Environment (SPICE) and Simple Mobile Services (SMS) is to provide user centric services seamlessly based on user context and personal information in 3G beyond network [7].

The key characteristic of user centricity in Web 2.0 is richer and easier user interaction which is enabled by a social side of the Web that offers for users a platform for collaboration, communication and sharing. Evolving Web scenery emphasize service ecosystem where users play important role as both service consumers and providers. Users collaborate in content and service creation, composition and sharing. Likewise access and usage of services in a personalized way is also important. Service oriented Architecture (SOA) paradigm gives ground for abstracting service and resource specific details and propounds composability, modularity and reusability of contents enabling users to manage, create and share contents and resources loosely coupled way [8].

2.1 Common Internet user's developed role

The Web 2.0 applications provide for ordinary users to share and use in collaboration the content what they have generated. Social media applications such as Youtube, Facebook and Wikipedia allow users who have no technical skills post their text, music, videos and images on the web. The concept of the end user has changed from simple consumer into user who creates the content in the web for others use [9]. User generated content (UGC) and social networking are two of the most significant aspects of Web 2.0 based on

environment to encourage user contribution, collective intelligence and community collaboration [10].

The Web is large information pool connecting people, places, ideas and information in useful and meaningful ways for all things locally and globally. Ubiquitous access is one of the main characteristics of the Internet enabling users to access to information and communicate with other users, every time, everywhere and with any device [11]. In the ubiquitous Internet scene, users play a more central role in the modelling and revise the Web experience upon their needs. As Web 2.0 itself emphasizes on user collaboration and participation [12], users begin perceiving and exploiting the Web as a platform to collaboratively create and share contents and to communicate with each other. Furthermore, users want to decide how to access services and contents, freeing themselves from traditional fixed PCs and adopting heterogeneous wireless devices such as palmtops, smart phones and portable audio players [13]. The vision of Internet of Services highlights multimodality, mobility, context awareness, service orientation and coordination. In this context the evolving Web landscape emphasizes a user centric ecosystem where both services and users play crucially important role. The Web 2.0 phenomenon has brought up that just by giving tools for end users they will create not only contents but applications which could not even imagine in collaborative manner faster than could ever imagine [14]. As innovation is ubiquitous, services should be flexibly created, adapted, and discovered by end-users. Leveraging this concept, next section elaborate user generated as a step towards a broad adoption of mobile service creation. Before moving into user generated services has to categorize what Web 2.0 applications are and what user generated content really is.

The types Web 2.0 application are based on community creation, platforms or tools for broad audience publication and collaborative tools among groups of people [15]. The types of Web 2.0 application are categorized in Table 1. Social networking is based on feature that allow people to express their individuality and grouping with other people with similar interest or social relations creating this way communities [15]. Currently popular Web sites Facebook, MySpace, LinkedIn and Twitter allow users to set up personalized profiles detailing personal information such as interest, hobbies, education

and post text based messages. Web 2.0 mashups are Web pages or applications that take data from other online sources and combine it to create new hybrid services. These kinds of mashups typically are mapping, image and shopping Web sites. The Image photo sharing site Flickr enable users to upload, share, comment on and categorize photographs. In Flickr users can also post photos to a blog and create photo pools. Other good examples of information mashups are Music Portl which allows assemble media featuring from sites across the Web such as Youtube, Flirck and Wikipedia and Unthirsty uses Google Maps and Happy Hour finder to shows for user the nearest place. As blogs represent the personal side of online publishing the wikis are based on collaborative creating, editing and storing of contents by group of users. Really Simple Syndication (RSS) is a tool to alert new postings which allow users to receive update of Web site avoiding this way continuous visit to Web site. [15] iGoogle allow users to create personal homepage by adding free social networking tools such as wikis, blogs, bookmarks and news [14]. Google calendar allow users to create shared calendars and alerts of appointments can be sent as Short Message Service (SMS) message or email to group of users [15].

Table 1. Types of Web 2.0 applications [15].

Type	**Example**
Communities that aim to unify their users by means of a common ideal such as social networking or knowledge sharing	Facebook, MySpace, LinkedIn, Twitter
Platforms or tools that help users create and share content with a broad audience. Mashup platforms let users retrieve content or functionality from arbitrary sources, mix it with other resources and expose it for further reuse by other applications	Flickr, Music Portl, Youtube, Wikipedia, Unthirsty
Online collaboration tools support users in collaboratively performing certain tasks, such as maintaining time schedules or processing text online	Wikis, Skype, iGoogle, RSS

In traditional way UGC is categorized as the contents of blog sites, social networking sites, wikis and discussion forums. This classical way is quite limited picture of content creation, because the content is primarily delivered and displayed to users as part of the Web site it belongs to and main purpose is to be directly consumed [9]. In Table 2. is

represented the categories of user generated content according the category type of the content such as content itself meaning here text, image, video and audio related to information web sites, social networking or consumer media. Another category is organization and structure creation by bookmarking and tagging. The third category is about creating functionality into Web sites with the help of different applications.

User created tags and bookmarking is a way for organizing the content. Bookmarking is done for personal reasons as the user makes link into own list of links [16]. Tagging is creating keywords of assigned information for image, file or internet bookmark. A tag is just a keyword, added metadata of the content without meaning or explanation of the item [16]. On a Web site, where many users tag multiple items the collections of the tags becomes a folksonomy which is type of distributed classification system [12]. Programmatic access of web sites applications has enabled feature that users can create a mashup to combine information and complementary functionality from other Web sites or Web applications. Web mashup server lets users to collect, connect and mash up any content of the Web.

Table 2. Categories of user generated content. [9]

Category	Example	Usage	Object layer
Content: text, image, video, audio	Photo, audio and video platforms, wikis, blogs	play, display, show	Platform, item
Organizationing/ structure	Bookmarks, Tags, links	hyperlinks	Platform, item
Functionality	Mashups	execute function	Platform

The social bookmarking, tagging, folksonomies and mashups indicate that Web contents and services are no longer immutable and preconfigured by third-party vendors and for ubiquitous innovation it is needed to involve users in the process of service creation and diffusion [13]. Furthermore, as technology improvements provide heterogeneous wireless and mobile devices, there is a need to let users to create services and contents available on the Web in more versatile and customized ways.

2.2 From user generated content to user generated services

Users are not only service consumers but users can be seen as service producers. In this matter have to move forward the concept from user generated content to user generated services. The next step in the UGC trend is user generated services (UGS). User generated services (UGS) let end users create their own personalized services using graphical tools, such as Yahoo Pipes and iGoogle [17]. Even though these early approaches towards end user service creation on the web exist, because of user interface issues and different architecture they cannot be directly transferred to mobile environments. Moreover as they require some programming skills, they are too complex for a widespread use and spontaneous service creation [18].

Whereas the Web 2.0 attracts academic and industrial world and breaks down the boundaries between producers and consumers, the research area of End-User Development (EUD) has struggled to make its objectives and techniques known to the world and its directions will need to be explored in the context of software services, open communities and mobile usage patterns [19]. The End-User Development (EUD) research community considers different approaches for end users to create software and studies concepts such as tailoring, configurability, end-user programming, usability, visual programming, natural programming and programming by example. These concepts form a robust base, but they need to be better integrated, and the synergy between them more fully exploited [20]. As the current end-user development methods are still at an early stage and are diversified in terms of terminology, approaches and subject areas considered, the methods can benefit from semantic technologies, for example by using annotations or rules to describe the behaviour of functional blocks [18].

The first decade of Internet use was dominated by strong separation of designers and users. The EUD is focused on the challenge of allowing users of software systems who are not primarily interested in software development as such, for example professional designers to modify, extend, evolve, and create systems that fit their needs [19]. The computational systems modelling some certain environment are never complete and evolve over time because the world changes with new requirements and skilled domain

professionals change their work practices over time as their understanding and use of a system will be very different after a month and certainly after several years [20]. The need for end-user development is essential, because if systems cannot be modified to support new practices, users will be locked into existing patterns of use [21]. The world of Web 2.0 and EUD research indicate an emerging change of new production trend in social, educational, civil and professional life has established new levels of adaptation, including the middle ground models from consumer and producer separation such as prosumers and professional amateurs [21]. The types of end users and the impacts and implications of media and technology usage are presented in Figure 1.

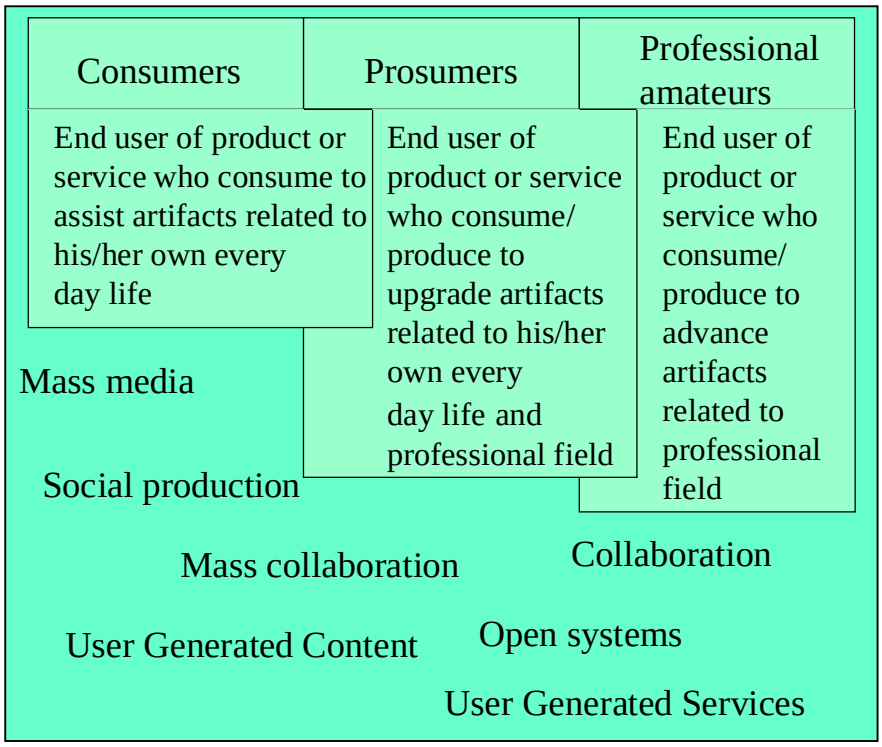

Figure 1. Types of end users.

The consumers are common Internet users who use Web Services to facilitate their life or for entertainment. Even tough the common Internet users are non-programmers, they are not anymore just passive consumers of published information as they participate to content creation by social production and mass collaboration [22]. Technology sophisticated prosumers are comfortable with the technologies they are used to and adapt new technologies quickly. These users modify artifacts to their own requirements by

experimenting, exploring, building, tinkering, framing, solving, and reflecting. The professional amateurs are innovative, committed and networked technology users working with professional values. The professional amateurs are a new social hybrid as their activities are mixed by the work and leisure, professional and amateur, consumption and production [21]. The definitions of end users are surrounded by the importance of open systems, user generated content and services and the move from guideline development to situated knowledge as showed in Figure 1. The open system have unfinished characteristic with admit of ongoing change is desirable rather than a to-be-avoided attribute characteristic. The user generated content can be creation of the artifacts with existing tools or changing the tools, for example writing a document with a word processor or writing macros to extend the word processor as a tool. In specific software environments such as open source software, the content is subject to the additional requirement of being computationally interpretable [21].

For the empowerment of user generated services the research field of EUD acknowledge the importance of situational applications where users are enabled to create an application to resolve and support some specific task. Leveraging this, in the context of user generated services is important to move from making systems just easy to use to making systems that are easy to develop. As on the Web mashup services enable to perform a specific form of end user development, while in the mobile domain current service creation approaches are very limited in terms of functionality or not customized to mobile devices and usage scenarios concerning user interfaces, capabilities and context information [23]. Going beyond the Web 2.0 trend that internet technology facilitates creativity, information sharing and collaboration among users, the future Internet of Services aims to enable users to adapt, customize and control services according their needs [8]. Next section explains the vision of the future Internet of Services and describes the methods to enable user driven service creation.

2.3 Enabling user driven service creation

Service oriented architecture (SOA) and service oriented computing (SOC) approaches allow different entities to be made as services [24]. The vision of these approaches is to

be able compose these entities into applications. This requires that services have to be loosely coupled as if there are changes in the environment it is possible for application to dynamically recompose itself to respond the new needs [8]. Service oriented architecture (SOA) is software architecture based on major concepts of application front-end, service, service repository and service bus. SOA is paradigm of encapsulating application logic in services with a uniformly defined interface and making these publicly available by the use of discovery mechanism [24]. The key advantage for SOA is the ease for making changes, because of the support of composition and coordination of autonomous and sharable services in a loosely coupled manner. SOA definitions range from technology driven approach to a new management approach on how to run the whole enterprise. The potential of SOAs is emphasized especially by merging business requirements and Information Technology (IT) infrastructures. For business requirements and IT infrastructures SOA enables independent development by separate teams, each one with its own delivery and maintenance schedule [24].

The Web 2.0 sites and applications are created to bring convenience and simplicity for end users and to improve user relevance and service usability. However, current service front-ends are far from user impact. Because of centrally controlled creation, the applications are still based on monolithic, inflexible and unfriendly user interfaces [25]. Service front end is a program interface and a front end application is one that users interact with directly. As front end refers to user graphical interface, back end term is used to characterize the server side or execution system at the back of the service [26].

2.3.1 The projects realizing the vision of the Internet of Services

The European Seventh Framework Programme (FP7) 2009-2010 is dedicated on research topic Service Front Ends (SFE) in the area of Service Architecture and Platforms for the Future Internet. The Services Front End (SFE) Working Group in FP7, together with the User-Services Interaction (USI) and Networked European Software and Services Initiative (NESSI) Working Group are focused on technologies enabling users with different level of expertise to search, use, compose, configure and share services with users' mobility, device - and context aware technologies [27]. Several projects are

working in the area of Service Front Ends with aim to enable users to adjust, customise and control services according to their actual needs [8].

The project Fast and Advanced Stroryboard Tools (FAST) extends the notion of gadgets allowing non-expertise users to interact with back end services. The main objective of the FAST project is to create visual programming environment to facilitate the development of composite applications from complex business processes of an enterprise that rely on traditional back-end semantic Web Services [25]. The project of Metropolis of Ubiquitous Services m:Ciudad aims to give users ability to create and provide on to fly small services so called microservices containing useful information to other remote users from their mobile devices [28]. The m:Ciudad project goal is to explore tools needed to allow users with mobile device to become a service provider and mobile platform architecture in order to be simple to use as to be same time efficiently provide semantics aware microservices [29]. The Open Pervasive Environments for migratory iNteractive services (OPEN) project strives for transparent interaction change between different platforms with same application in new context of use [8]. While above mentioned projects concentrate on end users, the project Service annotations for user interface composition called as ServFace focuses providing methodology and tools for software developers in order to develop user interfaces for applications in service oriented manner [8].

2.3.2 Service oriented architecture behind service usability

The design philosophy of Service oriented architecture (SOA) is a hybrid method to build applications by connecting different building blocks of services together in a loosely coupled way [8]. SOA allows software systems work together even they were developed by different organizations as corporations setup loosely coupled electronic business transactions by adhering to common standards for the description of their services. SOA based development of applications reduces development time [6] and obviates recompilation of the application for each single designated platform. With thanks to availability of reusable application building blocks can be avoided to rewrite whole service [30].

Despite SOA has gained popularity, it is still primarily adopted only in large corporations for internal integration and less for external consumption [8]. Companies still seem to be hesitant to expose their internal business services thorough the Web. The situation is changing towards open scalability and for example companies such as Yahoo, Google and eBay are exposing their IT services over the Web in order to be able to create new mashup applications [31]. While SOA is still enterprise specific solution, the part of FP7 project SOA4All aims to establish a service delivery platform for all types of and needs of Web Services and their functionality needs such as sensors, aggregators or hardware resources [32]. SOA4All architecture is built around main four components, which are SOA4All Studio for user front ends, Distributed Service Bus for communication infrastructure, Platform Services for expose functionality of published services and Business Services for third party Web Services and light-weight processes [32].

The SLA@SOI is one core pillar project of NESSI vision [8]. NESSI is the European Technology Platform dedicated to design and implement coherent and consistent open service framework for service based systems [33]. The main goal is to move from product-oriented infrastructure into service-oriented infrastructure (SOI). The establishment of the business relationships and the business, software, infrastructure chains require to support expanding service based economy. Therefore, to be able to provide certainty of quality offered by each service, be it business, software or infrastructure for all service consumers of all layers, a method for modelling agreed service certainty is needed [8]. Traditionally contracts are made for agreed service certainty. Service Level Agreements (SLAs) are such contracts in the digital world. The SLA@SOI project stands for the management of Service Level Agreements (SLAs) and implementation of SLA management framework that works in the service-oriented infrastructure (SOI) [8]. The Figure 2 shows above described projects and research areas to enable the vision of future Internet of Services (IoS).

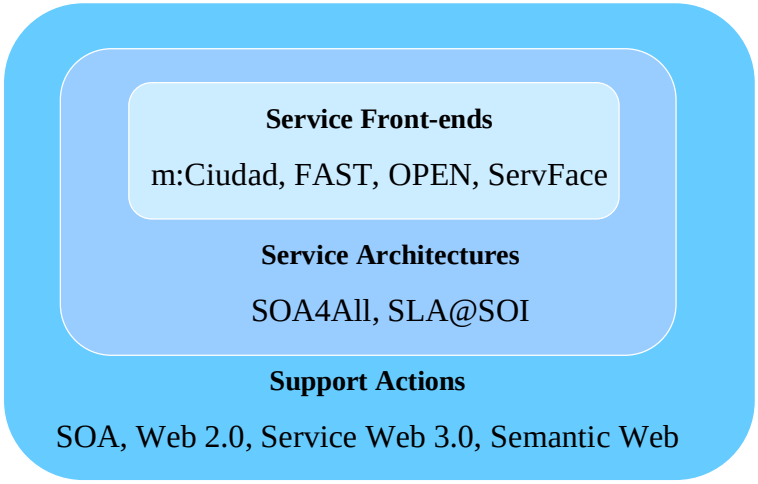

Figure 2. The projects and research areas for empowerment of IoS [7].

In the establishment of future Internet of Services (IoS) the project of Service Web 3.0 is a step towards above mentioned goals to provide solutions to integration and search that will enable SOA on worldwide scale [34]. Empowerment of new service economy requires semantic descriptions of services and their interactions that a shift from user to system navigation of services is possible. Semantic descriptions are needed for easily identifier services and that they could be composed seamlessly [35].

2.4 Challenges of user generated services

The current development of services demands more expertise in design because of the diversity of the technologies as there is various device platforms unique to each other [30]. Service creation and deployment possesses closed nature as operating systems and devices are designed in such a way that services can only be deployed on certain group of devices. In addition, the current communication between services possesses same closed nature as different services from respective providers work only in their own defined areas and limits [30]. The shift has to include moving from guidelines, rules and procedures to exceptions, negotiations, interventions and workarounds to complement and integrate entrusted expert knowledge with practice based and situated knowledge [21]. The challenge is to replace the closed, rigid and centrally controlled approaches by open, flexible, networked and distributed models. The emerging service economy needs so called pulling approaches that are user driven, flexibly accommodate resources of

diverse service providers and consumers, are open ended and are developed continuously based on learning and changing needs of different participants [6]. In the future service development is essential that users are pertained to the development. The Figure 3 shows present and future aspects of the shift to move from rigid and closed systems and guideline development to open-ended systems and innovative situated based development.

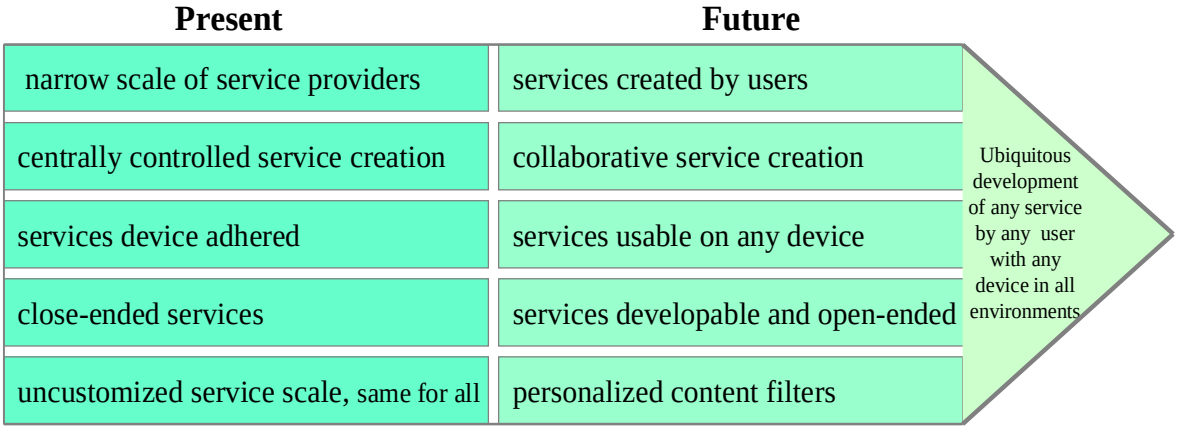

Figure 3. Present and future characteristics of service development.

The key issue in the user generated service enablement is design for unexpected use. Enabling highly intuitive, important is not for what some service is designed to improve user experience, the importance is in the design for unexpected use to affluence the user. In this context is essential easy move from application to another regardless the device. For non-programmers is necessary allow to use natural language. The users describe in natural language as they see the things and let the systems figure it out.

The dynamic content creation environment Web 2.0 which is currently turning into Web 3.0 called as Social Semantic Web will use rich domain knowledge and document level metadata to organize and analyze social media content. Essential is how the Semantic Web can enrich the Social Web which includes not only data or Web pages and the links between them but also people and connections among them and more importantly the connections that people make with data [36]. This data on the Web in the vision of Semantic Web is made more meaningful through labels for example marking up, tagging or annotating that follow some reference model being common dictionary, taxonomy,

folksonomy or ontology that represents a specific domain model. Ontologies are similar to taxonomies but whereas taxonomy indicates only class and subclass relationship, the ontology describes a domain completely. The ontologies have emerged from the area of artificial intelligence and have begun to be applied to the Semantic Web augmenting Web information with a formal, more specifically machine-interpretable representation of its meaning [37]. A direct benefit of this machine-interpretable semantics would be the enhancement and automation of several information management tasks, such as discovery or data integration.

In the UGS domain, the contents are produced and added in the Web fast making discovery process key issue allowing users to find relevant services suited to their needs. Categorization and browsing based on tags also work for services, but they can be tedious and depends on how users identify and express their needs. The tasks become more difficult when trying to find services in the first place and needs much more manual work when trying look does the service suit to the needs [17]. Annotations with ontology that represents a specific domain model, make Web based documents and consisting data machine understandable as well as easier to analyze and integrate them. Powerful reasoning over annotated data is allowed when applications use ontology, rules that range from simple to complex, whether they are explicitly stated or inferred from the ontology class properties and relationships [36]. The Figure 4 illustrates the changing Web, in which information is given well-defined meaning, enhancing computers and people cooperation.

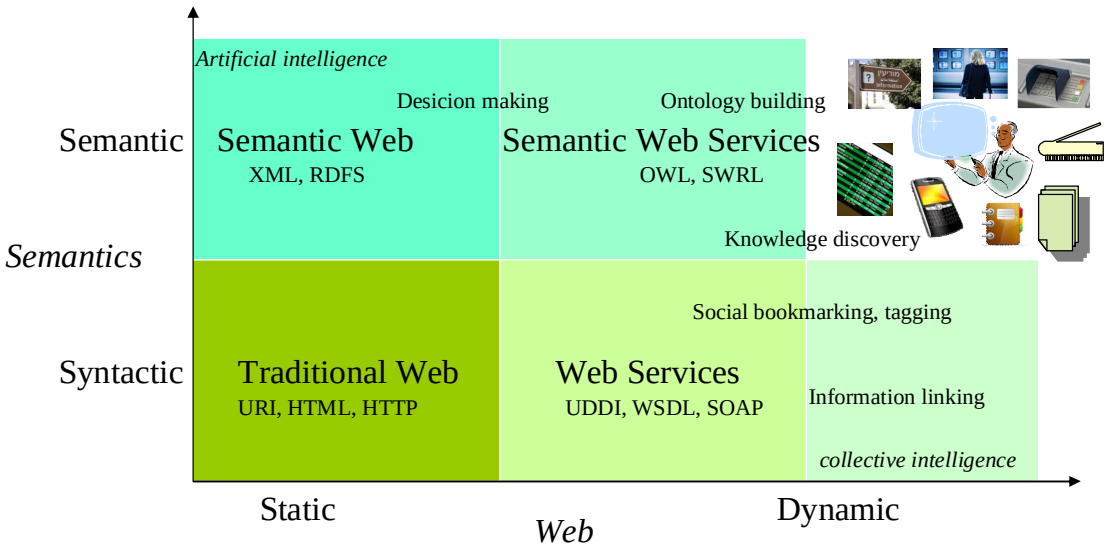

Figure 4. The Web as changing information pool

In Figure 4 the technologies are classified according to their level of expressivity at Semantics dimension and systematic information organization at Web dimension as showed. Large scale systems like the Web is very dynamic, while ontological commitment would require stability and formality [37]. It seems difficult to go forward along both dynamic Web and formality dimensions simultaneously. Like expected, the upper right corner is left empty for future inventions as showed in Figure 4.

3. SEMANTIC WEB SERVICES

The goal of Semantic Web Services is to accomplish the usefulness of Web Services by adducing computational machine readable representation of the service [38], which is referred here to as service description. The concept of Semantic Web Services has evolved from ubiquitous infrastructure of the Web in effort to change conventional Web Services into intelligent Web Services that enable automatic service discovery using prescribed semantic markup of Web Services and ontology conscious that are context aware agents and search engines [39]. In order to deploy intelligent multiagent systems on the open, unregulated and chaotic information pool environment as the Web is, it is needed to augment Web Service description through semantic annotation. The next section explains the importance of Service Oriented Architecture (SOA) in order to create semantic services.

3.1 Web Services and Service Oriented Architecture

The Web Services are activities that allow end users and software agents to invoke services directly. In the traditional Web model users just follow hypertext manually by browsing, but in the Web Services model users invoke tasks that facilitate some useful activity. These tasks to facilitate activities are based on remote procedure calling which support for example meaningful context based discovery of resources, fusion of similar information from multiple sites or commercial activities such as course advertising for distance learning [39].

The Service oriented architecture can greatly enhance the traditional model in the case of development of Web applications, since the SOA can built a service without of knowledge of client side, so any client system can avail those services. This is possible by fact that every Web Service is described with service description language and dynamically discovered by applications that need to use it and invoke through the communication protocol defined in its interface [39]. The Service oriented architecture is presented in Figure 5. The central component in Figure 5 is the service directory built as dynamically organised information pool pertaining to various services. In the first phase

of Figure 5 sequence the service advertises itself in directory as Web Services are assumed to advertise their availability and readiness to the directory. Therefore an agent can find out about the available services by looking up the directory and decide whether to automatically invoke a suitable service on the user's behalf or suggest that the user interacts with the service. As shown in Figure 5 at phase 2 the client application is allowed to query the registry for service details and at phase 3 interact with the service using service details.

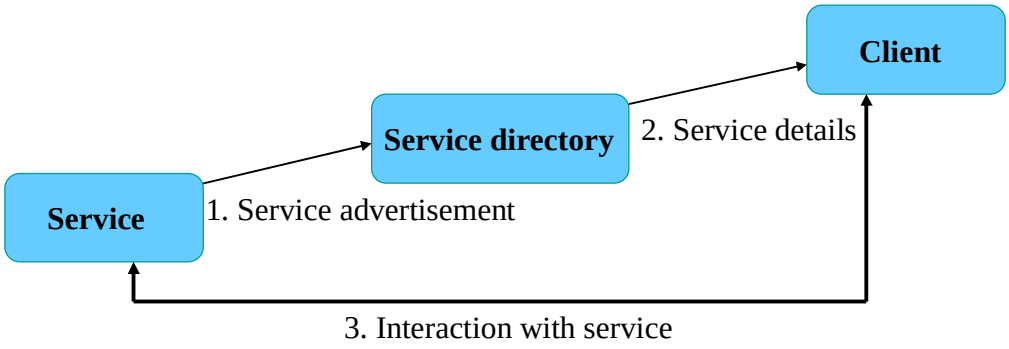

Figure 5. Service Oriented Architecture [39].

The characteristics of SOA are of aspect of service orchestration, loose coupling of software components in the distributed manner and the standardization of interface descriptions. Thomas Erl focuses on IT – business process development and therefore there can draw three main aspects of an SOA which are service descriptions, business processes and service management [40]. A proper description of services is a fundamental precondition for a service discovery, invocation, composition and service management. Modelling functional and non-functional information in a machine-interpretable and semantically enriched way is a basis for a service level management and service relocation [41].

3.2 Semantic Web

The vision of Semantic Web, stated by Berners-Lee is that computers will be able to reason about Web data and leverage the right sources of data for the tasks they are attempting to perform [42]. This requires the ready availability and acceptance of

ontologies about the data [43]. Web Service ontologies bring intelligence to Web Services enabling automatic service discovery, invocation, composition and monitoring without that user is required to go through Web pages manually and execute the service to see whether it satisfies needed requirements and fill forms of service manually [44].

As discussed in previous sections, the research focus of Semantic Web is to create an environment where software systems are enabled to have automatic and dynamic interaction. The Semantic Web aims to bring machine understandable information of the Web, which currently is available only in a human readable form [43]. By extending Web Services with semantic description, our information access based on a higher lever of services provided in the Web will significantly change. The services which vary from selling products such as book to graphics rendering could be advertised and discovered automatically. For example a company needing a service could locate a provider for this service automatically and set up a short term business relationship by making and receive the service in return for a payment in no time. By describing services and with infrastructural capabilities to discover services and enable their interoperation, several services could be found and combined into more complex service and if one component service is unavailable, a replacement could be found inserted rapidly in order that complex service could be still provided [45]. The Web Service technology allows the description of an interface in a standard way, but express nothing in machine-interpretable form what the software system does or what is needed and how is used to interact with it [38].

The Semantic Web technologies fit into a set of layered specifications constructed of Semantic Web languages. Formal specifications of Semantic Web are Resource Description Language (RDF), RDF Schema language and Web Ontology language (OWL). The query language of RDF (SPARQL) enables to extract information of decentralized collections RDF data [44]. The Semantic Web languages rely on Extensible Markup Language (XML) syntax [46]. Above described languages are presented in next sections.

3.3 Ontologies and schemas

The ontologies are designed by artificial intelligence community for formal knowledge representations of concepts and their relationships [44]. An ontology is defined as a formal specification of conceptualization consisting of a collection concepts, properties and interrelationships between concepts that can exist for certain information domain [38]. The main reasons for using ontologies are the communication purposes to facilitate information exchange, to infer internal structure of domain and reusability of domain information. Ontologies facilitate the exchange between cooperating parties by enabling shared understanding about domain [44]. On internal structure and operation of an implemented system can draw inferences with help of ontology. With help of ontology, intelligent agents can extract and gather information from multiple parties to provide a comprehensive view of domain's knowledge so that information can be reused. Ontologies can characterize knowledge in an application or domain specific manner performing domain ontologies or domain independent manner performing an upper ontology [47].

Machine-interpretable semantics of data is based on Web languages such as Extensible Markup Language (XML), Resource Description Framework (RDF) and the Ontology Web Language (OWL) [47]. The Semantic Web Services machinery could be employed to automatically tackle interoperability issues between ontologies at the conceptual level [43]. Meta-languages used for representing ontologies are XML Schema and RDF Schema [44].

3.3.1 XML and XML schema

The Extensible Markup Language (XML) addresses the data representation by focusing on the syntax rather than semantics of documents exchanged on the Web [47]. XML is a meta-language designed to transmit data and meaning of the data in machine and human readable form, this way XML provides a standard way to define the structure of automatic processing documents [44]. One of the most time consuming challenges for developers has been to exchange data between computer systems and databases containing data in incompatible formats. Converting the data to XML, information

exchange can be done between incompatible systems. On the other hand, XML is just wrapped information in XML element tags and it does not define what is needed to be done with data [35]. This interfere a potential service client from using certain Web Service because the need remains for a some person to be involved to interpret both the XML descriptions of the Web Services as well as the XML descriptions of the data that the services can exchange [47]. The XML descriptions of the data that the Web Services can exchange are represented as XML schema. XML defines rules to which every document must match in order to state that a document conforms to certain document type. The structure and to define constraints on the syntax of XML documents can be specified by Document Type Definitions (DTDs) or XML schema (XSD) [35].

XML Schema definitions are themselves XML documents. XML Schemas provide a rich set of datatypes that can be used to define the values of elementary tags. The schema definition for elementary tag is itself an XML document, whereas DTDs would provide such definition in an external second language. With XML schema author can express syntactic, structural and value constraints relevant to the document elements. For example, the element content type such as string, decimal, Boolean or float is specified by the schema [44]. XML schemas provide the namespace mechanism to combine XML documents with heterogeneous vocabulary [46].

Although XML provides facile syntax for encoding exchange data between computers by prescribing the data structure using XML schema, it does not contribute much to the semantic aspect of the Semantic Web as it does not provide machine interpretation of the Web data in advance [46]. The Resource Description Language (RDF) and RDF Schema are used as standard model to describe facts about Web resources to provide machine interpretation of the Web data [44].

3.3.2 RDF and RDF schema

The Resource Description Language (RDF) is designed for representation and processing of metadata about information resources on the Web [35]. As the Resource Description Language (RDF) defines a model for describing relationships among resources in terms of uniquely identified properties and values, it enables description of basic ontologies

[44]. The Resource consists of three objects as subject, predicate and object. The Resource itself is represented as subject, predicate defines the property name for the subject establishing relation between subject and object and a statement is referred as an object defining a property value of the subject [44]. A subject of RDF is either Uniform Resource Identifiers (URIs) or a blank node. Uniform Resource Identifiers (URIs) identify resources and the format is based on XML notation [47]. Objects can also be data values, called as literals which can be either plain or typed literals. Literals consist of lexical form and value. The lexical form is a Unicode string and indicates the syntax of the literal. The lexical form and the value are identical and may have a XML language tag as well in plain literals. Typed literals have a third part, a datatype URI which determines the range of acceptable values and the mapping from the lexical form into the value space [48].

The relationships between RDF resources, property names and property values are represented as labelled graphs built of collections of triples [44]. A single RDF triples is resource-property-value (OAV) triplet [48]. The labelled graphs are called RDF models [44]. An RDF model itself does not define the semantics of any application domain or makes assumptions about certain domain. To define features and semantics of domain meaning here ontologies, it is required to have additional facility to encode ontologies of domain. The RDF Schema (RDFS) encodes ontologies, whereas RDF itself describes just instances of ontologies [39].

RDF schema (RDFS) extends RDF by defining a class and property system [35]. RDFS provides XML based vocabulary to specify classes and their relationships and defines properties to associate them with classes and this way enables creation of taxonomies [39]. Using RDF and RDFS it is possible to describe complex properties of resources such as Web Services and complex relations between these resources. Since the rules for writing such descriptions are well defined the descriptions can be parsed automatically to extract information about resources [35].

RDFS can be used to develop basic ontologies, but automated discovery and execution of Web Services requires more expressive descriptions. To answer this need, W3C

proceeded to develop a specification of the Web Ontology Language (OWL) in 2004 [49]. RDF and RDFS are the base models and syntax for the semantic Web. On the top of the RDF and RDFS is possible to define more powerful languages to describe semantics. The most prominent markup language for publishing and sharing data using ontologies on the Web is the Web Ontology Language (OWL) which is described in the next section.

3.3.3 OWL

The Ontology Web Language (OWL) is designed to facilitate greater machine readability of Web Service than XML, XSD, or RDFS offers by providing additional vocabulary for term descriptions [35]. OWL is revision of the DAML+OIL which is a semantic markup language for Web resources that extends RDF and RDFS. DAML+OIL is combination of previous efforts as DARPA Agent Markup language (DAML) and Ontology Inference Layer (OIL). The Figure 6 shows OWL sublanguages layered structure and it's heredity to DARPA Agent Markup Language (DAML) program and Ontology Inference Layer (OIL) effort combined to deliver the DAML+OIL ontology specification language [50].

Like its predecessors, OWL vocabulary includes a set of XML elements and attributes with well defined meanings. An important DAML+OIL heritage is OWL's layered structure. The OWL vocabulary is built on top of RDFS vocabulary [35]. An important feature of the OWL vocabulary is its richness for describing relations among classes, properties and individuals [39].

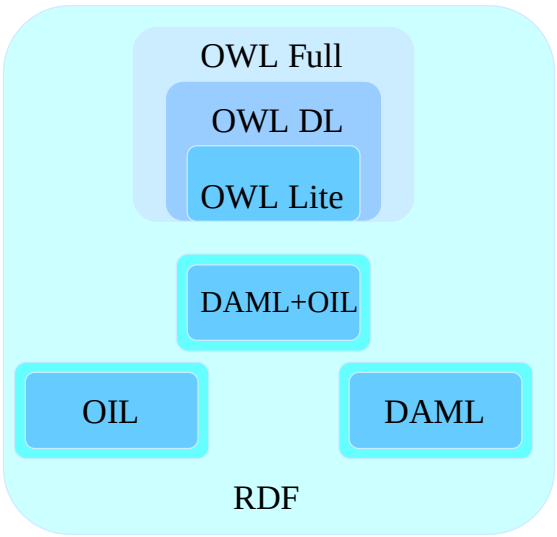

Figure 6. RDF influence to OWL [39].

OWL is combination of three sublanguages, built on top of each other. OWL Lite is intended to support the building simple classification hierarchies [39] and extends RDFS to include simple constraints [44]. The ability to specify constrains is restricted, because cardinality of properties is limited to have only 0 and 1 values, meaning that properties cannot have multiple values [49]. The next level sublanguage OWL Description Logic (DL) reflects foundation of its predecessor DAML+OIL [39]. OWL DL requires strict type separation between classes, properties and instances and also guarantees that all conclusions are computable and will finish in a finite time [39]. OWL Full does not have any limitations between classes, properties and instances and supports users who want unlimited expressiveness and syntactic freedom of RDF. OWL Full does not have any restriction on the cardinality values of properties, but it does not guarantee computational comprehensiveness and inferring. OWL Full can be viewed as an extension of RDF and OWL Lite and OWL DL are restricted ones of RDF [39].

3.3.4 SPARQL

The query language of RDF (SPARQL) is not intended for ontology specification unlike RDFS and OWL [48]. SPARQL can be used to extract information from RDF model in the form of URIs, blank nodes and plain or typed literals. SPARQL can extract also RDF subgraphs and construct new RDF graphs based on information in queried graph models

[39]. SPARQL queries match graph patterns against the target graph. A graph pattern is the only part of a SPARQL query that is sensitive to the semantics of the queried document and conceptually speaking the only part that interacts with the data [48].

The simplest graph pattern is a slight but significant generalization of RDF OAV triplet. The graph patterns may contain named variables in place of nodes called in RFD resources or links called in RDF as properties [39]. The simplest graph pattern is called basic graph pattern (GP) which can be combined in group graph patterns or union graph patterns. The group graph pattern is a set of graph patterns {GP1...GPn}[39]. SPARQL allows also the definition of optional graph patterns. An optional graph pattern is a made of a pair of graph patterns where the second pattern modifies solutions of the first, but does not fail matching of the overall optional graph pattern [51]. The dataset specifies an RDF data model over which SPARQL query is executed. The dataset may be a set of graphs, where the default graph is always present but does not have a name and a set of optional named graph each of which is identified by an URI. The solution modifiers allow modifications of the solution sequence, for example ordering the solutions, avoiding repetitions or limiting the number of solutions. A SPARQL query can have different result forms, among which the SELECT, ASK, and CONSTRUCT result forms [51]. The SELECT result form defines a set of variables, beginning with “?” or “$”. The ASK form simply checks if the query pattern has some solutions over the provided dataset and returns yes or no answers. The CONSTRUCT form returns an RDF graph specified by a graph template as the result RDF graph is formed by taking each query solution, substituting for the variables into the graph template, and combining the triples into a single RDF graph by a set union [51].

4. SEMANTIC WEB SERVICE DESCRIPTION APPROACHES

Current Web Service technologies such as Web Services Description Language (WSDL) [52], Universal Description, Discovery and Integration (UDDI) [53] and Simple Object Access Protocol (SOAP) [54] provide the way to describe at syntactic level the code running in a distributed environment on the Web. As discussed in previous chapter, the technologies provided by the Semantic Web are working towards a machine-interpretable Web and interoperable Web Services that intelligent agents can discover, execute and compose automatically. Web Service technology provides a standard and widely accepted way for defining automation of Web Service discovery and grounding. For example Universal Description, Discovery and Integration (UDDI) specify the structure and access to service registries [53] and Web Services Description Language (WSDL) defines an XML description of programmatic access to Web Services [52].

Web Service technology allows the description in a standard way, but says nothing in machine understandable form about what the software system does or what sequence of messages is used to interact with it. This lack can over come through the process of semantic markup of Web resources by adding information to the resources without changing the originals. Web resources must contain semantic markup descriptions which use terminology that one or more ontologies define. Ontologically annotated Web resources enable reasoning about their content and advanced query – answering services [39]. With ontologies it is able to advertise services in machine understandable way and allow more sophisticated discovery of services than is currently possible with UDDI. While in UDDI the registry content querying is based keyword based search, the ontology allows definition of service capabilities which is needed for advanced query – answering services [47].

One problem is the location of Web Services to make available a specific capability required by a potential client. The UDDI specification provides for a registry of service descriptions, but the descriptions are not formalized and are only useful when interpreted by a human reader [47]. Automated data transformation is another issue, because if the data definitions used by the Web Service do not match those used by the potential client,

a transformation is required and this is typically encoded using the eXtensible Stylesheet Language (XSLT) [47]. For communication between client and service is required to have two XSLT style sheets, one for each direction. As there is no automated means for interpreting the semantic data, both style sheets must be hand-coded [47].

As the WSDL definitions are fundamental in allowing potential users of a Web Service to determine how to interact with that service, it is crucial that this information is explicitly defined. In WSDL the recommended technology for the definition of data types is XML Schema [35]. The WSDL schema itself provides the means for describing behaviour of service. The drawback is that as aspect of interface description (WSDL) results in unambiguous meaning, computer systems are blocked from being able to interpret and reason over the description. This restricts the opportunities for automated Web Service discovery, composition and invocation and leads to a strong motivation for semantic markup [47].

The communication protocol Simple Object Access Protocol (SOAP) underlies all interactions among Web Services [54]. SOAP uses XML for exchanging information in structured and typed manner between the Web Service and its clients over native Web protocols such as Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP) [54]. SOAP is stateless and one-way message exchange protocol for specifying the formats that XML messages should use, the way in which they should be processed, specifying a set of encoding rules for standard and application defined data types and specifying convention for representing remote procedure calls and responses [35]. These messages are used as an envelope where the application encloses whatever information needs to be sent. Each envelope contains header and body. The header is optional and any additional information such as transactional interaction or security goes to header. The body is mandatory and contains the core information [35]. The major drawback of SOAP is the lack of semantics within the message. SOAP ignores the semantics of the message being exchanged through it and relies instead on related WSDL to provide message semantics [35].

Above mentioned basic Web Service technologies are deficient in capabilities to describe semantics of code fragments for realizing complex workflows and business logics which is major requirement for service recognition, configuration, composition, comparison and automated negotiation. For that lack number of approaches has been proposed. The next sections present the four leading ontology based approaches for representing Semantic Web Services. The focus is on structured semantic service descriptions such as Web Service Semantics for WSDL (WSDL-S), Semantic Annotations for WSDL (SAWSDL), Semantic Web Services Framework (SWSF) and Web Service Modeling Ontology (WSMO). The OWL based Semantic Markup for Web Services (OWL-S) is handled in chapter 5 more profoundly. Each of these approaches allows involving Web Services with ontology based semantic description of what the service actually does.

4.1 WSDL-S and SAWSDL

The explicit semantic annotation of the services is a very important step towards the exploitation of the full potential of the service oriented approach [55]. Different approaches try to provide this by adding semantics to conventional service descriptions like WSDL-S [56]. The purpose of semantic markup is to describe services capabilities, inputs, outputs and diverse constraints in formal language. Current standards on the Web Service area like UDDI and WSDL can form the basis for a service description and retrieval solution but are not sufficient [47]. A WSDL description provides the information that is needed to invoke a Web Service in a syntactically correct manner and can therefore be used for the technical integration of available services. WSDL does not provide a standardised way to describe precondition and effect of a service call and the semantics of the terms used to describe input and output of the service can only be included in the description by using appropriate terminology [47].

The METEOR-S group has developed a lightweight approach for augmenting WSDL descriptions of Web Services with semantic annotations called WSDL-S [57]. WSDL-S enables semantic descriptions of inputs, outputs, preconditions and effects of Web Service operations by taking advantage of the extension mechanism of WSDL. In contrast to the Web Service Modeling Ontology (WSMO) and OWL based Semantic

Markup for Web Services (OWL-S), WSDL-S does not specify ontology for the definition of Semantic Web Services nor gives any information how the referenced ontology should look like [47].

The Semantic Annotations for WSDL (SAWSDL) is based on a simplified form of WSDL-S. SAWSDL is bottom-up approach to Semantic Web Services where elements in WSDL documents are provided with semantic annotations through attributes provided using standard valid extensions to WSDL. The approach is agnostic to the ontological model used to define the semantics of annotated WSDL elements [58]. From SAWSDL perspective, the annotations are valued by URIs. SAWSDL, like WSDL-S is targeted at WSDL version 2.0 but it is also possible to use with WSDL version 1.1 with an additional non-standard extension [58]. While WSDL-S specifies the attributes for modelReference, schema mapping, precondition, effect and category, SAWSDL confines itself to attributes of model reference and two specializations of schema mapping, namely lifting schema mapping and lowering schema mapping. The model reference attribute can be used to annotate XML Schema complex type definitions, simple type definitions, element declarations, and attribute declarations as well as WSDL interfaces, operations, and faults [58]. The lifting schema mapping can be applied to XML Schema element declaration, complex type definitions and simple type definitions. All attributes defined by SAWSDL are defined by the XML Schema [47]. Particularly SAWSDL schema mapping intends to address Web Service discovery issues such as how overcome structural mismatches between semantic model and the service inputs and outputs. As a consequence of using SAWSDL implies the need to rely outside software to solve semantic heterogeneities as for lowering semantics is used XML technologies combined with SPARQL to pre-process data [58]. In real applications this tasks is probably assigned to external mediators. Also the transformations from OWL to XML can cause information loss as the XML is less expressive language [43]. As the reference model would be available only in OWL, SAWSDL would not be the best choice because a reference model defined in OWL has to be pre-processed additionally with OWL specific tools [43].

4.2 OWL-S

The Web Ontology Language for Services (OWL-S) defines a set of essential vocabularies to describe the semantics of Web Services, by defining service properties, capabilities, requirements, internal structure and details about the interactions with the service [59]. This markup of Web Services descriptions is intended to facilitate the automation of service discovery, invocation, composition, interoperation and execution monitoring [47]. Web Service discovery involves the automatic location of services being offering a specific service. The invocation is the automatic execution of a service based on markup information that describes required input, functions to invoke and format to follow for the invocation. Interoperation and composition allows for multiple services to act in conjunction to deliver a composite service. The execution monitoring tracks the status of Web Service execution so that users are made aware of the progress of their request. To enable above mentioned tasks, OWL-S defines classes that provide service profile with high level service descriptions, a model information how the service works, whether composition of services needs to be enacted and grounding information on how access the service in programmatic manner [44].

OWL-S has its roots in the DAML Service Ontology (DAML-S), released in 2001 and became a W3C candidate recommendation in 2005 [60]. At the time, DAML-S was the first progressive effort towards semantic annotation of Web Services. By switching from DAML+OIL to OWL, OWL-S adopted existing Semantic Web recommendations yet still maintain bindings to the world of Web Services by linking OWL-S descriptions to existing WSDL descriptions [59]. OWL-S does not attempt to replace or reinvent Web Service standards, rather this approach attempts to provide a semantic layer on non-semantic service description WSDL for Web Service invocation and extends UDDI for service discovery [44].

4.3 SWSF

The establishment of the Semantic Web Services Framework (SWSF) was motivated by the recognition of some shortcomings of describing Web Services could using OWL [61]. A significant problem is that Description Logic (DL) is not well suited to describing

processes. The Semantic Web Services Framework (SWSF) includes two parts, which are Semantic Web Services Language (SWSL) and the Semantic Web Services Ontology (SWSO) [61]. The Semantic Web Services Language (SWSL) is a generic language for specifying formally Web Service concepts and descriptions. SWSL consist of two sublanguages based on first-order logic (FOL) and logic programming (SWSL-Rules) [61]. While SWSL-FOL is mainly used for Web Service ontologies, SWSL-Rules supports service related tasks such as discovery, contracting and policy specification. The other part of SWSF, Semantic Web Services Ontology (SWSO) serves essentially the same purposes as Web Ontology Language for Services (OWL-S) by providing semantic specifications of Web services [61]. Similarly to OWL-S, SWSO possess a comparable service profile, model and grounding. Although the approach of SWSF might be very valuable because of its expressive languages, SWSF seems to be for now only a theoretical contribution to the development of semantic Web Services because no severe implementation efforts are known so far [47].

4.4 WSMO

The Web Service Modelling Ontology (WSMO) evolved from the WSMF as a result of several research projects in the field of Semantic Web Services like Data Information and Process Integration with Semantic Web Services (DIP), Adaptive Services Grid (ASG), Semantics Utilised for Process management within and between Enterprises (Super), Triple Space Communication (TripCom), Network of Excellence project (KnowledgeWeb) and the vision of Semantic Knowledge Technologies (SEKT) in the European Semantic Systems Initiative (ESSI) project cluster [62]. The Web Service Modelling Ontology (WSMO) framework provides Web Service Modeling Language (WSML) for the semantic markup of Web Services together with a reference implementation Web Service Execution Environment (WSMX) [47]. Web Service Modeling Language (WSML) defines syntax and semantics for ontology descriptions, but especially choreography and orchestration of Web Services are not fully specified. WSMO offers four key components to model different aspects of Semantic Web Services in WSML and these are ontologies, goals, services, and mediators [47]. Goals in goal repositories specify objectives that a client might have when searching for a relevant Web

Service. WSMO ontologies provide the formal logic-based grounding of information used by all other modeling components. Mediators are connectors between components that handle interoperability problems. Mediators pass over interoperability problems that appear between components at data as mediation facility of data structures, protocol level as mediation facility of message exchange protocols and process level as mediation of business logics to allow loose coupling between Web Services, goals or requests and ontologies. Each of these components are called top-level elements of the WSMO conceptual model and can be assigned non-functional properties to be taken from the Dublin Core metadata standard by recommendation [61]. The major criticism of WSML is the lack of formal semantics of its service interface and the lack of principled guidelines for developing the proposed types of WSMO mediators for services and goals in concrete terms [43]. The complete connection of WSML with W3C standards such as WSDL and SAWSDL seems to be missing and improvement for this seems to be on going work.

4.5 Comparison of Semantic Web Service description approaches

Web Services description has been the issue of many initiates, projects and languages presented in previous chapters. This section analyze the interaction process including automated discovery, interoperation, composition and invocation of Semantic Web Services, by comparing above presented five significant approaches for adding semantics in Web Service description. Comparison takes into consideration the versions OWL-S 2.1, WSMO D24v0.1, SWSF 1.1 and WSDL 2.0 for WSDL-S and SAWSDL. The comparison is carried out in a conceptual level identifying differences, overlaps and fragilities among interaction process stages of Web Services.

The conceptual models of different approaches should meet at least the following four aspects interaction process. The discovery process consist an act of locating a machine-interpretable description of a Web Service related resource that may have been previously unknown and that meets certain functional criteria. Discovery involves matching a set of functional and other criteria with a set of resource descriptions [63]. The goal is to find an appropriate Web Service related resource. The invocation is an act of a message

exchange between a client and a Web Service according to the service's interface in order to perform a particular task offered by that service [63]. The interoperation defines the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state [63]. This is also called choreography. Composition defines the implementation of the sequence and conditions in which one Web Service invokes other Web Services in order to realize some useful function, for example the pattern of interactions that a Web Service agent must follow in order to achieve its goal [63]. Specifically discovery issues involve the semantic representation and publishing of the services capability and requirements [45]. The comparison studies different approaches responding to discovery without service publishing issues of importing Semantic Web Services in registries like UDDI. The service interoperation is affiliate with the operation and the interaction with the other services. Composition involves issues of combining parts of several Web Services resulting to composite ones. The service invocation describes the mechanisms for actual use and execution of semantically described Web Services.

In general semantic service description covers many different aspects of the service, ranging from the actual capabilities in the domain of value to the ontological grounding of service parameters at the message exchange level. In any of the phases of discovery, invocation, interoperation and composition different aspects have to be taken into account. In the first place, the finding of relevant service offers should be based on details about how to communicate with the service as perceiving at discovery process the specification of the capabilities. The service capability descriptions serve as input for the discovery process, which compares the service requester's description with those of providers to determine which service offer is relevant for the request. This means that discovery operates on the ontological descriptions of the capabilities of a service rather than on specifications of Web Service interfaces. As a result, the discovery process returns references to considered relevant service descriptions together with references to associated Web Service interfaces. The requester agent has then the possibility investigate further the relevant service offers either by looking at their semantic descriptions in more detail or by directly calling the Web Service interface [47]. With respect to Semantic Web Services, parties offer via the Internet the Web Services with

semantic markup providing access to services by means of machine-interpretable metadata that tells agent what the service does [45].

In order that on discovery process can be determined the general service model the capability description from ontological vocabulary has to be interpreted. For this a discovery framework should define the domain independent part of the ontological vocabulary. This part of the ontological vocabulary is usually being fixed by choosing upper-level ontology for services, such as OWL-S or WSMO. The upper-level ontology provides a very general conceptual model for services that is independent from any application domain in opposite to domain ontology. In fact, upper-level service ontology can be seen as an ontology whose domain is the service and therefore it forms the foundation of any capability description by providing basic concepts for service such as Service, ServiceParameter and ServiceCondition [45].

One of the main purposes of Web Services is the automation of application integration within and across organizational boundaries. This implies necessarily the need for interoperation between services. This interoperation can be between services in an organization or crossing different organizational boundaries. To ensure automatic interoperation, description must be defined interpretively using explicit semantics. The interoperation of Semantic Web Services in an application is centralized on two corresponding principles [64]. The first principle is strong de-coupling of the various components that realise the application, including the hiding of information based on the difference of internal business intelligence and public message exchange protocol interface descriptions. Second principle is strong mediation services enabling communication between any parties in a scalable manner, including the mediation of different data formats, terminologies and interaction styles [64].

The Web Services semantically represent the functional capabilities of the services and use semantic discovery techniques for to find and compose these services into a process but a common fallacy of such is the assumption that a semantic match ensures interoperation. To understand this, for example in the case of a process that uses two Web Services with heterogeneous message schemas, the input and output message schemas are

incompatible and the output of the first service is supplied as an input to the second service. The process of resolving these heterogeneities and transforming one message format to another is also referred to as data mediation as described below. A simple solution to achieve data mediation between the services is to manually create a mapping from the first service's output to the second service's input. However, this mapping would have to be created every time services in the process are changed or upgraded, potentially making the number of generated mappings very large [65]. An alternate solution to this problem is mapping the inputs and outputs of the services to a conceptual model and using those mappings for interoperating between the services [65].

The heterogeneity problems are usual in system integration and especially in more open contexts such as those of service oriented architectures together with Web Services. The mediators are used to overcome the heterogeneity problems between different sources of information, software components and knowledge elements [64]. The heterogeneity problem must be handled at different levels. These levels of heterogeneity are usually classified as data mediation, ontology mediation and protocol mediation [64]. The data mediation is concerned with the transformation of the syntactic format of the messages exchanged by different services. To be exact, the service requester may provide an input for the service provider that is not in the format that the latter is expecting and other way round. For example, the parameter values in a SOAP message may appear in different places of the message body or header with different orders. Ontology mediation is concerned with the transformation of the semantic models used by the service requester and provider to express the messages that they exchange [64]. It is probable that the service requester and the service provider use different ontologies with different degrees of complexity to refer to the content of their messages. Protocol mediation is also known as choreography mediation and it is concerned with the problem of non-matching message interaction patterns. Specifically, two or more services exchanging messages may use different interaction patterns for example one of them sends only one message while the other expects two [64].

The composition allows service to be defined in terms of other simpler services. A workflow expressing the composition of atomic services can be defined in the service

ontology by using appropriate control constructs. This description can be grounded on a syntactic description such as Web Services Business Process Execution Language (WS–BPEL) [66]. In order to realize composition Semantic Web Services are considered to be on three levels. At first level the capability states the mandatory inputs, outputs, preconditions and effects of the service as well as its non-functional properties, all taken from a specific given ontology [66]. At second the choreography describes how the service can be consumed from a user point of view with messages exchange patterns [66]. Finally the optional orchestration describes the workflow of the service, including internal computations and calls to external and composed Web Services. Orchestration is generally hidden to the user [66].

The invocation involves a number of steps once the required inputs have been provided by the service requester. First, the service and domain ontologies associated with the service must be instantiated. Second, the inputs must be validated against the ontology types. Finally the service can be invoked or a workflow executed through the grounding provided. Monitoring the status of the decomposition process and notifying the requester in case of exceptions is also important. [67]

The Web Service lifecycle includes publication, operation, discovery, composition, interoperation, invocation, monitoring and recovery [68]. Here discovery, interoperation, composition and invocation within service lifecycle are studied as main service interaction processes. The comparison examines semantic description approaches relevance and support of service interaction. In the next section Semantic Web Service description approaches are compared by a set of concepts which realize the service interactions above with semantic descriptions.

4.5.1 Comparison of conceptual models for interaction processes

The comparison is held in the order of WSDL-S, SAWSDL, OWL-S, SWSF and WSMO analysing their conceptual models according to discovery, interoperation, composition and invocation actions.

Within the Semantic Web Services activities around the METEOR-S project, the METEOR-S Web Services Discovery Infrastructure (MWSDI) is an infrastructure of registries for the semantic publication and discovery of Web Services. The Semantic Annotation of WSDL (WSDL-S), which was developed within METEOR-S project, also supports service discovery [38]. Since WSDL-S does not commit to a specific Web Service ontology or language as OWL-S or WSMO do, one of its characteristics is that semantic information in WSDL-S tags can be expressed in a wide range of standards, languages and formalisms including RDFS, OWL, WSML or even legacy UML descriptions [69]. This is possible because the matching techniques employed in MWSDI mostly work on the labels and some taxonomic and property-related structure of ontological elements, using various concept similarity measures [69]. Mappings from service parameters to ontological concepts are captured in UDDI-specific tModels and UDDI's retrieval facilities are used to perform discovery. However, these techniques do not fully exploit logical inferring with ontologies that have some richer expressiveness. WSDL-S provides set of extension attributes and elements for associating the semantic annotations [38].

WSDL-S hooks for attaching semantics to WSDL components and provides five elements and attributes including modelReference, schemaMapping attributes, precondition, effect and category elements [69]. The modelReference attribute is for one-to-one mapping between a WSDL entity and a concept in a semantic model. The schemaMapping attribute is for mapping between a schema and ontology concepts [69]. Precondition and effect are elements WSDL interface operations to specify conditions that must hold before and after the operation is invoked. Category element provides a pointer to some taxonomy category used on a WSDL interface and is intended to be used for taxonomy-based discovery or when the WSDL interface is filed in a repository [70].

SAWSDL is restricted and homogenized version of WSDL-S. The main difference between SAWSDL and WSDL-S is that the precondition and effect have been discarded. Category has been replaced by the more general modelReference extension attribute which in SAWSDL can be used to annotate XML Schema complex type definitions, simple type definitions, element declarations and attribute declarations as well as WSDL

interfaces, operations and faults [71]. Finally schemaMapping has been decomposed into two different extension attributes liftingShemaMapping and loweringShemaMapping to specifically identify the type of transformation. These mappings can be used during service invocation [71]. SAWSDL does not specify a language for representing the conceptual model, rather it represents incremental approach to introducing semantic characterization of Web Services into mainstream Web Service approaches. SAWSDL aims to provide semantic characterization of a service input and output types, which can be useful in disambiguating those types in the context of simple forms of service discovery [72].

OWL-S Web Service is specified by four descriptions which are Service Profile, Model, Grounding and Service that connects the other three. OWL-S leverages on OWL to support capability based discovery, automatic composition and invocation of Web Services [73]. OWL-S conceptual models Service class instance Service Profile answers the question of what the service does with capability specification, publishing all information needed for service discovery expanding UDDI by means of OWL-S/UDDI mapping [74]. The ServiceProfile class provides a bridge between service requesters and service providers. OWL-S Service profile has two main uses including request of Web Services with a given set of capabilities by providing template for service requests and advertisement of Web Service capabilities such as non-functional properties, quality of service (QoS) and classification in service taxonomies [74]. Profile functionality description deals with issues of information transformation, representing the inputs and outputs of a service and the behaviour and the state change by the execution of it, specified by preconditions and effects. The inputs, outputs, preconditions and results (IOPEs) contained in Profile description are referenced instances of Service Model Ontology IOPEs [74]. Finally Profile is associated with a list of service parameters such as class ServiceParameter that accompanies the service description. In addition, Profile defines serviceClassification and serviceProduct associations to specify the type of service provided and the products that are handled by the service [73].

OWL-S Service Model describes how a service works exposing internal processes of service and facilitates invocation, composition of Web Service and monitoring of

interaction. The service model defines the inputs, outputs, preconditions and effects (IOPEs) and the control flow of composite processes [74]. The service Model views interaction of the service as a process and distinguishes atomic, simple and composite processes. OWL-S Service Grounding is a specification of service access and invocation information building upon WSDL to define message structure. For the definition of service grounding both OWL-S and WSDL are required. OWL-S Profile and Model are considered as an abstract representation of a service, whereas Grounding provides the concrete physical binding layer. Service Model and Grounding together give everything needed for using the service [74].

Semantic Web Services Framework (SWSF) emerged from the work in service composition which might require more expressivity than is available in OWL and is therefore based on logic programming, first-order logic and policy research. The Semantic Web Services Framework (SWSF) includes two major components, the Semantic Web Services Ontology (SWSO) is a conceptual model and the Semantic Web Services Language (SWSL) is a language used to capture full semantics using extensible family of axioms to define the conceptual model [75]. The Semantic Web Services Language (SWSL) appears in two variants which are SWSL-FOL and SWSL-Rules [76]. SWSL-Rules is a logic programming language including features from courteous-logic programs, HiLog and F-Logic and can be seen as both a specification and an implementation language. SWSL-Rules provide support for service-related tasks such as discovery, contacting, and policy specification [75]. Within SWSF discovery is rule based. Service descriptions are expressed in the SWSL-Rules formalism and discovery is realised by executing rule based queries. The querying is performed with transaction logic, a rule-based formalism that supports the explicit representation of change basing the idea of early work on discovery in the context of WSMO. However, there is not much work on discovery with SWSL-Rules published recently [75]. The SWSF service ontology SWSO-FOL, also known as (FLOWS) is expressed in first-order logic. The First-order Logic Ontology for Web Services (FLOWS) is an extension of the Process Specification Language (PSL) that was originally developed to specify and share specifications of manufacturing processes standardized by ISO 18269 [76]. In the formal PSL ontology the notion of activity is a basic construct corresponding to a manufacturing

or processing activity [76]. Each of the constructs used to derive functionality of the semantics constructs by the use of Process Specification Language (PSL) and so overcomes some initial problems with the workflow semantics of the OWL-S process. SWSF Process Model includes not only inputs and outputs, but also messages and channels and therefore claims to provide an enhanced process model as compared to OWL-S [59]. Core part of this PSL extended by FLOWS is called PSL Outercore, and the resulting FLOWS sub-ontology is called FLOWS-Core [75]. Models of PSL Outercore correspond to trees, whose nodes correspond to occurrences of atomic processes, with an edge connecting nodes [77]. With this as a starting point, properties of Web Services and their compositions are intuitively described specifying constraints on those execution trees [77].

For interoperability FLOWS had the additional objective of acting as a focal point, enabling other business process modeling languages to be expressed or related to FLOWS. As FLOWS is a modular and extensible ontology, it is hence possible to provide alternative extensions to represent different approaches to message handling, choreography, and orchestration. FLOWS can serve as Interlingua ontology that can facilitate interoperability of Web services that use different ontologies [77]. As such, FLOWS use of first-order logic provides a unifying foundation for sufficient expressivity, well-defined semantics and a diversity of automated reasoning tools. However, being based in first-order logic, most decision problems concerning FLOWS specifications are undecided and the FLOWS work is still quite preliminary needing continuous and augmentative study of Web Services models that capture different subsets of the overall FLOWS model [77].

The Web Service Modeling Ontology has its conceptual origin in the Web Service Modeling Framework (WSMF) [78]. WSMO conceptual model has four top-level elements, which are Ontologies, Web Services, Goals and Mediators. WSMO makes use of the Meta-Object Facility (MOF) gaining a benefit that the model and the languages can be ultimately used to describe Semantic Web Services being separated from one another [78]. Ontologies in WSMO introduce terminology used in other elements and provide the way by which enhanced information processing becomes possible and complex

interoperability problems can be solved. A WSMO Web Service contains the definition of services in terms of a capability describing the functionality and in terms of an interface providing method to interact with the service. A WSMO goal can be seen as a description of services that would potentially satisfy the requester's desires because are representations of objectives that would be fulfilled through the execution of the service and additionally they can be descriptions of services that would potentially satisfy the user desires [78]. Mediators are a core element of WSMO and aim to resolve heterogeneity problems at the data, process, and protocol levels and are used to solve interoperability problems between goals, ontologies or services [78]. Mediators describe elements that resolve interoperability problems between different elements. For example between two ontologies the mediator has the job of aligning, merging or transforming the imported ontology to resolve any existing heterogeneity issues that may arise by importation. It is possible to add non-functional properties onto all WSMO elements. The elements defined by the Dublin Core Metadata Initiative are taken as non-functional properties. Non-functional properties are mainly used to describe non-functional aspects of a description, such as the creator and creation date and to provide natural-language descriptions. Dublin Core is a set of attributes that define a standard for cross domain information resource description [78].

In WSMO conceptual model discovery operates on WSMO Capability elements as abstract semantic service descriptions while concrete input and output parameters and other communication details are treated in a separate interface element and therefore the grounding of service descriptions in WSDL specifications does not affect discovery. For discovery WSMO ontologies introduce a common terminology for the concepts representing service characteristics. They provide concepts, relations among these concepts and a set of logical expressions called axioms which explore the semantics of concepts, relationships or function definitions in the ontology [78]. In order to define the provided functionality of a service, the notion Capability of a service is associated to a number of preconditions and assumptions specifying the world state before the execution of the service and post-conditions and effects describing the state after the execution of the service. Goals are also related to service discovery procedure as they describe aspects

related to the requirements of the end user by addressing problems that are designate by those services [78].

For interoperability WSMO introduces the Service class element for representing the service operation and data transformation. A Service class is associated to a capability element describing its functionality and is described by a number of interfaces denoting how the capability of the service can be fulfilled. As solution for interoperability problems arising in data transformations WSMO uses different mediators [78]. ooMediator links two ontologies resolving possible mismatch issues between them. ggMediator connect goals to one another and allow for relationships between different goals to be specified. wwMediator links two services and wgMediator links services with goals, meaning that a service realises the goal that is linked to [78]. For composition purpose, interface class is associated with the choreography element for providing the necessary information for communicating with the service and the orchestration class element that describes how the service makes use of other services in order to achieve its capability. The differences between these two descriptions are the communication and cooperation purposes. The choreography provides a description of how to interact with a service for communication purposes, while the orchestration describes how the overall function of the service is realized through cooperation with other services in terms of the functionality required from other services in order to realize this service [79]. The interface of a service is presented in a machine-interpretable manner, based on the Abstract State Machines (ASMs) methodology allowing for software to determine the behavior of the service and to reason about it [79]. Considering the invocation in WSMO service grounding the relation of WSMO with WSDL is concerned with two different directions. In the first one WSDL elements extend the WSMO descriptions in order to specify the implementation details and on the other hand WSMO elements are used to extend and semantically annotate the implementation level WSDL [78].

4.5.2 Comparison of service interaction support

Empowerment of automatic services interactions is crucial bridging the gap between commercial and semantic Web. The key challenges of service description approaches are

to bring behavioural intelligence to the Web Services in terms of verification of correctness of operation of services and to provide powerful methods and tools for internal representation of services. The ontologies and description languages are natural way to express aspects of service descriptions and their role is to classify service for interaction purposes.

To consider Semantic Web Services interaction issues, WSDL-S and SAWSDL does not aim to provide a comprehensive framework to support more sophisticated approaches to discovery, composition, or any of the other service related tasks that Semantic Web aims to automate. Taking into consideration the discovery issues, in OWL-S service Profile presents the intended purpose of the service describing both the functionality offered and the one needed by the requestor, whereas WSMO distinguishes the provider standpoint from the requestor by, introducing goal and capability which are two different concepts [80]. As far as expressiveness issues are concerned functionality descriptions, OWL-S framework does not state any restrictions as a service can be presented by none or more than one profiles. Web Services in WSMO can be associated to none or more goals by using the right mediators and to only one capability. In WSMO, due to separation that abstract semantic service descriptions are defined in capability elements and other communication details are treated in a separate interface element, WSMO discovery is currently farther away from integration with concrete Web Service technologies such as WSDL and UDDI and from implementation in tools than METEOR-S Web Service Discovery Infrastructure (MWSDI) or OWL-S matchmaking [81]. Similar of OWL-S service Profile and WSMO goal and capability is the abstract part of the WSDL 2.0 extension called WSDL-S and its homogenized version SAWSDL. In addition to discovery, SAWSDL specification mentions that SAWSDL annotations can be used during composition and invocation. However it is important to understand that SAWSDL is of very little use unless there is an additional specification of conventions and guidelines for what can be referred to in some particular semantic framework and what it means by doing so [74]. In SWSF, SWSL-Rules provide support for discovery, contacting, and policy specification with help of courteous-logic program features. Because of query based discovery by Transaction Logic provides logical justification for

choosing right service from alternative offers, but due the ontological role separation, SWSF does not have direct impression of user desires [82].

In SAWSDL Web Services interoperate by reusing lifting and lowering mapping. SAWSDL by itself does not guarantee interoperability rather it is the standard basis for monitoring of Semantic Web Service descriptions. Noting that WSDL-S is associated with set of tools of METEOR-S, in the approach the operation of a Web Service is described by the concrete part of the WSDL-S document and the Abstract WS-BPEL process model [55]. To view interoperability, OWL-S Service Model represents how a service works. Each Web Service is viewed as a process in Service Model and similarly in WSMO the Service class describes the provided service functionality. WSMO Capability can be considered similar to the process concept of OWL-S and more precisely of the atomic process, while simple process is not considered in WSMO. WSMO relies on interoperability with strong mediation. Different kinds of mediators are used to link together the core WSMO elements, dealing with the heterogeneity problems inherent to a distributed environment. OWL-S does not unambiguously consider the heterogeneity problem in the language itself, rather treating it as an architectural issue that mediators are not an element of the ontology but are part of the underlying Web Service infrastructure [81]. In contrast to OWL-S, which is based on description logic (OWL DL) that is not expressive enough to allow the full axiomatization of the process model, SWSF is based on much more expressive first-order language that facilitates the definition of the concrete service ontology (SWSO). SWSO hold rich behavioural process model based on ontology for manufacturing processes (PSL) extends it with concepts required by Web Services, such as messages, channels, inputs, and outputs. The two sublanguages of SWSL, SWSL-FOL (FLOWS) and SWSL-Rules are similar, but they have different interpretation of same expressions. Therefore they are semantically incompatible and should not be used in a same document [83].

Dealing with composition problems, in WSMO interface describes both the choreography and the orchestration of a composite service using a model based on Abstract State Machines technology. In OWL-S the process model defines a composite process and in the same way WSMO introduces the interface class. Here the difference between OWL-S

and WSMO is concerned with the restriction of process model as in WSMO using multiple interfaces can be specified different models for service choreography and orchestration. However, the explicit definition of Web Service choreographies in WSMO is still unspecified, while OWL-S provides a detailed description of the service model, allowing the definition of atomic, simple and complex processes, together with their control and data flow. In addition, the current version of WSMO does not sufficiently define how the orchestration is described, which makes the linking two services with wwMediator and the definition of the decomposition of the service very challenging if not impossible [81]. For WSDL-S and SAWSDL-S, in METEOR-S approach the composition mechanism is carried based on the standards introduced by WS-BPEL. SWSF and also WS-BPEL are process-oriented using main idea of message driven architecture that service is described by procedural messages [83].

Considering the invocation issues, the majority of the provided services are syntactically described, so the clients are also implemented to deal with syntactic information concerning service description and invocation. Apart from the WSDL-S and SAWSDL which introduce an extension of WSDL for semantically annotating its parts, OWL-S, SWSF and WSMO establish grounding mechanisms based on WSDL. OWL-S/WSDL, SWSO/WSDL and WSMO/WSDL grounding models annotate the WSDL parts with references to the corresponding ontology concepts and they relate parts of the service ontology with right WSDL description parts such as operations and message parts [47].

The Table 3 presents the comparison of correspondences and differences among five service description approaches according methods explained above how these approaches support service interaction. OWL-S, WSMO and SWSF shares the vision that ontologies are essential to support automatic discovery, interoperation, and composition of Web Services. SWSF builds loosely on OWL-S to provide a more comprehensive framework, in the sense of defining a larger set of concepts. SWSF uses SWSL to define SWSO an ontology of service concepts that takes advantage of SWSL's greater expressiveness relative to OWL to more completely axiomatize the concepts. Although SWSF is interesting and very valuable, it is unclear how all paradigms such as FLOWS, ROWS and SWSL-Rules of this approach work together [76]. In addition, no serious

implementation efforts are known so far and thus SWSF seems to be only theoretical contribution to the development of Semantic Web Services. As SWSF with SWSL and SWSO, the WSMO effort defines expressive Web oriented language, the Web Service modelling language (WSML) provides a uniform syntax ranging from description logics to first-order logic [78]. Like OWL-S, WSMO declares inputs, outputs, preconditions, and results associated with services using quite different terminology.

Table 3. The conceptual models comparison according service interaction support.

Organization/ Interest Group	**Member submission**	**Language/ Formalism**	**Service interaction support**			
			Discovery	**Interoperation**	**Composition**	**Invocation**
W3C/ SAWSDL Working Group METEOR-S	WSDL-S	Extension to WSDL 2.0	WSDL UDDI Category	Abstract WS-BPEL process Input, output, precondition, effect	WS-BPEL process	WSDL
W3C/ SAWSDL Working Group METEOR-S	SAWSDL	Extensions for WSDL XML, XML Schema	WSDL UDDI modelReference	lifting/lowering mapping for mediation inputs, outputs, operation, interfaces, faults	WS-BPEL process	lifting/ lowering mapping for invocation
W3C Workshop on Frameworks for Semantics in Web Services	OWL-S	OWL	Profile	Process Model Input, output, precondition, effect	Process Model Composite process, control construct	Grounding OWL-S/ WSDL
W3C/ Web Services Activity	SWSF	SWSL SWSO	SWSL-Rules	Process Model FLOWS-Core	Process Model FLOWS-Core, control constrains	Grounding SWSO/ WSDL
W3C Workshop on Frameworks for Semantics in Web Services	WSMO	WSML	Goal, Capability	Service Capability, precondition, postcondition, assumption, effect	Interface Choreography, Orchestration	WSMO/ WSDL

Unlike OWL-S, WSMO does not provide a notation for building up composite processes in terms of control flow and data flow, rather it focuses on specifying internal and external choreographies, using an approach based on abstract state machines. Other difference is that WSMO emphasis on the production of a reference implementation of an execution environment, the Web Service execution environment (WSMX) and on the

specification of mediators for example mapping programs that solve interoperability problems between Web Services [70]. In WSDL-S the way it allows to specify a correspondence between WSDL elements and semantic concepts is very similar to how the OWL-S grounding works and in fact, a notation similar to OWL-S declarations could be used to specify the referents of the WSDL-S attributes. The major difference between WSDL-S and the OWL-S grounding is that with WSDL-S, the correspondence must be given in the WSDL specification, whereas with OWL-S it is given in a separate OWL document. Like WSDL-S, SAWSDL is not actual framework for complete modelling of Semantic Web Services and need concrete service ontology that can be attached to SAWSDL. In fact, the major Semantic Web Services approaches OWL-S and WSMO have started to adopt SAWSDL and use it for grounding purposes.

4.5.3 Comparison of approach selection for service description creation

All these above compared approaches with their languages differ in terms of expressiveness, complexity and tool support. The dynamic service interactions are difficult tasks in traditional service infrastructures and human intervention is often required. The Semantic Web Services paradigm tries to keep the intervention of end users to the minimum, automating as much as possible the discovery, composition, invocation and interoperability of Web Services. The usage of Web Services in the mobile domain is still not as advanced, widespread and established as in desktop computing. Despite the technical progress in mobile computing, most of its devices, principally PDAs and mobile phones only provide inadequate means for the interaction with Web Services and the presentation of their contents, which constrains the mobile usage of Web Services and the usability of applications based on them.

In the ubiquitous computing paradigm, specifically in mobile computing the physical environments are described as full of computational resources facilitating user interaction and access to the resources and information, that is services everywhere and all the time [84]. In these environments the interaction between the user and the system should be intuitive, pleasing and natural. One of the main features of this kind of environments is the mobility resources, that makes these environments highly open and distributed, which

need ad-hoc deployment and execution, specifically situation customized deployments, integrating available software and hardware resources at a suitable place and time [84]. This is only possible if devices are noticed as autonomous and independent resources in the system. As a result, the SOA paradigm is very suitable for this kind of environment since the applications and devices that support them are abstracted like loosely coupled services that can be integrated in bigger systems. SOA and Web Services enable software to be exposed in the Internet as self descriptive services which can be registered, published, discovered, invoked, composed and remotely accessed at runtime. Various service providers, service requestors and service directories collaborate in SOA. Service providers publish service advertisements in the service directories as service directories handle service discovery requests by identifying suitable services matching the requests. For example, with OWL-S the Semantic Web Services are described in an unambiguous manner allowing for a potential service requestor to place a capability search in a service registry. Web Services description based on ontology can depict the functional features, the performance features and the semantic features of Web Services. By incorporating the link between actors and concepts into the model of ontologies have only benefits to bring in terms of more meaningful and easily maintainable conceptual structures. However, the creation of service descriptions is left on the hand of professionals because of the complexity of ontologies and description languages.

In order to offer user provided services, firstly have to provide to users opportunity and way to semantically markup their services that services are discoverable and invocable over networks. By giving the tools for end users to create service descriptions of their services, it provides a more intuitive and natural way for services to interact, which could leverage their usage, dissemination and availability.

According the tools provided by above compared Web Service description approaches, unlike SWSF without any serious implementation efforts, the OWL-S offers large scale of tools developed in and outside of Semantic Web Service community. The tools for OWL-S service creation are described in chapter 6. For WSMO tool support the WSMO Studio is a quite complex set of Eclipse plugins. In addition, OWL-S has been used in very different configurations in Web Service architectures that adopt the SOA set of a

service registry, producer and consumer interactions, in peer-to-peer (P2P) systems using a Gnutella-based architecture, in multicast-based universal plug and play (UPnP) systems used with wireless devices, in architectures that exploit a centralized mediator to manage the interaction with services and perform different forms of translation between service consumers and providers [85]. As OWL-S and WSMO are quite complex with their conceptual structure, the SAWSDL is very lightweight approach, but does not provide full expressiveness of service functionalities. Both OWL-S and WSMO provide also non-functional properties representation so that the service description is readable both by human and machine. Although WSMO seems suitable for smart ad-hoc environments with respect of ability define predicates and variable types that occur in preconditions, assumptions, effects and postconditions in ontologies, it lacks of supported tools. A major criticism of WSMO is that it is drifting from W3C standards [85].

Even tough OWL-S may need several improvements, it is built on existing standards and provides well defined conceptual structure and several tools for the creation of semantic service description with OWL-S. The one of main objectives of OWL-S, as in this chapter reviewed, is to be comprehensive enough to support the entire lifecycle of service tasks [85]. OWL-S is selected for semantic service description creation done by end users trough mobile device because of its ease of compliance to architecture of semantic assistant.

5. WEB ONTOLOGY LANGUAGE FOR SERVICES

The Web Services can be described by three general aspects such as “What does the service do”, “How does service work” and “How service is invoked” [86]. The doing expose the service functionality, the working expose procedures of acts and the invoking expose the set of needed message formats and supported protocols necessary for functionality. In order to create an ontology that can provide terms defining above mentioned three aspects of Web Services, ontology is needed to have general conceptualizations and usable across all domains. An ontology describing general classes and properties and loose of any particular domain is called upper ontology [47].

OWL-S is an upper ontology used to describe the semantics of services based on the W3C standard ontology OWL. Being upper ontology, OWL-S does not make any assumption with respect to the processing environment or Web Service architecture and has been used in very different configurations [85].

5.1 Service descriptions using OWL-S

The primary objectives of OWL-S are to build on existing Web Service standards and existing Semantic Web standards representational framework in which to describe Web Services, to support automation of service management and use by software agents and to be inclusive enough to support the entire lifecycle of service tasks [85]. OWL-S structure consists of three ontological parts for describing different aspects of Semantic Web Services. The first is Service Profile describing the functionality a Web Service offers, including the constraints and non-functional properties that influence functionality. The aim of Service Model is to describe Web Services functionality through a behavioural model. OWL-S seeks to build on top of WSDL and SOAP by mapping elements in the Service Model to elements in the WSDL description. This part of the OWL-S ontology is called the Service Grounding [43].

The semantic service description should at least reveal input, output of the service and two sets of conditions. The preconditions have to be met before executing the service

properly and the effects that are conditions that hold after the successful execution of the service [87]. These four service description elements Input, Output, Precondition and Effect are often referred to as IOPE. OWL-S defines the capability a service offers in terms of a state transition as having its inception in the research area of artificial intelligence. In OWL-S it is possible to specify the inputs and outputs expected to be sent to and received from a service along with preconditions that must hold before the service can execute and the effects of the service executing [85]. The intent is that along with arbitrary non-functional properties, it should be sufficient information for a discovery agent to be able to decide if a desired service matches any of the service description in the set of candidate OWL-S Web Service descriptions available to it.

OWL-S provides a framework for describing both the functions and advertisements for Web Services including a process specification language, a notation for defining process results, and a vocabulary for connecting these descriptions in OWL with syntactic specifications [47]. As thinking advertisement scenario, for example to find a vendor of "furniture" it would be necessary to include companies that advertise the sale of "homeware". A software agent using OWL and OWL-S could apply ontological knowledge to wider its search to include requests for providers of homewares as this would be annotated as a superClass of furniture. If a vendor used a different vocabulary where "decoration" took the place of "furniture", cross-ontological statements involving the equivalentClass construct of OWL would enable the agent to translate between the two.

5.2 OWL-S Structure

The OWL-S conceptual model is constructed with the service profile, process model, and grounding sub-ontology [85]. The service profile is used to describe what the service does, the process model is used to describe how the service is used and the grounding is used to describe how to interact with the service. Figure 7 shows OWL-S conceptual model and the relationships between the top-level classes of the ontology. In Figure 7, an oval represents an OWL class, and arrow represents an OWL property. The class Service provides an organizational point of reference for a declared Web Service. Only one

instance of Service will exist for each distinct published service. The properties presents, describedBy, and supports are properties of the OWL class Service. The classes ServiceProfile, ServiceModel, and ServiceGrounding are the respective ranges of properties [85].

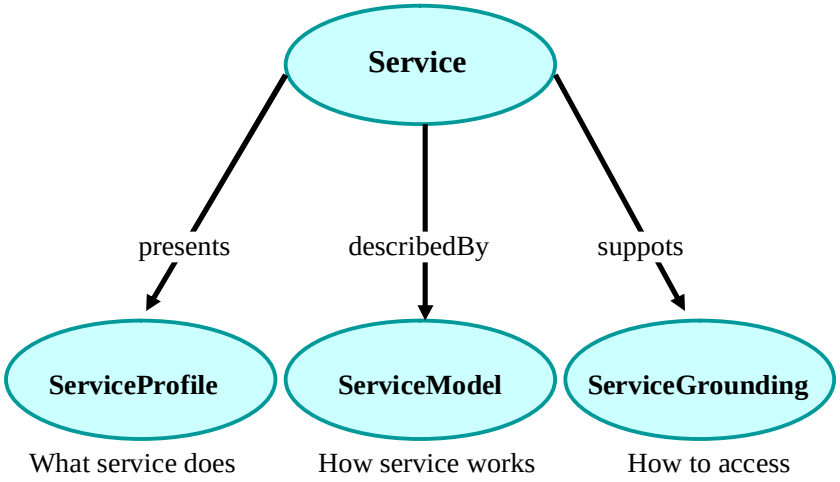

Figure 7. The conceptual model of OWL-S [85]

The service profile tells what the service does giving the type of information needed by a service seeking agent to determine whether the service meets its needs. The service model tells how the service works describing what happens when the service is invoked. For non-trivial services such as composed of several steps over time, the model may be used by an agent at least in four different ways. An intelligent agent can use service model to perform a more thorough analysis of whether the service meets its needs, to compose service descriptions from multiple services to perform a specific task, to coordinate the activities of the different participants during the route of the service performance and to monitor the execution of the service. Grounding specifies the details of how an agent can access a service [88]. Typically grounding may specify some well know communications protocol such as RPC, HTTP-FORM, CORBA IDL, SOAP, Java remote calls, OAA, Jini and service-specific details such as port numbers used in contacting the service [87].

The class Service organizes the parts of service description. Instances of Service can consider as an application programming interface (API) notification for service entry.

Each instance of OWL class Service can have optionally one or more profiles and models. In addition, if OWL-S ontology includes a model, it has to be associated with one or more groundings. Figure 8 shows an example of the Service class in OWL-S 1.2 how the fictitious book buying service is modelled. Each instance of Service presents one or more instances of ServiceProfile and may be describedBy an instance of ServiceModel [88]. When a ServiceModel exists, the Service supports one or more instances of ServiceGrounding [88].

```
<service:Service rdf:ID="ExpressCongoBuyService">
<service:presents rdf:resource="http://www.daml.org/services/owl-s/1.2/CongoProfile.owl#Profile_Congo_BookBuying_Service" />
<service:describedBy rdf:resource="http://www.daml.org/services/owl-s/1.2/CongoProcess.owl#ExpressCongoBuy" />
<service:supports rdf:resource="http://www.daml.org/services/owl-s/1.2/CongoGrounding.owl#ExpressCongoBuyServiceGrounding" />
</service:Service>
```

Figure 8. The Service class in OWL-S 1.2 [89]

5.2.1 ServiceProfile

The OWL-S Profile describes a service as a function telling, who provides the service, what functions the service computes and an assortment of features that specify characteristics of the service [87]. With the goal of supporting advertising and discovery based on capability, the OWL-S profile allows service providers to advertise what their services do, and service requesters to specify what capabilities they expect from the services they need to use [85]. The provider information includes contact information that refers to the entity that provides the service, for example communicating ways of a provider in terms of contactInformation property, which can be restricted to some other ontology, such as FOAF or VCard [87]. Providing explicit description of capabilities, they do not have to be extracted from incidental properties. In discovery process can be found services that are most likely to satisfy the needs of a requester by exploiting the structure of OWL-S profiles and their references to OWL concepts.

In OWL-S profile can classify three description types, which are functional, classification and non-functional description as shown in Table 4. The functional description of the service is expressed in terms of the information transformation produced by the service,

from the inputs that the service expects to the outputs it generates and the transformation in the domain, from a set of preconditions that need to be satisfied for the service to execute correctly, to the effects that are produced during the execution of the service [85]. The functional description constitutes Input, Output, Precondition and Result connected to ontology Process IOPE properties, which is an important part of OWL-S. The OWL-S classification description indicated by the profile type and the service categories supports the description of the type of service as specified in taxonomy of businesses or other suitable domain specific ontology. The non-functional description allows making distinctions between services that do the same thing but in different ways or with different performance characteristics [85].

The Table 4 shows OWL-S Service Profile three description types; functional, classification and non-functional description with Profile properties and explanation of their usage. The first properties such as presents and presentedBy simply express that the profile describes the service. The properties ServiceName, contactInformation and textDescription form the basic information of the service. A semantic name can be given to the service using serviceName and any information can be represented in textDescription with free text descriptions. The next five properties which are Process, Parameter, Precondition, Result, Input and Output are used for the functional description and refer to the process description in ServiceModel [41]. For referring the process that models the service is used has_process to provide the URI as the unique ID identifying the selected operation on the server side [85]. The classification description is based on ServiceParameter, serviceClassification, ServiceCategory and serviceProduct [85] that may be used to refer to existing taxonomies that may not be described in OWL using mapping to OWL ontology of services or products, such as the North American Industry Classification System (NAICS) or United Nations Standard Products and Services Code (UNSPSC) [41]. ServiceParameter is used to specify additional features of the service consisting of the actual name of the parameter defined as literal or URI in serviceParameterName and sParameter linking to the value within OWL ontology [41].

Table 4. The properties in OWL-S Service Profile.

Service Profile	Properties/ subclasses	Explanation
Profile	presents presentedBy	The service is described by the profile
Functionality	serviceName contactInformation textDescription hasParameter	Name of the service that is being offered can be used as an identifier of the service. In the text description is summarized what the service offers and what the service requires to work indicating any additional information. The contact information refers to humans or entity responsible for the service
	has_Process hasPrecondition hasInput hasOutput hasResult	Range over instances of inputs, outputs, precondition and result as defined in the Process Ontology. The result specifies under what conditions the outputs are generated and what domain changes are produced during the execution of the service
Classification	serviceParameter sParameter serviceCategory categoryName taxonomy value code	The Service Parameter is an expandable list of properties and value of the property is an instance of the class ServiceParameter. The Category refers to some ontology or taxonomy of services. The name of the actual parameter and category is given with Parameter and Category Name which can be a literals or URI of the process property. sParameter points to the value of the parameter within some OWL ontology. The taxonomy can be URI of the taxonomy or URL where the taxonomy resides or the name of it. The value refers to the value in a specific taxonomy and there may be more than one value for each taxonomy. Same way code refers to each type of service stores the code associated to a taxonomy.
	serviceClassification serviceProduct	Classification defines a mapping from a Profile to OWL ontology of services and Product to OWL ontology of products.
Non-functional		Non-functional properties include serviceName, textDescription, ServiceParameter, ServiceCategory, serviceClassification and serviceProduct

In Table 4 non-functional description is located down on left side because its description includes different non-functional properties. The non-functional description includes serviceName and textDescription from basic information and ServiceParameter, ServiceCategory, serviceClassification and serviceProductfrom classification part [41]. For example within ServiceParameter can specify additional features such as quality rating [88]. These non-functional properties distinguish characteristics between similar services that do the same thing but in different ways or with different performance [85]. As a notice, in the profile is not required to list all inputs, outputs, preconditions and

results, because it is possible to leave out some if they are unimportant for the advertisement and discovery. In addition, in OWL-S Profile the subclasses may be created for certain domains or types of service and specialized with appropriate properties. For example with shipping services there might be a ShippingServiceProfile subclass with the additional property geographicRegionServed, with GeographicRegion and its range from appropriate online ontology [85].

Because of impossibility to provide a complete set of attributes for the representation of service parameters of which many are domain-specific, the OWL-S provides solution to allow extensible mechanism to define the service parameters they need according to actual conditions for the providers and consumers [85].

5.2.2 ServiceModel

As with help of OWL-S profile software agent has identified a Web Service likely to be relevant to its goals, a agent needs detailed model of the service to determine whether the service can meet the needs, and moreover, what constraints must be satisfied and what pattern of interactions are required to make use of the service. Considering the tasks of software agent, the ServiceModel enables an agent to perform an analysis in detail of whether the service can accomplish its needs, compose service from multiple services to perform a specific task and monitor the execution of the service [88]. To fully understand OWL-S ServiceModel described as process has to give declaration of inputs, outputs, preconditions and results. The inputs are the object descriptions that the service works on and the outputs are the object descriptions that it produces [88]. These descriptions not known when the process is defined and all what can be expected is the specification of their types as OWL classes. A precondition is a requirement that must be true in order for the service to operate and as effect is a requirement that will become true when the service completes, the results consist of effect and output specifications [85]. More precisely, the result field specify the values produced and effects that occur when the process is performed by a web-service client [87]. Both inputs and outputs can be conditional executed and this information concerns the conditions that need to be met for the service to be invoked and the impact the execution of the service. OWL-S support the

specification of conditions of inputs and outputs using logical formulas to express preconditions such as Knowledge Interchange Format (KIF) and Planning Domain Definition Language (PDDL) or either XML-based such as SWRL and RDF based SPARQL [85]. Condition specifications are critical to the definition and execution of the process [74].

The OWL-S definition of Process has a set of associated features which are above mentioned inputs, outputs, preconditions, results linked to the Process concept by OWL properties such hasInput, hasOutput, hasPrecondition and hasResult. In OWL-S is specified with the process the possible patterns of interaction with service. A process in OWL-S can be atomic, simple or composite. The atomic and composite processes are capable for invocation processes while simple process can be used to provide abstracted views of atomic or composite processes. An atomic process has no internal structure and is a single interchange of inputs and outputs process description with exposed IOPEs [87]. A composite process is hierarchically defined workflows, consisting of atomic, simple and other composite processes. The composite process workflow consists of a set of component processes linked together by control flow and data flow structures. The control flow is described using typical programming language or workflow constructs such as sequences, conditional branches, parallel branches and loops. Data flow is the description of how information is acquired and used in subsequent steps in this process. The process workflows are constructed using different control flow operators [85]. The Figure 9 presents partly the Profile and overall structure of ServiceModel, in which Service class is presented in ServiceProfile and described by the ServiceModel.

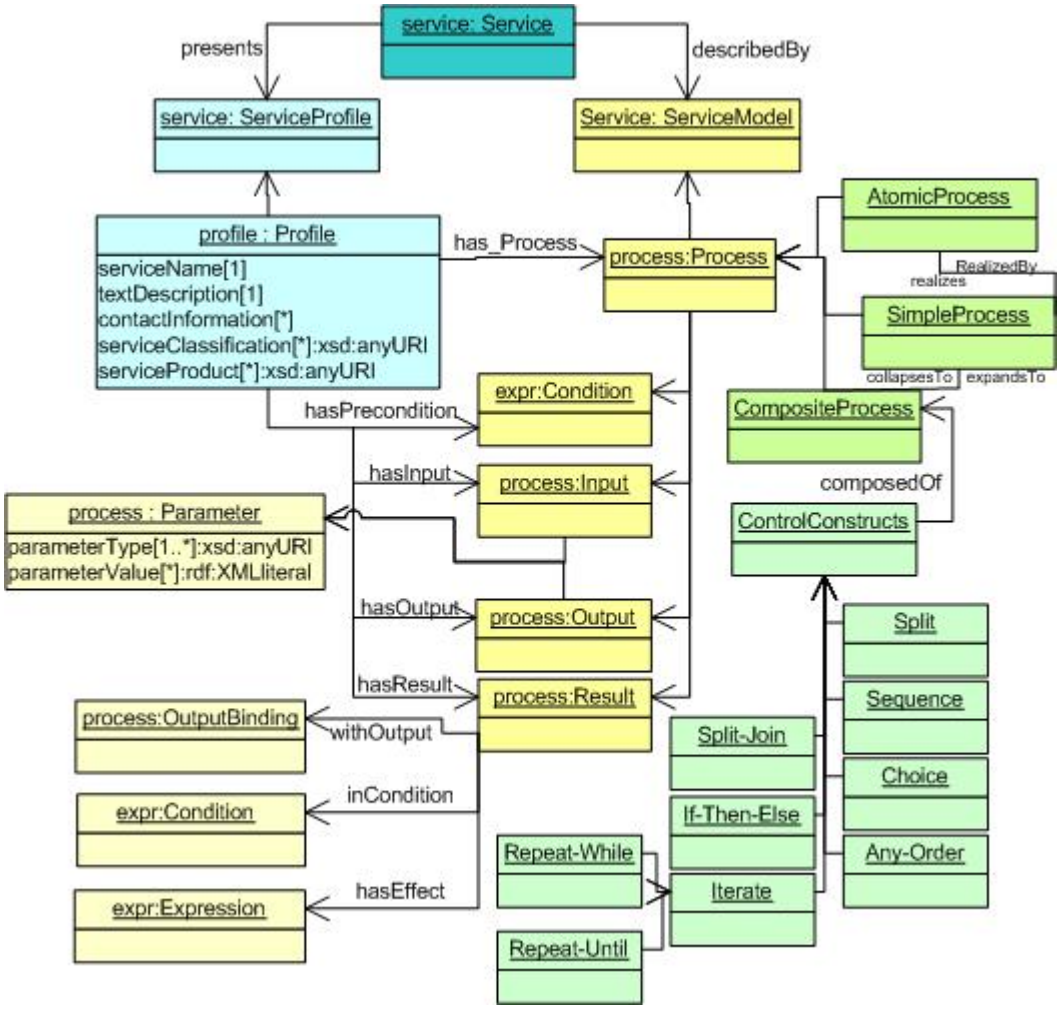

Figure 9. The structure of OWL-S ServiceModel.

As mentioned above, the ServiceModel is defined as a process. The Input, Output and Precondition are explained above, but as showed in Figure 9, the result specifies set of alternative results individualizing inCondition, withOutput and hasEffect. While the conditions, in which result is produced is defined with inCondition, the output parameters are specified with withOutput and hasEffect specifies set of effects in result [85]. The atomic process correspond to a single interaction with the service, for example a single operation in WSDL document. The composite processes have multiple phases each of them being an atomic process connected by control and data flow [85]. Simple processes can be used for planning or reasoning by allowing multiple views on same process. An atomic process realizes a simple process and simple process is realized by an atomic process. Simple process can be expanded to a composite process and composite process

collapses to simple process [47]. The control flow operators include Sequence, Any-Ordered list, Choice, If-Then-Else, Iterate, Repeat-until, Repeat-while, Split and Split+Join In composite processes the decomposition can be specified by Sequence and If-Then-Else control constructs [85]. Link between the ServiceModel and the description of the concrete realization for a Web Service provided by WSDL is given with the ServiceGrounding which is explained in the next section.

5.2.3 ServiceGrounding

The OWL-S ServiceGrounding is used to specify how the abstract information is realized by concrete information exchanges between the consumer requesting a service and the service provider [85]. The abstract information exchanges are detailed by atomic process descriptions and in grounding atomic processes are mapped to WSDL operations, where the process inputs and outputs, described using OWL, are mapped to the operation inputs and outputs, described using XML Schema. In case of Composite processes, which are composed of atomic processes, are grounded in the same way with the additional requirement of OWL-S process engine to interpret the defined control and data flow [47]. To allow for dynamic binding to a service provider, grounding can be supplied at runtime, setting free the developer or user of service consumer code from having to select a specific service provider when that code is deployed.

The OWL-S release 1.2 uses WSDL 1.1 and there is not yet update for use with WSDL 2.0 [85]. The Figure 10 shows the relation with WSDL and OWL-S, where arrows represent explicit mappings declared in OWL-S grounding. Whereas OWL-S allows for the definition of abstract types as OWL classes based on description logic, WSDL specifies message types using XML Schema by default and consequently OWL-S/WSDL grounding uses OWL classes as the abstract types of message parts declared in WSDL and then relies on WSDL binding constructs to specify the formatting of the messages [85].

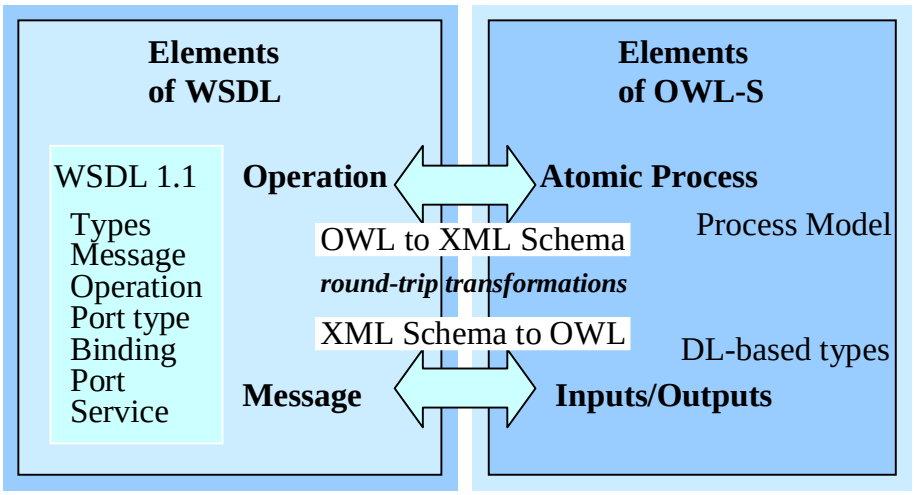

Figure 10. The relationship between WSDL and OWL-S in grounding

The Figure 10 shows that WSDL operations correspond to OWL-S atomic processes and the set of inputs and outputs of an OWL-S atomic process correspond to messages in WSDL, where single input of atomic process becomes part of a message to WSDL operation. Operations and messages give utility by their relationships with bindings for example SOAP and HTTP messages and other elements of WSDL.

The Figure 11 presents the ServiceGrounding model including properties referenced by the grounding that relate the process model to corresponding messages of WSDL. The OWL-S process model of a service is mapped to a WSDL description through grounding [87]. The grounding provides a bridge between the syntax- and protocol-oriented WSDL and the OWL that relies on semantic elements defined using description logics. As shown in Figure 11, the most abstract description of a Web Service is in terms of messages, operations and port types. The class AtomicProcessGrounding defines the atomic process to which grounding is applied [85]. The class WsdlAtomicProcessGrounding defines the grounding of a particular atomic process and expresses related, but complementary information to the WSDL declarations related to operations and messages. The wsdlOperation property of a WsdlAtomicProcessGrounding specifies the portType and operation pair from the WSDL specification. The wsdlInputMessage and wsdlOutputMessage properties specify how the atomic process maps into a request-response pattern of WSDL operation. The wsdlInput and wsdlOutput properties specify

message maps such as associations between OWL-S parameters and WSDL message parts [88]. This information allows an OWL-S service consumer to find in the WSDL specification the information about the concrete embodiment of the abstract parameters [88].

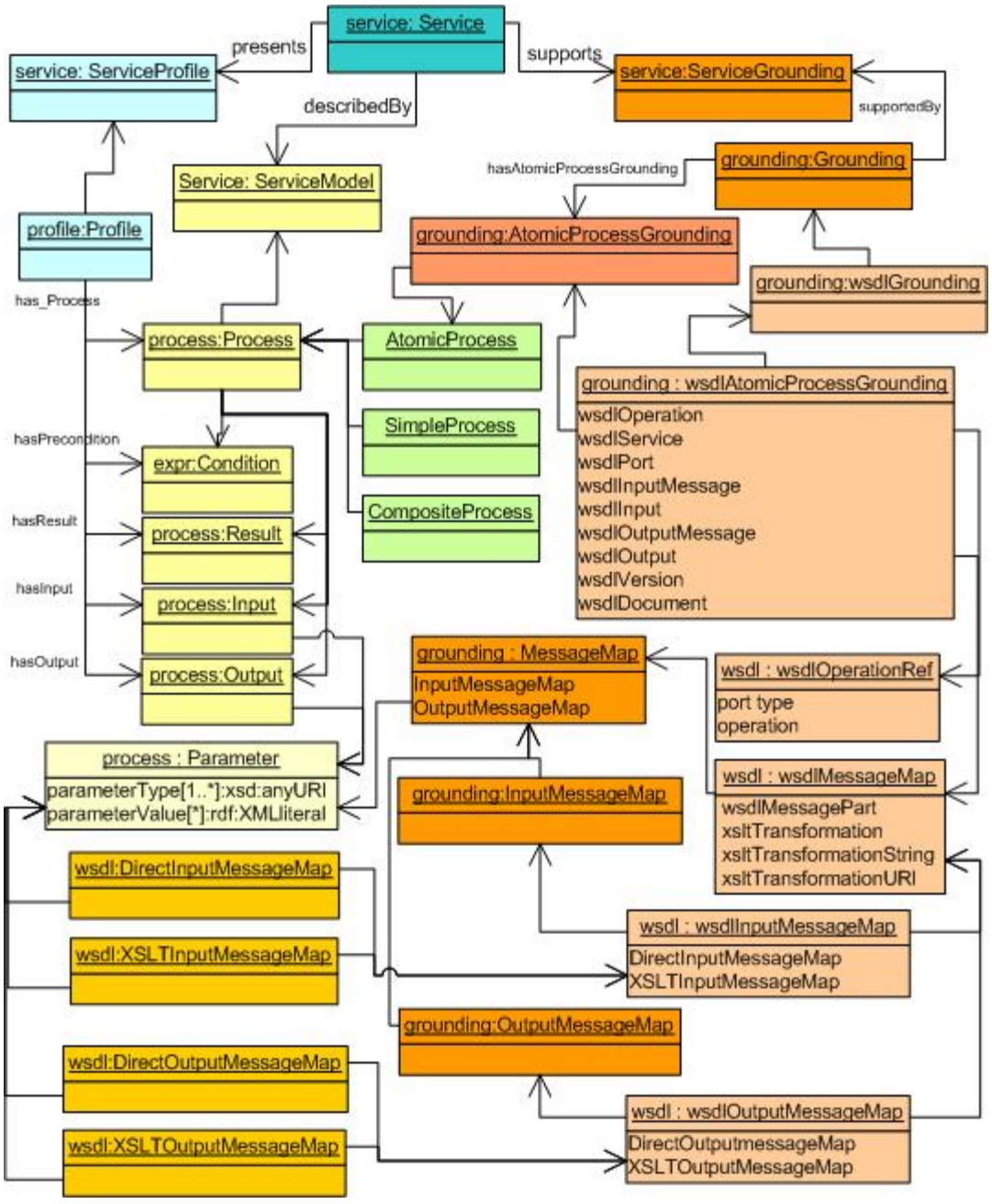

Figure 11. The OWL-S grounding

Although OWL-S grounding provides incomplete automation model for the service invocation, as the current release leaves a lot for assumptions, there has been proposals to improve OWL-S grounding with SAWSDL. Similar to OWL-S, the SAWSDL aims to enrich WSDL with semantic annotation. As mentioned earlier, SAWSDL provides a standard means by which WSDL documents can be related to semantic descriptions, such as those provided by OWL-S and WSMO and it has been started to adopt for grounding purposes. The objective of SAWSDL use is to derive OWL-S grounding automatically from SAWSDL annotations [90]. SAWSDL provides two basic semantic annotation constructs by Model References and Schema Mappings. SAWSDL attribute modelReference is a semantic model reference from elements in WSDL or XML Schema to concepts in a semantic model, usually ontology or taxonomy via URIs [90]. Model references can be used on WSDL interfaces, operations, message parts, and on XML Schema elements or types. Model references can have many uses, they can provide a classification of a WSDL interface, what a WSDL operation does and define the semantics of the inputs and outputs of WSDL operations [90]. Similarly OWL-S and SAWSDL acknowledge the importance of expressing the category of a service within a given taxonomy. SAWSDL provides category information by annotating interface definitions. OWL-S provides this information in the Profile through its type specification or through the property serviceCategory [91].

Other similarity is that both OWL-S and SAWSDL support the use of transformations to map WSDL messages to OWL, typically mapping rules implemented with Extensible Stylesheet Language (XSL) Transformations [91]. As OWL-S uses OWL as the ontology, any data can be represented in RDF/XML and therefore, for transformation between the XML data and the semantic ontological data can be used XML transformation language such as XSLT [85]. In SAWSDL lifting schema mappings specify how XML Schema types for WSDL type definitions are transformed to a semantic model, whereas lowering schema mappings do the opposite translating semantic model data to express in an XML document [92]. Accordingly SAWSDL annotations are in OWL since OWL-S does not handle any other type of semantic annotation [91]. These schema mappings are likely to be used with XSLT primarily, but the SAWSDL specification does not require XSLT or any other particular mapping

language. The schema mapping annotations are the only aspects of SAWSDL that are clearly intended for use only at runtime. The use of an XSLT syntax based transformation approach from OWL to XML is problematic, because there are generally a number of different ways that the same content can be serialized in OWL [91]. It can be quite complicated to write an XSLT script that handles all the different variants. However, OWL-S allows for the use of precondition expressions to bind variables to values in the semantic model typically values passed in as inputs, and it specifies that a runtime environment should pass these bindings into corresponding variables declared in XSLT [91]. The level to which this alleviates the problem will depend upon the manner in precondition expressions are written. In principle, it is possible to use preconditions to break down complex OWL individuals into primitive elements, consequently avoiding the issue of handling multiple possible serializations [90]. In addition, in SAWSDL preconditions and effects can be added as input and output model references only, which makes it difficult for any service matchmaker to identify them as such in general and before actually analyzing the name and content of referenced semantic models [93]. Another modelling problem is the handling of WSDL faults, because they can be represented in OWL-S with conditional results, but the problem is that there is no knowledge in SAWSDL of what are the conditions of a fault. SAWSDL specifies only the annotation of the semantics of content of the message, instead of the conditions under which the fault occurs [90]. These problems can be overcome by adding a specification of preconditions and effects to SAWSDL [91] and in the future METEOR-S working group hopes to influence to the standard by introducing these concepts [94].

The use of SAWSDL within OWL-S grounding would accumulate a thin OWL-S semantic model that contains a Profile specifying at least the service category and a Process Model specifying only the atomic processes which also are partially specified since SAWSDL does not provide yet any information on their preconditions and effects. The major semantic service descriptions such as OWL-S and WSMO were not designed for mobile usage as so because of their magnitude. More lightweight approaches have been proposed such as MicroWSMo annotation mechanism for RESTful services and WSMO-Lite to annotate WSDL descriptions using SAWSDL allowing users to add semantics with a shorter learning curve and less intrusive way [95]. These standards are

aimed to be lightweight and compatible extensions of current standards in Web Services [96]. WSMO-Lite proposes a bridge between WSDL, SAWSDL and domain specific ontologies, but it does not prescribe concrete language for functional semantics and as effects and conditions are language-dependent, it cannot specify semantics for them [97]. Although these approaches are lightweight, OWL-S provides already well defined semantic structure and also lighter model with SAWSDL usage in grounding serving this way inventive approach to create service descriptions through mobile device.

6. TOOLS FOR SERVICE DESCRIPTIONS CREATION

The ontologies are central enabling technology for the Semantic Web being key components in many applications including search, e-commerce and configuration. However, the creation of ontologies is not trivial or easy tasks. The development of ontologies demands the use of various ontology building environments and software tools. Tools that provide can be applied to several stages of the ontology life cycle including creation, implementation and maintenance of ontologies are called ontology editors. Building environments are used for building a new ontology either from scratch or by reusing existing ontologies, which usually supports editing, browsing, documentation, export and import from different formats, libraries and they may have attached inference engines [98]. in order to be able to create OWL-S ontologies, in this chapter is given overview of main available ontology building environments and ontology editor tools for OWL based ontology creation.

6.1 Development environments for OWL-S descriptions

Protégé is one of the most widely used editing tools and it has been used by experts for domain modelling and for building knowledge-base systems because it enables the building of distributed ontologies on the Web with migration, integration and version control capabilities [99]. Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-specific support for creating knowledge models and entering data [99]. Furthermore, Protégé can be extended by a plug-in architecture and Java based Application Programming Interface (API) for building knowledge-based tools and applications [100]. In order to build applications that exploit ontology, an API is needed to enable access and manipulation of ontology written in OWL.

The main framework that is used for this is the Semantic Web Framework for Java called Jena [101]. By providing RDF and OWL API and with SPARQL use as a query language to access RDF or OWL data, Jena allows straightforward manipulation of the RDF and

OWL [102]. Jena is capable of manipulating and parsing RDF statements, and storing them as graphs, but is much more than just RDF parser because it is capable of converting OWL specified ontologies to an RDF form [102]. To treat OWL, in Jena ontology is treated as a special type of RDF model, OntModel. Additionally the implementation is bound to a particular RDF implementation. The Jena Ontology API is language neutral as the Java class names do not mention the underlying language, because the OntClass Java class can represent an OWL class, RDFS class or DAML class [102]. To represent the differences between the various representations, each of the ontology languages has a profile listing the permitted constructs and the names of the classes and properties. In addition, Jena supports both memory base and persistent ontology models [101].

In the integration of Semantic Web Services more closely with the programming environment used to develop the service implementations, the integrated environment for OWL-S service construction and deployment (OWL-S IDE) developed by Carnegie Mellon University provides a plugin for Eclipse integrating the semantic markup with the programming environment [85]. Java code can be written in Eclipse and by running Java2OWLS tool to generate OWL-S directly from the Java sources [85]. However, OWL-S IDE does not provide any graphical visualization of services or processes. To visually build OWL-S services, the Java based tool DINST offers the functionality to edit service description in a purely graphical manner in which service creation process sets up layered service ontology where OWL-S is used as upper service ontology [85]. Another Carnegie Mellon University's OWL-S development environment (CODE) [103] is implemented as Eclipse plugin supporting developers of Semantic Web Services from the generation of Java code to the compilation of OWL-S descriptions, to the deployment and registration with UDDI. CODE provides a WSDL to OWL-S translator (WSDL2OWL-S), Eclipse-based editor for creating and editing OWL-S profiles, process models and groundings, and the OWL-S Virtual Machine to automatically generate the client code and execute it [104].

A Hypermedia-based OWL Ontology Editor, called Swoop, is ontology engineering environment developed by the MIND lab at University of Maryland. Similar to Protégé, Swoop is used to edit ontologies [105]. As Swoop is inspired by the hypermedia, it is Web-based OWL ontology editor containing multimedia markup extension. The Web ontology editing browser contains OWL validation offering various OWL presentation syntax views [105]. In addition, it has reasoning support and provides a multiple ontology environment where ontologies can be compared, edited and merged [105]. Another ontology building environment, Adaptiva is developed by the University of Sheffield [106]. Adaptiva implements a user-centered approach to the process of ontology learning and it is based on using multiple strategies to construct ontology, minimizing input by using adaptive information extraction. A tool has been integrated with a rule learning system Amilcare, for adaptive Information Extraction from text designed for supporting active annotation of documents for the Semantic Web. This tool focuses mainly on knowledge acquisition for knowledge management [106].

6.2 Development tools for OWL-S descriptions

The OWL-S Editor enables development of semantic descriptions for Web Service [107] leveraging the existing functionality of popular ontology development tool Protégé and provides a comprehensive set of capabilities for creating and maintaining OWL-S service descriptions by utilizing Protégé pluggable architecture to extend it [100]. The OWL-S Editor is built on top of the Protégé OWL Ontology Editor, allowing users to build complex process models using a graphical user interface, hiding the complex constructs. With this editor users can query and visualize the knowledge base and to export it to different formats. As the main point of interaction, OWL-S Editor presents the user with a tab inside Protégé where the OWL-S tab is separated into two parts, one lists OWL-S instances divided into service, profile, process and grounding instances and another tab is to show a specialized editing mode for the chosen type of OWL-S instance. Other functionalities are the WSDL support, IOPE management and a special window shows the relationships of all top-level OWL-S instances graphically [107].

Similar to OWL-S editor, the Ontolink generates executable OWL-S descriptions from WSDL files [85]. This tool generates semi-automatically grounding specifications by letting the user define mappings between ontology and XML schema definitions through a graphical user interface. Mappings between ontologies are generated using XSLT and automatically wrapped inside a translator service to transform the output of the service to compatible format for service consumer [85]. Another OWL-S editor tool provides a graphical user interface for users to create Semantic Web Service descriptions while using WSDL standard UML Activity Diagrams [108]. Web Services description development methodologies are based on a mapping from WSDL to OWL-S and on modeling a composite service for control and data flow using standard UML Activity Diagrams. OWL-S description is generated entirely from a WSDL specification and therefore no direct user input is required to do mapping [108].

OWL-S API is a developed Java API that supports the majority of the OWL-S specifications to read, execute and write descriptions. The API provides functionality for parsing, serializing, validating, reasoning, and executing services for various versions of OWL-S. This Java API provides functionality for parsing, serializing, validating, reasoning and executing various versions of OWL-S services [109]. The Specification and Execution tool (SPEX). The SPEX tool supports model-driven approach based on UML and allows the developer to complete the overall development cycle after the UML modelling activities have been completed. UML activity diagrams must be specified using the composition pattern. This requirement limits the flexibility that the modeller has to create optionally complex activity diagrams and imposes a strict layered approach. The SPEX tool has been integrated with the OWL-S API to provide a simple mechanism for specifying input parameters and inspecting output parameters using a graphical user interface [110]. Also another model-driven approach describes Semantic Web Services using UML [111]. In this approach mappings are created between the UML class diagram and subset of ontology modelling constructs. This is expressive enough to specify of the OWL-S ontology using a combination of UML class, activity and sequence diagrams. However, this approach does not support the specification of service grounding.

All above overviewed ontology building environments and tools are suitable for OWL-S creation, editing and maintenance. As mentioned, to ontology creation is complex tasks and done mostly manually. To help developers with ontology creation, these Semantic Web tools are aimed to help by visualising, modelling and partly automating to creation process. Nevertheless the great attention that Semantic Web tools are receiving in the computer science area, none of above overviewed ontology building environments and tools is designed to use in Mobile computing environments. The next chapter reviews current efforts in the area of ontology usage and creation in mobile domain.

7. SEMANTIC SERVICE DESCRIPTIONS IN MOBILE COMPUTING

The next generation of mobile systems is arising as new generation of mobile device users can explore the mobile Internet with its new features, services and applications. Accessing services anywhere, anytime, and the irrespective of the network is essential to meet users' requirements. Semantic web technology and the arrival of universal and mobile access to internet services provides additional features like knowledge-based, location or context aware information [112]. The vision of a mobile Web in which the computing environment will be composed of various devices that are carried by different users as they go through their daily routine becomes a reality with the improvements of wireless networks, bandwidths and client device capabilities [113]. Semantic Web Services provide enabling technology to make this vision a reality. Although the necessity of Semantic Web technology and in particular ontologies for realizing a context-aware mobile computing infrastructure is broadly acknowledged, most of the services available on the Web are designed to be accessible from desktops and PCs and currently Semantic Web based applications and services can scarcely be deployed on mobile devices because of the lack of suitable implementations. In order to realize flexible mobile access to distributed Web resources it is needed to combine methods from the Semantic Web and the service oriented approaches as put forwards by the Web Services community enabling the provision different services and information sources in machine understandable and truly intelligent ways.

The pervasive computing uses web technology, portable devices and wireless communications to facilitate ubiquitous access. Ubiquitous access can be implemented on a variety of devices, such as PDAs, laptops, mobile phones and information appliances such as digital cameras and printers. Main goal of pervasive computing and its ubiquitous access is that the mobile users get transparent access to resources outside their current environment [114]. The pervasive computing aims that applications can make better use of the underlying networking facilities and computing devices without requiring complicated interactions with the users [115].

Ideal is that a mobile user may enter a new environment, get the semantic descriptions of the available services and automatically connect to them in order to gather information about the environment or to access new services. Despite the progress of Web Service computing and even though increasing number of mobile devices supports Web Service technology, the mobile usage of Web Services is not as common as it could be and still factitious rather than naturally evolving [116].

The enabling technology for realising mobile interaction with Web Services enhancing the interoperability, extensibility, expressiveness and independence of Web Services is the Semantic Web Service technology. The key factor is the service descriptions as they can be reused and adapted to different mobile devices, target platforms, user profiles and interaction designs. The Table 5 demonstrates how service descriptions are the ultimate support for realizing the visions of different computing technology fields. In the first line of Table k are listed the technology fields which merge together in essence to enable Semantic Web services in mobile environments. The technology field that combine here are the pervasive computing with ubiquitous access in mind, the Web technologies with their basic standards, the Service Oriented Architecture with its service triangle and the Semantic Web technology driving the power of semantics. Below of these computing technology fields is presented the goals, that is, their visions that their technology enables. Under of computer technology fields visions is the service descriptions and explanation how service descriptions enable realize these visions.

Table 5. The service descriptions enable visions of different technology fields.

Pervasive computing	**Ubiquitous access**	**Web Technologies**	**SOA**	**Semantic Web**
Mobile computing Wireless technology Smart environments/devices Location awareness context awareness Device aggregation/interoperation Human-computer interaction		UDDI SOAP WSDL Mobile Agents	Service triangle	Ontologies Semantics Semantic description languages
Self-adaption to enable application and system behaviour change, switch different processing state Application mobility to support transfer to any device and migrate itself to continue computation Application and supporting systems sensitive to context information to achieve best perfomance		Data access and exhange to various clients	Service repository for service advertise-ment	Ontology capture and describe knowledge in explicit way Semantics provide sensitiveness of context
SERVICE DESCRIPTIONS				
Service descriptions enable automation of functionality sharing in pervasive environments	Service descriptions extend UDDI/SOAP with service capability based discovery	Advertised service in repository matches requested one by service descriptions	Semantic service descriptions define service functionalities in machine interpretable way	

A key challenge in the domain of ubiquitous computing is to assist the users proactively. Without the usage of semantic description approaches, even partial automation is impossible. However, all semantic description approaches described in chapter 5, require quite complex service description structures, which poses a problem in mobile computing already. In the case to support mobile peer-to-peer (P2P) environments, for example in ad hoc networks functionality sharing and interoperability of Web Services, it is not appropriate to expect users to put a lot of effort into devising correct descriptions of the services their devices or applications can offer. If there were tools to easily create service descriptions and a place on the internet where service providers and users could take advantage of them with their devices, the Semantic Web could help to bring the vision of intelligent environments to life. Motivated by service description creation on small devices such as mobile phones, the next sections views some current efforts towards better interoperability of Web Services and mobile devices though semantic service descriptions written in OWL-S.

7.1 Solving challenges of mobile computing with semantic descriptions

As well as the Semantic Web technologies support service lifecycle tasks in traditional Web, in the mobile computing environments Web Services should be able to discover, compose, invoke and interoperate automatically with each other. This section is about the technologies and methods empowering and enabling Semantic Web technologies in mobile computing scenario where context awareness is fundamental. Web Services provide a loosely coupled infrastructure for service description, discovery and execution. In the traditional Web Services model, service requestors find the appropriate service by placing a request to the service registry, often implemented with UDDI, obtain the results of the chosen services expressed in WSDL and send SOAP messages to Web Service provider. Web Services are designed for fixed networks and accessed from desktop PCs, with a fixed and low error connection to the network [112]. Consequently the main objectives of recent efforts are to extend current services and applications designed for fixed networks to mobile users in a seamless and transparent way. The main problems arising in the traditional model, is that UDDI service discovery is performed primarily by service name, precisely keyword matching and not by service capabilities. UDDI tModels may be regarded as a vocabulary where service descriptions are unstructured and intended for human comprehension. Therefore, different services with the same capabilities can be categorized in different business categories. Other problems are that WSDL does not contain any information about the capabilities of the described service and that, SOAP messages overhead is often greater than the service parameters exchanged between communicating parties. These challenges can overcome by semantically enriched Web Services expressed in OWL-S. In addition, providing wireless access by using Mobile Agents enables query and invocation semantically enriched Web Services without the need for simultaneous, online presence of the service requestor. This kind of service setting is ideal for mobile computing, where user terminals are not necessarily online during their entire session [117].

Mobile computing systems and architectures make use of distributed environments and both service discovery and its automation are the key features that a large scale open distributed system must provide in a way that clients and users may take advantage of

shared resources. The semantic service discovery involves discovery and semantic matchmaking. An OWL-S-based service uses the profile to advertise existing and requested capabilities accordingly used for the discovery task and mapped onto UDDI [118]. The OWL-S matchmaker extends UDDI with a capability port (APIs) to send, receive and process semantic queries. A layer over UDDI is developed to handle semantic data, which is based on the mapping between OWL-S profiles and UDDI records through tModels since OWL-S provides a capability based mechanism and in the same way the mappings that link the model to the required WSDL files. The software includes the jUDDI registry and an OWL reasoner such as RACER, in order to process the OWL-S description present in the UDDI advertisement [118].

In OWL-S both service profile and process model are thought of as abstract specifications of a service. In order to actually interact with a service it must be bound to grounding providing the necessary details of how to embody this interaction in terms of message format, transport protocol and addressing. The most widely used grounding employs WSDL 1.1, which is included in the OWL-S release. More recent initial deal with grounding based on WSDL 2.0 and relation to SAWSDL. The achievement of UDDI registry able to process SAWSDL descriptions is related to the possibility of an analogous mapping of WSDL 2.0 definitions in UDDI that will enable UDDI queries based on artifacts and metadata. The first projects about this interaction are for example inside the METEOR-S project [119] that with the Lumina component aims at supplying a unified WSDLS/ SAWSDL to UDDI mapping structures, which enables the user to discover the services based on the operation level [120]. A hybrid matchmaker is most convenient for semantic service matching in highly dynamic environments. For this, the OWLS-MX provides hybrid Semantic Web Service matching. OWLS-MX utilizes both logic based reasoning and non-logic based information retrieval techniques for Semantic Web Services represented in OWL-S [121]. The hybrid matchmaker has been successfully used in mobile e-health systems for emergency medical assistance and repatriation planning, the Health-SCALLOPS system and the CASCOM system [122]. The Context-aware Business Application Service Co-ordination in mobile Computing Environments (CASCOM) research project combines agent technology, Semantic Web Services, P2P technology, context-awareness, and mobile computing for intelligent peer-

to-peer mobile service environments in which services are provided by software agents exploiting the coordination infrastructure to efficiently operate in highly dynamic environments [122].

One challenge in mobile computing is to bring services directly to mobile users, because of mobile phones and services do not interoperate as effortlessly as they should and delivered services have to be adapted to a wide range of different mobile client platforms. Web Service and mobile device interaction has to rely on built-in browsers or proprietary service clients and interaction becomes tedious, complex and inflexible as mobile device generally suffers from small screens, fiddly keys and joysticks as well as narrow menus. This adds to the general problem of adapting mobile application interfaces to different devices, platforms and their individual properties and constraints [123]. One solution to this challenge is to extend Semantic Web Service descriptions with abstract interface annotations and use them for the automatic generation of adapted user interfaces. These interfaces support and facilitate the mobile interaction with physical objects and as a result the mobile interaction with corresponding Semantic Web Services [116].

One of the main benefits of OWL-S is the definition and assignment of self defined abstract parameter types which are required for the correlation between physical objects and service parameters. Abstract parameter types are part of the general knowledge between physical objects and associated Web Services [116]. In order to provide additional information for the generation and rendering of user interfaces, OWL-S service descriptions can be extended with additional ontology, such as the Service User Interface Annotation (SUIA) ontology [123]. Service User Interface Annotation (SUIA) ontology main class AbstractUIMapModel serves as a collection for several parameter mappings represented by instances of the class ParamterMap which are attached by the property hasParameterMap [123]. The information encapsulated in a ParameterMap complements the description of a specific input or output parameter in the OWL-S service ontology which is indicated by the object property hasServiceParameterRef. The remaining properties in the ParameterMap express information that is needed for rendering the interfaces and increasing their usability [123].

7.2 Initials for service descriptions on mobile environment

Mobility, context awareness and ubiquitous access are goals of future computer networks. As services and devices are promising to become more and more pervasive consequently the development of mobile devices and applications able to support this continuous cross of roles in different devices and environments become increasingly important. This was a major goal of MobiLife project focusing context awareness, privacy and trust, adaptation, semantic interoperability and their embodiment in novel services and applications that mach key use scenarios of end users everyday life [124]. The goal of the MobiLife project was to bring advances in mobile applications and services within reach of users innovating and deploying new applications and services based on the evolving capabilities of 3G systems and beyond [124]. MobiLife uses within its mobile services architecture an upper ontology that is a version of OWL-S called MobilOWL-S, specialised to represent services for mobile computing. MobilOWL-S is an extension to OWL-S aiming to enrich ServiceProfile description with both functional and non-functional properties of a service with a representation of the type of service and with a representation of wide range of parameter types and with specification of taxonomy of services [124]. The goal of MobiLife with MobilOWL-S is that he service provider and service requester in turn use the best combination of features to describe what services they provide and what services they expect [124]. MobilOWL-S promises good specialization of OWL-S for mobile computing, but seems to be still at initial stage.

Inspired by mobile user-centered services on the Semantic Web, the authors of MobiLife research have presented MobiOnt and MobiXpl which are early prototypes for semantic based service discovery on mobile terminals [125]. Conceptually, MobiOnt serves as a semantic service portal implementing semantic service ontology lookup and service discovery and the MobiXlp mobile application allows users to explore relevant service categories and to express their preferences in a graphical way [125].

Semantic Web technologies are usually incorporated in the infrastructure of desktop and web applications, but currently can not be entirely deployed on mobile devices because of the lack of suitable implementations for semantic information storage and management.

Many recent projects make use of Semantic Web technologies, but the processing of contextual data is done on external servers and applications rather than on the device itself. Distributed service frameworks for semantic information handling and management are useful for a variety of scenarios, but involve risks in case of connectivity problems and server failure.

Few research studies integration of Semantic Web technology onto mobile devices and employ technologies such as RDF for the realization of a powerful, adaptable and scalable context-aware infrastructure for capturing contextual data that can be used for requesting ambient services and data sets from linked data repositories. In the work of context-aware approach for integrating Semantic Web is aimed to facilitate the deployment of context sensitive mobile Semantic Web applications by a multi-layer architecture where each layer is responsible for a certain group of context-related tasks such as sensing, acquisition, aggregation, interpretation, inference and representation. The research takes into account the effectiveness of processing and storing capability of triple-based data on devices with limited processing power, memory and power capacity [126].

Another similar research is about mobile RDF replication and synchronization taken its motivation of Personal Information Management (PIM) [127]. Personal Information Management (PIM) has got its interest in the form of Semantic Desktop [128], which has been the starting point inspiration also of this thesis. These both studies utilize the Jena Ontology Management API for Mobile Clients called Micro Jena (µJena package). Micro Jena prototype, mostly called as µJena, was developed in the Politecnico di Milano University [129]. The µJena is a lightweight porting of the Jena Ontology API, being a reduced version allowing process RDF serialized in N-Triples format [130]. The open source µJena is focused on the design and development of an API for handling RDF and OWL ontologies on mobile devices. In these researches of adaptive RDF data replication on mobile devices has come up with the conclusion that µJena provides most advanced framework for ontology and inference support for mobile environments of other RDF frameworks such as Mobile RDF [126].

7.3 Semantic information processing on mobile scenario

The greatest challenge raises the fact that mobile devices capabilities are limited in terms of computing power, memory capacity and network bandwidth. Distributed contextual data management can be problematic for example because of server breakdowns or limited cellular radio coverage availability and also the data transfer can be expensive. Because of connectivity uncertainty and mobile capability restrictions, it is important to have feasible solutions of semantic information management for mobile devices. As mobile capabilities are restricted, it is not realistic process very large semantic data sets entirely on mobile device. On the other hand, the connectivity problems require that some semantic information processing should be possible on mobile device as there is no network availability.

There have been recently few researches about information organization in terms of Personal information management (PIM) towards on mobile semantic desktop [131] and semantic information processing in terms of RDF replication on mobile device [127]. Although information management and filtering is crucial factor within realization of context aware mobile settings, the implementation of semantic desktop is directed towards desktop computers as organization of data on mobile devices is far more difficult because lack of storage capacity [131]. The idea of Semantic desktop is integrate desktop applications and the data managed on desktop computers using semantic web technologies [132]. The Semantic Desktop approaches utilize ontologies to build conceptual relationships between resources and define a concept hierarchy that can be utilized for information retrieval. The SeMoDesk is based on Semantic Desktop ideas aiming to support personal information management on PDAs by letting users to define and manage their personal ontology in order to improve context sensitiveness of mobile systems in terms stored personal information related to user's current locations and time.

In the SeMoDesk system-centric view of context-awareness is replaced by a user-centric view. The recent research of developing a framework allowing for adaptive RDF graph replication and synchronization on mobile devices takes part to personal information management handling as no network connectivity is available. In the work of adaptive

RDF graph replication on mobile devices has been implemented an Android application based on the μJena packages for retrieving N-Triple serialized RDF data that can be stored and processed directly on a mobile device [127].

Another similar research to adaptive RDF graph replication discusses how existing semantic knowledge bases can be utilized in creating annotations on mobile devices taking part to the problems of different access methods of semantic knowledge bases designed primary for desktop applications instead of mobile applications [133]. Many of many applications are being developed using the Android SDK. The popular ones among mobile application platforms are the Java Platform, Micro Edition (Java ME) and Google's Android mobile platform. The main disadvantages of Java ME are a slower application development and performance. The Android SDK is a set of tools developed by Google to facilitate development of mobile applications using Java and overcomes Java ME limitations by providing APIs to build richer applications [112]. As there is no unified way to access the semantic knowledge bases and as they are typically designed to be accessed by desktop web browsers, it is argued that a quick way to implement the access to the knowledge bases from a mobile application is to use a Web Service interface.

Considering access to the semantic knowledge base, for example if a Java ME or Android application is being developed for mobile devices that support the Web Services API, it does not take much work to generate stubs from the WSDL description and start accessing the knowledge base. However, many of the knowledge bases do not provide these interfaces, it is often required to implement the access through HTTP requests and responses, which however means more work especially as different semantic knowledge bases offer different access methods. In addition, knowledge bases use different protocols and for example, DBpedia and Freebase knowledge bases can be accessed by HTTP GET requests, but the requests to Freebase must conform to Metaweb Query Language (MQL) while DBpedia mobile uses SPARQL queries [133].

Although above discussed researches and projects are important towards semantic information usage on mobile scenario, they focus merely information representation and

retrieval in RDF from. In order to provide mobile access to internet services and to ensure Web Services interoperability, is needed to use more expressive semantic service description language, such as OWL-S. The main interests with these researches and projects are the point of view of contextual information management with the idea of Semantic Desktop and the usage Jena Framework and its compact form µJena. In addition, none of these focus on service creation itself on mobile device. The only projects related to user generated services, discussed in chapter 2, such as the m:Ciudad and OPUCE. However, these projects focus also on their own platforms where certain types of services can be produced on specific platform designated way. This limits the innovativeness in large scale as open possibilities should be unwrapped for every common user within Web Service creation on mobile device.

7.4 Information system design architecture on mobile platform

Model View Controller (MVC) is standard design pattern is one of common architecture especially in the development of rich user interactions GUI application and in web domain [134]. Model View Controller (MVC) is known for its extensibility, maintainability, re-usability and testability capabilities [134]. MVC architecture defines three classes of modules dividing an application to a data model, controller and view as showed in Figure 12. A model contains all the data, a controller is responsible for the data flows and manipulation within the application and one or more views upon the data present the information to the end user [135]. The interaction between the view and the model is managed by the controller being an event listener that responds to an event and changes the model or the view [134].

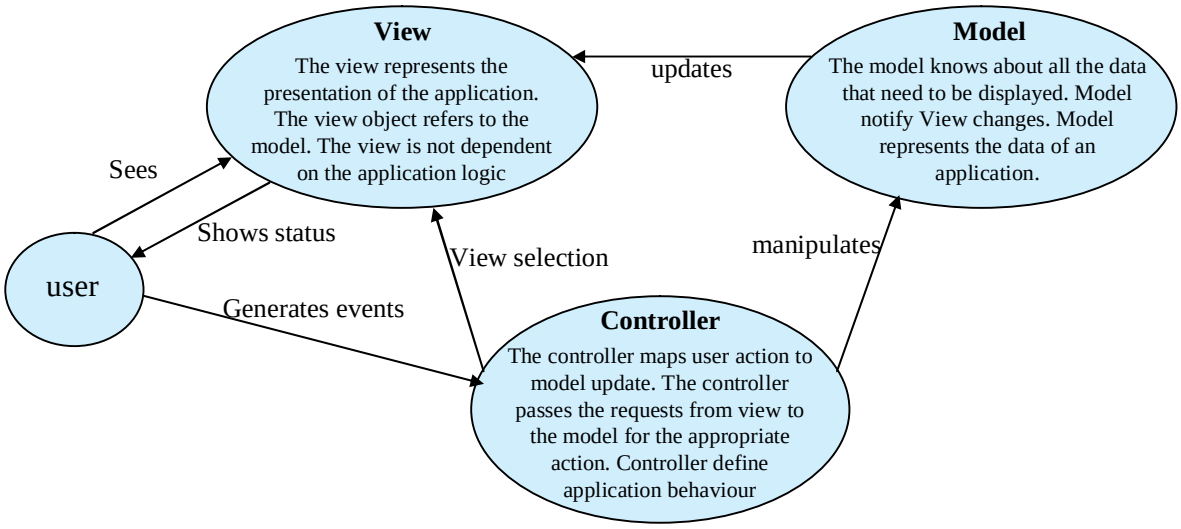

Figure 12. The Model View Controller architecture

MVC assumes that a model does not know which views are plugged to it and therefore architecture allows multiple views on the same model and a decoupled change of the views. The MVC architecture provides a strict separation of view and controller which may introduce an extra complexity to the system. Controllers cannot be fully decoupled from the views because they have to understand the events triggered by the view. As a result, each view needs its special controller. However, it is possible to integrate view and controller into a single component without the strict MVC separation. For example Modern GUI libraries such as Swing are based on MVC architecture integrating view and controller [136]. For instance, the Swing list has two classes, JList which corresponds to view and controller and ListModel which corresponds to the model. Swing provides a model-view separation for most of its components [137].

The MVC architecture in general reduces performance. Each update of the model causes an update of the views and a user event causes the controller to contact the model. This is critical when model and view are distributed over a network, for example in Web based client server applications [135]. Thus model-view interaction has to be brought to a minimum and respectively a part of the model has to be kept on the client side. MVC fosters a transparent reuse of model components independently of their visual representation. For example a typical model element is a list box with the associated buttons for adding, modifying and deleting list items as well as an editor dialog for

editing them. The logic of the list box such as list handling and buttons is quite simple and accordingly can be easily reused [137]. The view of the list box can vary significantly because for example the list view could be a single- or multi-column list or a combo box and the buttons could be right aligned or bottom aligned. A button could be left out or exist twice and so on. By separating model and view a developer can reuse the model component but is not restricted to use a certain view template [137].

8. USE CASE FOR MOBILE SEMANTIC ASSISTANT

As this thesis is about enabling the end users to create service description of their own services, the key questions is about why end users should be empowered or involved with service description creation. The answer can be found from the current projects reviewed in chapter 2, which are all about making services available and empowering the non-technical common Internet users with different devices and platforms by making tools for them to realize future visions. Considering the enabling technology, the Semantic Web aims to machine interpretation that services can be automatically discovered, invoked, composed and reused. Although Semantic Web technologies are recognized and deployed in traditional Web environment, is has not yet been deployed as well as it could be on mobile computing scenario.

Web Service technology is capable of deliver information in meaningful and powerful way through the Internet and Web Service languages greatly facilitate the description of networked applications as well as their interoperation with clients. Because of this, the Semantic Web Service technology provides the most promising methods for connecting mobile devices to pervasive services. Ontology provides specification of a conceptualisation of a knowledge domain and here knowledge domain includes the services themselves. Conceptualisation of a knowledge domain includes information that what types of services exist and how they relate to each other. By using ontologies, it is possible to infer about the properties of services such as types, relationships and hierarchy as well as the personalisation support of different services, even if different terms are used to represent the same ideas. As ontologies are machine understandable, as such a computer can process data with references to ontologies. A computer can infer facts from the originally provided data through the knowledge encapsulated in the ontology. The use of ontologies enables systems to share common understanding of the structure of information and reuse of domain knowledge, make domain assumptions explicit and separate domain knowledge from the operational knowledge. The knowledge based service description creation with ontologies are left on the hands of professionals and still the trend is moving towards that end users should be seen as service providers

and in addition the systems where the Web Services are used are moving towards distributed systems until pervasive environments including set of resources such the Web, mobile devices, PCs, networks, Web Services and users. Because of this emerging set of service usage anytime with any device in machine-interpretable way, has to study what are the available technologies to realize empower end users and which are the currently available tools suitable for service description creation on mobile device with limited memory and process capacity.

What is needed here for empowerment of end users are ways for sophisticated user support, resulting in semiautomatic generation of service description. Knowledge management and semiautomatic service description creation is supported by a semantic assistant. Semantic Web technology and mobile access to Internet services provides additional features such as knowledge-based and context aware information. For this, an upper ontology for Web Service description OWL-S is employed. OWL-S provides well-defined semantic structure to describe services with their capabilities and functionalities. One advantage to use OWL-S in the semantic assistant for mobile device is the creation of service descriptions more machine-interpretable and therefore allows accessibility and dissemination of services created, provided and requested via small portable devices. The idea behind the Semantic assistant is to generate complex service description on behalf of the user. The greatest challenge is to employ knowledge management tools that are lightweight enough to fit in mobile scenario and same time powerful enough to be able handle complex semantic descriptions of Web Services. In the next section is described the use case of Mobile Semantic Assistant on mobile device.

8.1 Description of use case for Mobile Semantic Assistant

The main goal of Mobile Semantic Assistant framework development is to support common user with description creation beside Web Service development on mobile device. The user involvement within service creation in pervasive environment and Semantic Web technologies in mind, the Mobile Semantic Assistant aims to automate the creation of semantic descriptions. Considering the use case for semantic assistant deployed on mobile device, the basis is to have a Web Service that is discoverable and

interoperable in machine-interpretable way that any client can use the Web Service. The starting point for Mobile Semantic Assistant is, assuming that in the future and with emerging technologies and portable devices is possible to create any Web Service, the user is able to create the semantic description without the need to enforce the user to use other application for description creation and leave the place or device with which the user is working. The main idea beneath Mobile Semantic Assistant is to use existing tools with possible modifications to create OWL-S service descriptions that can brokered through SOA.

Considering the use case scenario, the user is some business owner handles all every day businesses with his/her mobile device, let's say a mobile phone. Business owner can be a hotel, restaurant or car rent owner or goods vendor. Considering the scenario where for example a hotel owner wants to expand his businesses through the Internet. The business would be available as a Web Service in Internet. Most effortless way is to create Web Service and its semantic description for the user is to provide to the user possibility to do all this by his/hers most used device, through a mobile phone. As the business owner publishes service in Internet, because of service semantic description, the service is discoverable and usable for any other clients.

Although the focus is on semantic description creation, it is important to show in which scenario the description is created and more importantly automatically discovered and invoked. The Figure 13 presents the use case of overall Web Service semantic description creation and usage within SOA concept. In the first case the user input semantic description of the Web Service. The OWL-S described service is published through mobile service provider on the Internet. In the second case the service provider publish service description to Proxy server such as UDDI registry. The network connection, that mobile phone uses, can be for example Wi-Fi, GPRS or 3G. In the last phase any client, be it mobile or desktop client, can find the service description in UDDI registry and the matchmaker, such as OWL-S MX selects OWL-S described services to satisfy a given query.

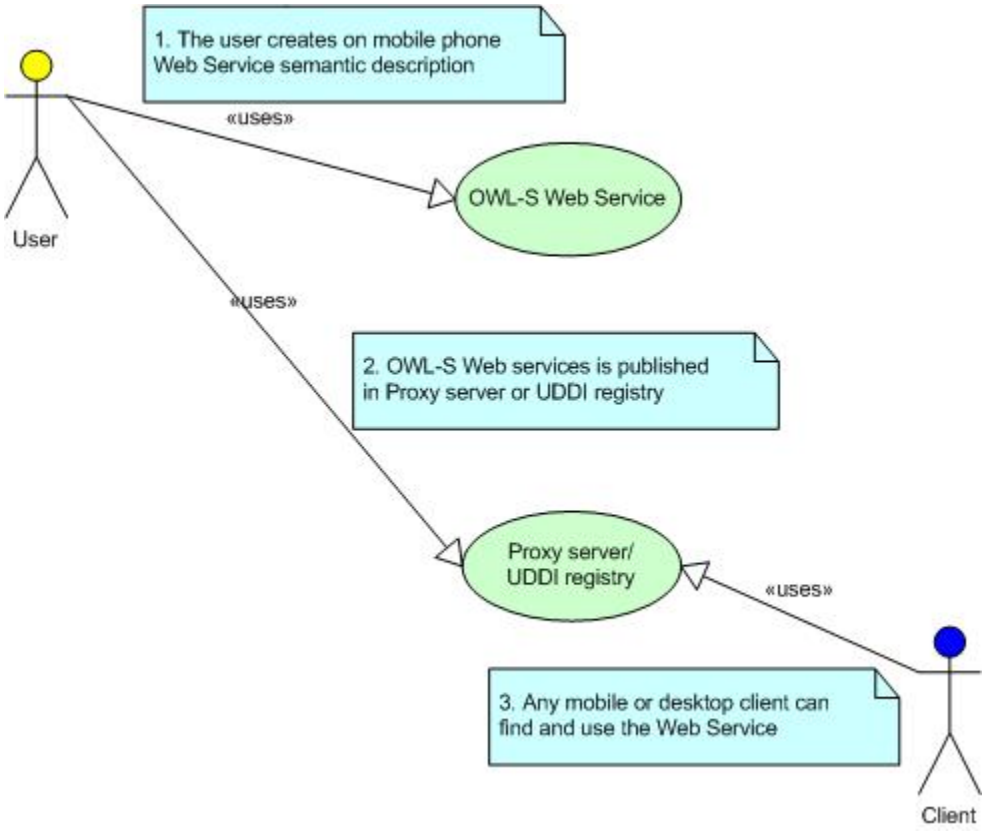

Figure 13. The use case of overall service description creation and usage.

The focus of this thesis is on the first case, concentrating on service description creation done by end user effortlessly with feasible tools suitable for mobile phone. The previous sections have treated what tools exist for service description creation in OWL-S, how OWL-S described services are used in mobile scenario and what are the suitable methods as well as tools for knowledge management to be utilized on mobile devices. The next sections explain in detail the core idea of service description creation on mobile device and the framework of Mobile Semantic Assistant.

8.2 Service description creation on mobile device

Service description on mobile device should be as effortless as possible. The difficulties bring the knowledge management since it requires a lot memory space and process capacity. Very typical for user centric programs is that what convenient the system is and preferable for common user to use, that more complex processes the system runs behind.

In mobile scenario both the service description creation and generation process should be straightforward in order to add value and assist user without taking too much capacity and time. The service description creation part should be for user simple enough to use and logical to understand the functions. The end users are the first priority took into account in system design. The end user is considered here as prosumer or professional amateur who wants produce upgrade or advance services related to his/her own every day life and professional field. In end user perspective is assumed that user is not interested in underlying technology, but if willing to explore the service creation, for user is given possibility to create service with own determination of all aspects of OWL-S. As discussed in chapter 2, the ability to modify applications to suit ones own needs is major step towards future open ended systems.

In Mobile Semantic Assistant design is aimed at facility of modification that the outcome is what the user needs. This can be achieved by giving range of selection choices to user. In addition, Mobile Semantic Assistant design is aimed at simplicity on functional side in order that it is customizable either by a common user or a programmer if willing to go deeper to formal logics. Considering the service description itself and conceptualization, ontologies and construction of semantic description are the core of what a common user may not have any knowledge. As construction of service ontologies and semantic descriptions is challenging for professionals, the service description construction has to be simple enough for a non-technical user in order to be able to build a complex Web Service.

The procedural of service description creation is presented in the set of Figure 14. Mobile devices, such as mobile phones have small screens, which bring additional challenges for the design of user interaction. The user interaction is aimed to design so that the user has possibility to configure Web Service description of own choices or select ready made configuration for different types of services. To understand how service description creation would look like for user, the set of Figure 14 presents the user interaction on mobile device. The templates for user interactions are provided by Mobile Semantic Assistant.

Figure 14.a

Figure 14.b

Figure 14.c

Figure 14.d

Figure 14.e

Figure 14.f

Figure 14.g

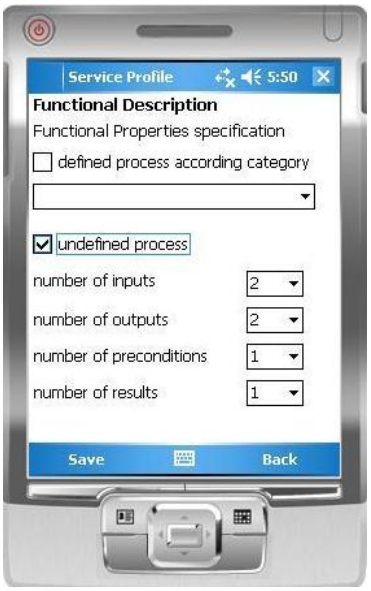

Figure 14.h

Figure 14.i

Figure 14.j

Figure 14.k

Figure 14.l

Figure 14.m

The set of Figure 14 show general view step by step of the generation of OWL-S description. The Figure 14.a shows the starting phase of OWL-S description generation in which the user can select the sub-ontologies for top level ontology Service and set the Web page, where the ontologies are located. The user can select which sub-ontologies are generated by selecting from OWL-S Service Profile and Service Model with the Service Grounding. As discussed earlier, the OWL-S Service Profile is required for service discovery. In addition, if in the OWL-S is included Service Model, there has to be included the Service Grounding as well in order that service is able to invoke. In the start the user input the Web page address and the system locates selected service sub-ontologies according that Web address generating for them own resources. By pressing next button, user continues to the next step of procedure. The Figures 14.b to 14.e show Service Profile generation with minimum settings required to establish OWL-S Service Profile. More precisely, in Figure 14.e is showed the non-functional description settings for Profile establishment. As shown in Figure 14.b, the user sets name for the service and gives a brief text description of the service summarizing what the service offers.

The contact information is expressed either with FOAF or VCard for the business. For service classification purposes the user can select for categorization either NAICS or

UNSPSC industry taxonomies as showed in Figure 14.b. Considering the service discovery, the ideal matchmaking occurs when both service provider and requester refer their classification to common ontology and therefore shall take existing ontologies into account. The user can select to refer NAICS to classify business and select proper NAICS code for service. For example, considering a hotel business scenario, the user would select from NAICS menu Accommodation and Food Services for business classification and Hotels and Motel for code as presented in Figures 14.c and 14.d. If no proper existing classification ontology or taxonomy for service or product exists, the Service Category option can be leaved out. By pressing save button the user continues to the next phase which is contact information creation if that has not been generated earlier. For contact information the user can select for example VCard to express business information as showed in Figure 14.f. However, if the contact information is generated already earlier on mobile device for example in device owner information, the Mobile Semantic Assistant system retrieves the contact information and applies it to generate FOAF or VCard. Ones FOAF or VCard is generated, Mobile Semantic Assistant utilizes this information on next service description generation times. Consequently, if the contact information is generated utilizing device owner information or FOAF or VCard exist in the Mobile Semantic Assistant files the phase of contact information creation as shown Figure 14.f is omitted. Supposing that user wants to generate solely the Service Profile for service advertising and discovery purpose only, the service description is ready after contact information creation which would be generated from device owner information or information has been set earlier, user moves from phase shown in Figure 14.e to the end of description creation shown in Figure 14.l service validation phase. After accepting Service Profile the service with one profile would be published. Supposing that user has selected in the start both Service Profile and Service Model with Grounding, the procedure continues to the functional description part of Service Profile after Profile's non-functional description creation.

The functional description creation procedure is shown in Figures 14.g to 14.i. Figure 14.g shows that the user can select either from defined process according the service category or either define own new process. In defined process the process is configured according the service category. The Mobile Semantic Assistant suggests apply the

service category selected on the Profile's non-functional description creation phase. For example if user has selected at non-functional description creation phase Hotels and Motels for business classification purpose, the assistant system suggests to use it as shown in Figure 14.g. Hence the next procedure is the revision of the inputs, outputs, preconditions and results giving to user possibility to correct this specification of what the service provides. If user has selected defined process, he/she can review the inputs, outputs, preconditions and results and is able also change them according his/hers service needs as shown in Figure14.i. In defined process the inputs, outputs, preconditions and results are set as well as their relation to the Service Model process. Properties Input and Output for process relate Service Profile to Service Model process:Input and process:Output same as properties precondition and result of Service Profile relate Profile to Service Model expr:Condition and process:Result. For new process definition is given selection of undefined process as shown in Figure 14.h in which user can define him/her self the process if none of ready configured process according service category do not match to the service. In undefined process, the user selects the number of inputs, outputs, preconditions and results. The next procedure is definition of functional description shown in Figure 14.i where user can write and define wanted number of inputs, outputs, preconditions and results.

The next step of service description creation is determination of Service Model process type and Service Grounding shown in Figure 14.j. In the Mobile Semantic Assistant framework is by default supported atomic process for Service Model process type. By selecting only an atomic process, the process description is exposed with inputs, outputs, preconditions and results defined in Profile functional description. By selecting atomic and simple process, the simple process is then used for abstracted views of atomic process. By selecting composite process, the process includes atomic, simple process and the process workflows including data and control flow. The basis is that the process workflows are defined by default for different service types. The composite process selection may need additional definition from user if user wants to create new type of service which process workflows are not defined by default. The OWL-S Service Grounding is the last part of top level service ontology creation. The Figure 14.j presents the selection of WSDL or SAWSDL specification for service grounding and the basis is

that these specifications are defined by default. The Figures 14.k and 14.l show the OWL-S description status and validation. OWL-S top level otology is successfully configured if the system gives valid announcement or it has to be edited if the system gives not valid announcement as shown in Figure 14.k. As systems gives to all valid announcement, the user can either still edit by selecting wanted part or move until publishing by selecting ok and pressing continue button. The Figure 14.m shows example of successful Web Service description creation announcement. The configured OWL-S Web Service is published by pressing publish button.

On the whole, as user uses the standard specifications for OWL-S description creation, the description with Service Profile, Model and Grounding is generated in seven steps. Assuming that user has set the owner information for contact information setting, when user wants to create only the Service Profile the description process consists of four steps including phases 14.a sub-ontology selection, 14.e non-functional description, 14.l description validation and 14.m description publishing. When users wants to create all parts of OWL-S description including Service Profile, Model and Grounding, by utilizing standard specifications of functional description the service description process consists of seven steps including phases 14.a sub-ontology selection, 14.e non-functional description, 14.g functional description, 14.i approve of functional description, 14.j Service Model process type and specification for Service Grounding selection, 14.l descriptions validation and 14.m publication of OWL-S description.

OWL-S is quite large Web Service ontology and therefore the creation requires number of phases especially as all top level ontologies such as Service Model and Grounding are desired to include. Indeed, the OWL-S is quite large of its structure and therefore the user interaction is made such way that it is easy for user to understand the parts of the OWL-S without excessive creation phases. As shown in the set of Figure 14, the creation of all OWL-S parts can be made in eight phases, if user has not set device owner information or when contact information has to be set at first time. Certainly the creation series can be made different, depending what is wanted to show to the user and which kind of inputs are required from the user. Above described example of OWL-S service creation indicates that effortless service description creation even with complex description

structures can be possible even on mobile device. However, very distinct view of the service description model is showed for the user, as all the complex structures are built on background in the systems. The next section describes in detail the whole framework of Mobile Semantic Assistant and the methods to create complex structures of OWL-S without requiring too much processing capacity of mobile device.

8.3 Mobile Semantic Assistant framework

The Mobile Semantic Assistant is a user-centric ontology creation and processing tool for mobile clients. The basic premise is that the service description creation is effortless, taking into account the non-technical users, and the mobile device's limited memory, processing capacity of and the small screen. In the previous section has been described the user interaction and the creation phases of complex service description. The user interaction presentation shows that even large and complex service description is possible to create with partially ready set ontology configuration. Provided partial ontology configuration enable the creation process is made minimal for user and the user does not have to go through too many phases. This method also enables, that for the user does not have to show very complex parts such as OWL-S control constructs of composite process. In addition, by providing appropriate construction selections can overcome limited screen view and tedious text input with keypad or touch screen typical for mobile phones as the user does not have to write excessively. As a result, the partially ready set ontology configuration is main requirement of Mobile Semantic Assistant framework.

In this section the overall framework of Mobile Semantic Assistant is described in detail. The service description creation system architecture is based on Model-View-Controller (MVC) design pattern. The Figure 15 shows the MVC model of Mobile Semantic Assistant framework. The advantage of MVC model used Mobile Semantic Assistant framework is the separation of user input, the presentation and the modelling of the service description based on OWL-S. In Figure 15, the controller presents the eventual core of Mobile Semantic Assistant. The model presents domain of OWL-S files creation supported by Jena Framework. The controller maps the user input from view to the model. In the model the OWL-S creation is updated and depending on creation stage the

controlled selects appropriate view for user. Precisely, the model includes configured OWL-S templates in to which the controller sends the updates and simultaneously manipulates the OWL-S files in order to be able to send updated stage for user to view. The user's task is to select the parts of service description such as top level ontologies of OWL-S, define the service type and fill out given description forms according the development stage.

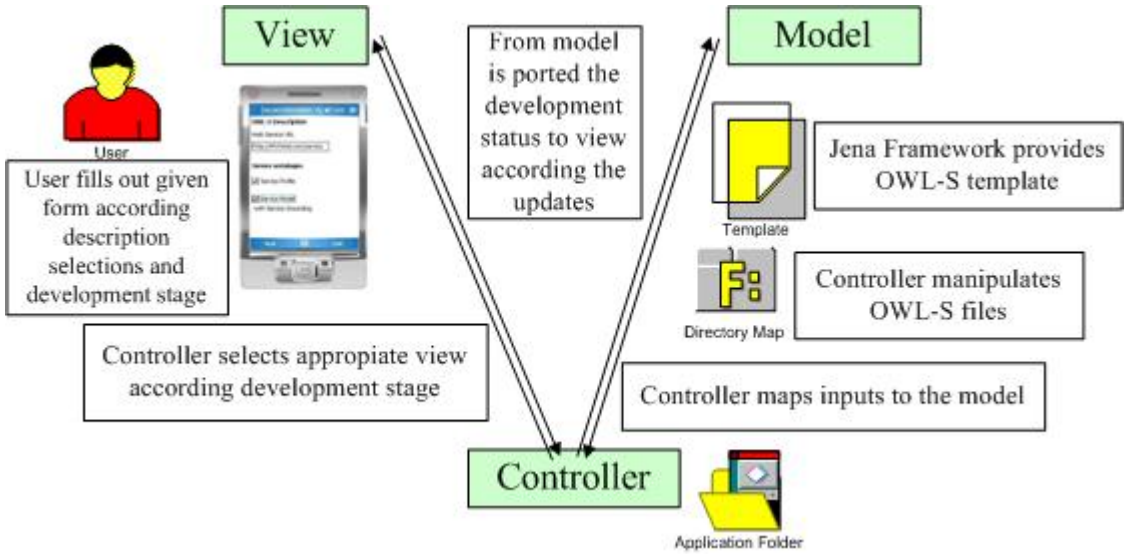

Figure 15. Mobile Semantic Assistant based on MVC design pattern.

The overall Mobile Semantic Assistant framework structure and its requirements are showed in Figure 16. As described above, the view is the user interface including user inputs of Profile, Model and Grounding description. The Mobile Semantic Assistant core handles the description generation and user input transmission to the model. Specifically the information transmission is centralized on description templates. The description templates are provided by Jena Framework. As the sub-ontology bases are configured and Jena Framework propose compact version meant for mobile device, it is here called Jena tool employed with OWL-S description setting as shown in Figure 16. The description templates are configured for each own sub-ontology. The Profile description template is classified into non-functional and functional description types. The non-functional description type includes in addition the classification description type. The functional description defined in Service Profile description in view, has straight relation

to the Service Model and its process. The Service Model and its process selection purpose is to define which process or processes to include in the overall description and which to leave out if not required. As well as the Service Model purpose is to set up wanted process, the Service Grounding purpose is to determine which specification is used. Although the OWL-S 1.2 supports only WSDL 1.1, here is possibility to use also SAWSDL as there have been initials to support SAWSDL in grounding. Assuming that OWL-S is capable to utilize SAWSDL within grounding in the near future, here is given the possibility to use WSDL 1.1 or SAWSDL with WSDL 2.0 in grounding. These specifications require different mappings which are defined in the Grounding description template. The WSDL and SAWSDL mappings are configured in the core of Mobile Semantic Assistant. The WSDL file itself is generated in Web Service registry and therefore the Web Service framework on server end will execute the appropriate WSDL file. Specifically the client sends invoke command to Web Services framework which generates appropriate service result in WSDL.

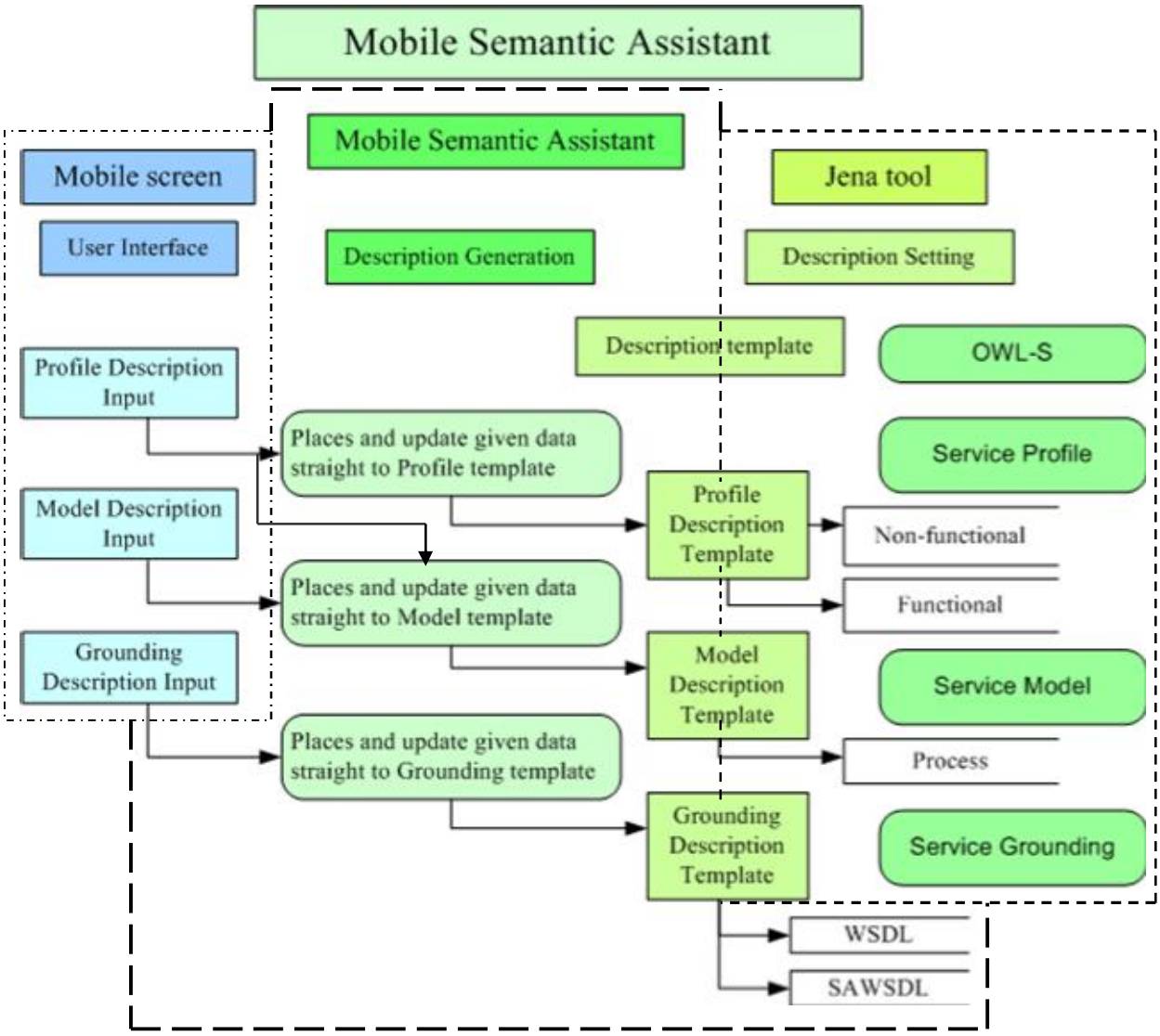

Figure 16. Mobile Semantic Assistant framework structure

As mentioned above, the aim is to use configured service templates assisting effortless creation of OWL-S description. To do this, the Mobile Semantic Assistant framework is required to provide a default base of OWL-S. The service description creation by OWL-S default base is shown in Figure 17 and 18. The OWL-S template for service is updated according the user input and specifically the sub-ontologies are generated or leaved out according the user selection. In Figure 17 the OWL-S service template presents top level ontology for the Service class. As the user selects sub-ontologies Profile, Model with Grounding, these are generated to present, describe by and support the top level ontology Service.

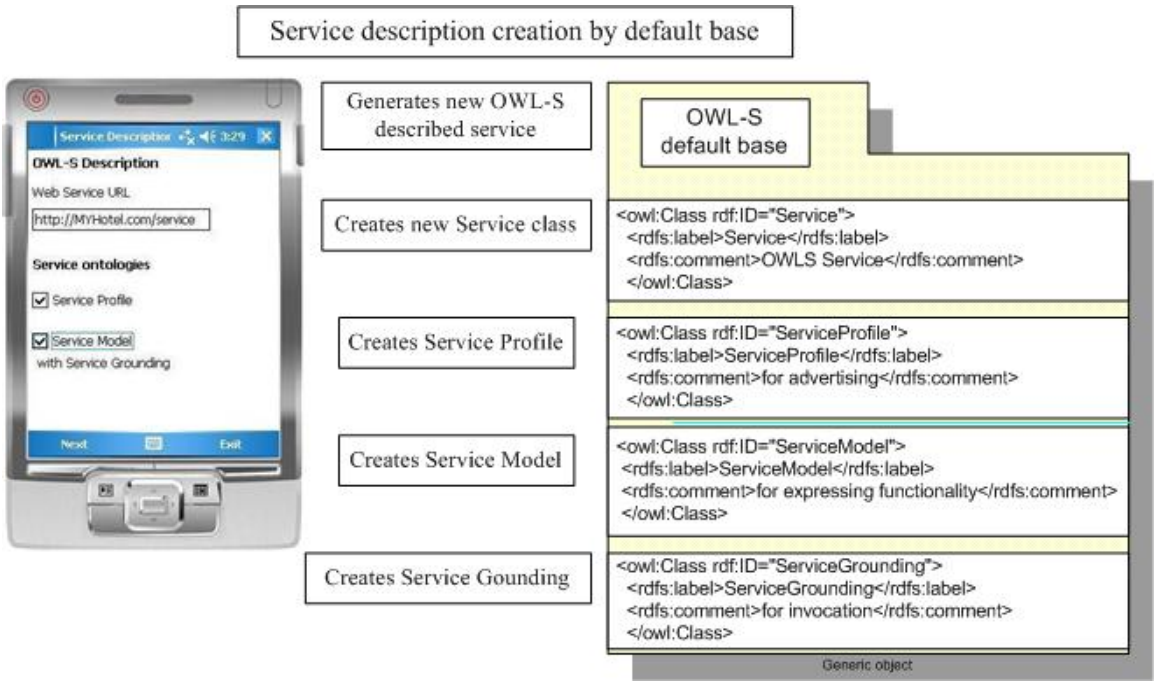

Figure 17. Service description creation by OWL-S default base

In the fist service description phase of new service generation the user gives the root of the resource name space to where top level ontology is located creating for each sub-ontology of OWL-S own resource location. For example, the user gives “myhotel.com/service” as default Web Service URI and the service description resource directories are generated according given URI as shown in Figure 18. The Jena tool includes defined classes and properties required to set up full OWL-S description. According to user input, the OWL-S template is update and resources are generated to own directories according given root of the resource name space as showed OWL-S service template highlighted with red colour as an example.

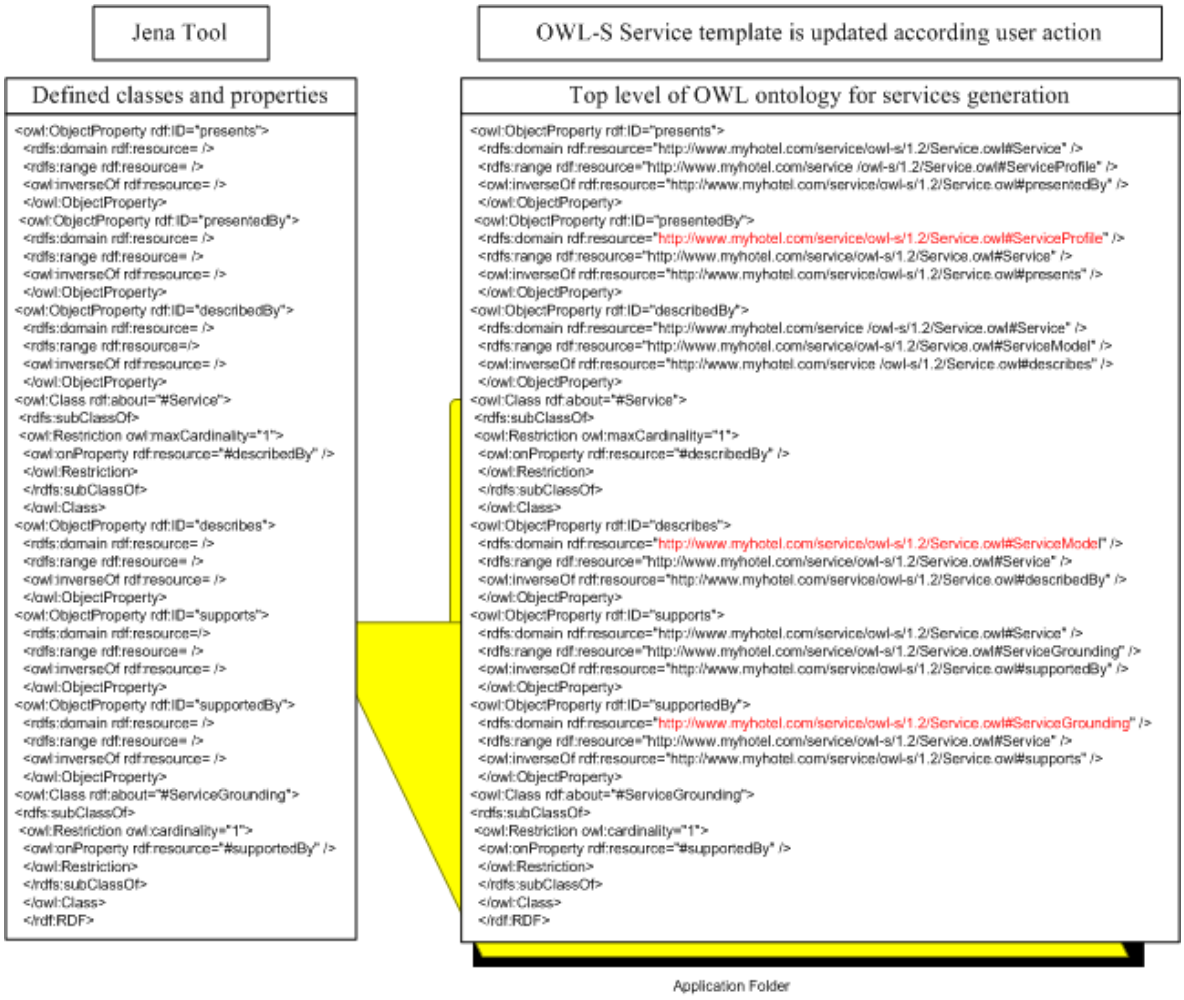

Figure 18. Service description creation by OWL-S default base

In the next section is discussed about implementation of OWL-S description base creation and OWL ontology for services configuration using Jena framework. The Jena framework is used for OWL-S description generation creating for each sub-ontology separate OWL files which are then later published in the Internet for other users' usage.

8.4 Implementation possibilities for mobile enabled service description creation

The Jena framework is used for OWL-S description setting in order to provide OWL-S base for description creation. The core idea is that the OWL-S description templates are updated by user inputs and OWL-S files of new OWL-S description is published for service discovery and consumption. In this section is explained how to create RDF/OWL

syntax in Jena framework with examples. The examples of RDF/OWL syntax generation are implemented using Eclipse programming environment. In order to be able to create the OWL files, the RDF is created first based on user input. Finally the OWL files are published to Web Service registry from where other user can discover and utilize these created services. Below is showed an example of creating RDF through Jena API. The required basic library packages are ontology, RDF and vocabulary as shown in Figure 19.

```
import com.hp.hpl.*;
import com.hp.hpl.jena.ontology.*;
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.util.FileManager;
import com.hp.hpl.jena.vocabulary.*;
```

Figure 19. Import basic library packages.

One important note is, that the Jena supports only predefined vocabulary and if the user infers the vocabulary, it is added in property and the result can be default Jena syntax (j.0). Therefore, a method *setNsPrefix* is given to create own vocabulary based properties as presented in Figure 20.

```
 // create an empty Model
            Model model = ModelFactory.createDefaultModel();
// setting own prefix
            model.setNsPrefix("ServiceProfile", serviceuri);
```

Figure 20. setNsPrefix method for own vocabulary based properties creation.

In general it is assumed that the user has supported vocabulary in order to create the RDF with Jena for user's own service description. As the Jena includes only standard and predefined vocabulary, in order to accomplish wanted result, the properties are defined as shown in Figure 21 where the variables are taken from user inputs. For example, *serviceuri* variable is taken at first phase of service creation from user input when defining the URI for service. Other properties variables such as *servicename*, *servicetype* and *servicecomment* are got as well from user inputs.

```
Property servicename1 = model.createProperty(serviceuri + "servicename");
Property servicetype1 = model.createProperty(serviceuri + "servicetype");
Property servicecomment1 = model.createProperty(serviceuri + "servicecomment");
```

Figure 21. Defining the properties.

The next step is to create resource based on the provided property. As resources are created, RDF files can be generated. The Figure 22 presents resource creation.

```
Resource ServiceProfile = model.createResource(serviceuri+ "Resource")
            .addProperty(servicename1, servicename)
            .addProperty(servicetype1, servicetype)
            .addProperty(servicecomment1, comments);
```

Figure 22. Resource creation based on property.

Finally the RDF file can be created for example in test.owl which will be later called to handle the OWL functionally. The example of the resulting RDF file is showed below in Figure 23. The highlighted parts with red colour are got from user inputs. Accordingly the URI, service name, type and comments are questioned in the Profile Description input (see Fig. 16).

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ServiceProfile="http://myhotel.com/reservationservice/rs#" >
  <rdf:Description rdf:about="http://myhotel.com/reservationservice/rs#Resource">
    <ServiceProfile:servicename>myhotel</ServiceProfile:servicename>
    <ServiceProfile:servicetype>hotel reservation</ServiceProfile:servicetype>
    <ServiceProfile:servicecomment>good and reasonable</ServiceProfile:servicecomment>
  </rdf:Description>
</rdf:RDF>
```

Figure 23. Generated RDF file according user input.

The RDF file is used in next step, calling it inside the OWL createFactory model, which will generate the OWL based resource on property provided in RDF. The creation of OWL ontology model with ModelFactory method is showed above of Figure 24. Below in Figure 25 is shown OWL result.

```
OntModel ontl = ModelFactory.createOntologyModel();

//create the pathway ontology
          Ontology ont = ontl.createOntology(namefilepath);
//import the level ontology into the pathway
           ont.addImport(ontl.createResource(namefilepath1));
```

Figure 24. ModelFactory method for OWL model creation.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://myhotel.com/reservationservice/rs#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <rdf:Description rdf:about="http://myhotel.com/reservationservice/rs
/test.owl">
    <owl:imports rdf:resource="http://myhotel.com/reservationservice/rs
/test.owl"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Ontology"/>
  </rdf:Description>
</rdf:RDF>
```

Figure 25. The OWL file resource

Previous examples show the generation of OWL files. Consequently, one OWL file for each of sub-ontology of OWL-S is generated. As all wanted sub-ontologies are successfully generated, eventually it is assumed that user is connected to Wi-Fi or GPRS service through mobile phone to utilize the Internet where Mobile Semantic Assistant publishes the created OWL-S service with invocation command to generate WSDL 1.1 or 2.0. For example OWL-S Editors core idea is to generate OWL-S description from a WSDL file, whereas Mobile Semantic Assistant core idea is to generate OWL-S files according user inputs and action having command line for WSDL 1.1 or WSDL 2.0 generation inside Mobile Semantic Assistant functionality centre. Above described method of service description creation supported by Jena framework requires that properties and variables are defined in order to be able utilize user inputs for RDF result and final OWL result. Since Jena ontology support is limited to build on top of RDF, the structure is generated first in RDF as in above described method. Inside the Jena can be

used SPARQL for RDF data search as SPARQL queries can be created and executed using Jena's *QueryFactory*.

Considering mobile devices limited memory space and processing capacity, the µJena in reasonable option for Mobile Semantic Assistant. However, for now the µJena supports only N-triples format. In order to employ µJena in Mobile Semantic Assistant, µJena would need conversion to support RDF files creation. Another implementation possibility is to define classes and properties in pure OWL avoiding the RDF files generation. However, in order to be able define classes and properties in pure OWL, the Jena would need extensional OWL vocabulary definition. This could be possible by building extensional OWL vocabulary in the source code since Jena is open source. This would require full OWL hierarchy classification and extensional OWL vocabulary creation. The extensional OWL vocabulary would be possible to create by supporting OWL-S creation with Mobile Semantic Assistant. The OWL-S default base in Mobile Semantic Assistant would be equivalent to extensional OWL vocabulary in Jena. The required OWL classes, properties and variables would be found in Jena vocabulary for the OWL-S default base. In addition, as µJena is a compact version of fundamental RDF platform Jena framework, same way as µJena has been created supporting N-triples, as well for OWL ontology creation would be possible to implement compact version of Jena with OWL vocabulary for OWL-S default base. The Jena package for OWL-S default base could contain for example the library of OWL vocabulary, the source code and javadoc for this minuscule Jena version.

In this section described Jena framework usage methods are implementation possibilities for mobile enabled service description creation. The Jena framework provides feasible ground for OWL-S files creation and management for required OWL-S default base for Mobile Semantic Assistant. Considering future implementation options, the µJena is potential example of compact Jena framework providing certain libraries with certain information format. The proposition for mobile Semantic Assistant implementation is to create micro format and fast OWL supported Jena minuscule version.

9. CONCLUSIONS

The main goal of this thesis was to draw out the importance of role and involvement in the Semantic Web scenery and future Internet of Services evolution and emphasize end users involvement in Semantic Web Service description creation. The vision of future Internet of Services is to empower users' collaboration in content and service creation, composition and sharing. Many projects working on realizing the vision of Future Internet are dedicated on device and context aware technologies enabling users with different level of expertise to search, use, compose, configure and share services. These projects such as FAST and m:Ciudad researches are focused on their own developed platforms architectures and methods in order to enable users to customise and control services according to their actual needs and as well provide semantics aware services.

The Web 2.0 has risen up the users' collaboration in content and service creation and sharing. The user generated content (UGC) has been the resulting trend emerged with Web 2.0 phenomenon. Inherently the user generated services (UGS) is emerging trend in the future Internet of Services vision. As in UGC domain the organization and categorization of content is done by user generated tags and bookmarking, in the services domain to description creation is done merely by professional and experts because of complexity of Semantic Web Services descriptions. Categorization and browsing based on tags work for contents as considering discovery process, but discovery issues become more challenging with services when trying to find services and require manual work of the users in determination of whether does the service satisfy the needs or not.

The Semantic Web aims to bring intelligence to Web Services with semantic descriptions enabling automatic service discovery, interoperation, composition and invocation. With help of ontologically annotated Web Services, intelligent agents are able to reason about their content enabling machine-interpretable semantics and automating information management tasks. The comparison of Semantic Web Services description creation conclude that OWL-S features support sufficiently the required information management tasks such as service discovery, interoperation, composition and invocation. Moreover, OWL-S possesses well defined conceptual structure that is feasible to chop into pieces

and infer in order to apply for semantic description creation done by end user and employ on mobile device. In order to be able realize OWL-S description, in this thesis has been studied the existing tools for OWL-S description creation. Since existing development environments and tools for OWL-S description creation are destined to employ solely on desktop computer environment with prosperous possessing capabilities for semantic information management, these tools such as OWL-S Editor and SPEX tool are not applicable on mobile domain. Therefore resent initial efforts for semantic service descriptions employment on mobile computing field has been presented and studied possibilities for semantic information management handling done on mobile device.

Currently Web Services ontology creation is left on professional developers' responsibility and the ontology building environments and tools are designed almost merely to be used on desktop computing domain. Therefore in this thesis have been studied implementation possibilities for semantic description creation on mobile device designed to be used by non-expertise end user, such as prosumer or professional amateur interested to create service description of own Web Service that is discoverable and interoperable in machine-interpretable way for other users usage. In consequence of the study on semantic information handling on mobile domain, this thesis presents initial design of Mobile Semantic Assistant framework. The Mobile Semantic Assistant framework is aimed to facilitate service description creation in OWL-S for end user. The use case of Mobile Semantic Assistant and presentation of service description creation on mobile device indicate that with ready configured ontological base it is possible to create even structurally large service descriptions such as OWL-S is. The presentation of service description creation on mobile device shows also that from the user is not required excessive or complex settings by providing easy standard configuration settings, yet same time give possibility for user to set own specifications to description if needed. After study on programmatic environments for OWL, the Jena framework was selected with possible improvements for Mobile Semantic Assistant framework. The study can be expanded further by editing the Jena framework suitable for supporting ontological database and the service description creation on mobile device. The recommendation for the future is to enable mobile user generated Semantic Web Service descriptions in order to realize vision of IoS and Semantic Web.

REFERENCES

[1] Hepp M. Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies. Internet Computing, IEEE , vol.11, no.1, page(s):90-96, Jan.-Feb. 2007. ISSN: 1089-7801

[2] Simple Mobile Services. [www-document] [Retrieved: October 12, 2009]. Available at: http://www.ist-sms.org/

[3] Open Platform for User-Centric Creation and Execution (OPUCE). [www-document] [Retrieved: October 14, 2009]. Available at: http://www.opuce.tid.es/

[4] Davies M.; Gil G.; Maknavicius L.; Narganes M.; Urdiales D.; Zhdanova A.V. m:Ciudad: An Infrastructure for Creation and Sharing of End User Generated Microservices. [pdf document] [Retrieved: October 14, 2009]. Available at: http://www.mciudad-fp7.org/pdf/mCiudad-FIS2008.pdf

[5] SPICE consortium. IST-FP6 project SPICE. [www-document] [Retrieved: October 12, 2009]. Available at: http://www.ist-spice.org/

[6] Schroth, C.; Janner, T. Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services. IEEE IT Professional Vol.9, No.3, page(s):36-41, June 2007. ISSN: 1520-9202

[7] Networked European Software & Services Initiative. Next Generation Services Front Ends in the Future Internet of Services: Research Challenges and Opportunities. [pdf document] [Retrieved: October 13, 2009]. Available at: http://ec.europa.eu/information_society/events/cf/ict2008/document.cfm?doc_id=8467

[8] Developing the Future of the Internet through European Research]European Commission. Future Networks & Services. Developing the Future of the Internet through European Research. [pdf document] [Retrieved: October 12, 2009]. Available at: http://www.ifap.ru/library/book325.pdf

[9] Hagemann S.; Vossen G. Categorizing User-Generated Content. [pdf document] [Retrieved: October 20, 2009]. Available at: http://journal.webscience.org/155/2/websci09_submission_57.pdf

[10] Seshadri S.C. Realizing the Potential of User-Generated Content and Social Networking. [pdf document] [Retrieved: October 11, 2009]. Available at: http://www.infosys.com/offerings/industries/communication-services/white-papers/Documents/realizing-potential-user-generated.pdf

[11] Ubiquitous Internet research group. [www-document] [Retrieved: October 12, 2009]. Available at: http://cnd.iit.cnr.it/

[12] O'Reilly, Tim. 2005. What is Web 2.0. [www-document] [Retrieved: November 02, 2009]. Available at: http://oreilly.com/web2/archive/what-is-web-20.html

[13] Boari M.; Corradi A.; Lodolo E.; Monti S.; Pasini S. Coordination for the internet of services: A user-centric approach. COMSWARE 2008. page(s):434 – 441, 2008. Print ISBN: 978-1-4244-1796-4

[14] Hierro J.; Oliphant A. User generated services and user generated content: Similarities and Differences. [pdf document] [Retrieved: October 10, 2009]. Available at: http://services.future-internet.eu/images/2/22/Report-UGC-Panel-FISO-FIA-Madrid.pdf

[15] Högg R.; Meckel M.; Stanoevska-Slabeva K.; Martignoni R. Overview of business models for Web 2.0 communities. Proceedings of GeNeMe, 2006, Page(s): 23-37 [pdf document] [Retrieved: October 10, 2009]. Available at: http://www.alexandria.unisg.ch/publications/31411

[16] Šnuderl K. Tagging: Can User-Generated Content Improve Our Services? [pdf document] [Retrieved: October 20, 2009]. Available at: http://unstats.un.org/unsd/statcom/statcom_09/seminars/innovation/Innovation%20Seminar/Slovenia-Tagging.pdf

[17] Baladron C.; Aguiar J.; Carro B.; Sanchez-Esguevillas A. Integrating User-Generated Content and Pervasive Communications. Pervasive Computing, IEEE, page(s): 58-61, 2008. ISSN: 1536-1268

[18] Davies Marcin. Towards a Semantic Infrastructure for User Generated Mobile Services. The Semantic Web: Research and Applications, Springer Berlin / Heidelberg, page(s): 924-928, 2009. ISBN: 978-3-642-02120-6

[19] Future Perspectives in End-User Development. Klann M.; Paternò F.; Wulf V. Future Perspectives in End-User Development. End User Development, Spinger, page(s): 475-486. 2006. ISBN: 978-1-4020-5386-3

[20] Lieberman H.; Paternò F.; Wulf V. End User Development: An Emerging Paradigm. End User Development, Spinger, page(s): 1-8. 2006. ISBN: 978-1-4020-5386-3

[21] Fisher G. End-User Development and Meta-design: Foundations for Cultures of Participation. End-User Development, Springer Berlin / Heidelberg, page(s): 3-14, 2009. ISSN: 1611-3349

[22] Benkler Y. The Wealth of Networks: How Social Production Transforms Markets and Freedom. New Haven, Conn: Yale University Press, pages 515, 2006. ISBN 0-300-11056-1

[23] Shin, Y.; Yu, C.; Chung, S.; Kim, S. End-User Driven Service Creation for Converged Service of Telecom and Internet. Fourth Advanced International Conference on Telecommunications (AICT), page(s): 71–76, June 2008. Print ISBN: 978-0-7695-3162-5

[24] Erl Thomas. Service-Oriented Architecture Concept Technology and Design. Prentice Hall, pages 792, 2005. ISBN-10: 0-13-185858-0

[25] Lizcano D.; Soriano J., Reyes M.; Hierro J.J. EzWeb/FAST: Reporting on a Successful Mashup-Based Solution for Developing and Deploying Composite Applications in the Upcoming "Ubiquitous SOA". Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM '08, page(s): 488 – 495, 2008. ISBN: 978-0-7695-3367-4

[26] Nestler T.; Dannecker L.; Pursche A. User-centric Composition of Service Front-ends at the Presentation Layer. [pdf document] [Retrieved: November 26, 2009]. Available at: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-540/ugs2009_submission_2.pdf

[27] ICT-Information and communication Technologies. [www-document] [Retrieved: November 06, 2009]. Available at: http://cordis.europa.eu/fp7/ict/

[28] Urdiales D.; de las Heras R.; Narganes M.; Davies D.; Zhdanova A.V.; Christophe B.; Maknavicius L.; Immonen M.; Heinilä J.; Barnaghi P. m:Ciudad – Unleashing mobile user-provided services. In Proceedings of the W3C Track @ WWW2009 conference, 23-24 April 2009, Madrid, Spain.

[29] Davies M.; Gil G.; Maknavicius L.; Narganes M.; Urdiales D.; Zhdanova A. V. m:Ciudad: An Infrastructure for Creation and Sharing of End User Generated Microservices. In Proceedings of the Poster and Demonstration Track at the 1st Future Internet Symposium, CEUR Workshop Proceedings, Vol. 399, September 2008, Vienna, Austria. ISSN: 1613-0073

[30] Droegehorn, O.; Sian Lun Lau; Klein, N.; Koenig, I.; Porras, J. Service Creation for End-Users. Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on 15-18 Sept. page(s):1 – 5, 2008. Print ISBN: 978-1-4244-2643-0

[31] Domingue, J., Fensel D., González-Cabero, R. SOA4All, Enabling the SOA Revolution on a World Wide Scale. In Proceedings of the 2nd IEEE International Conference on Semantic Computing (ICSC 2008), Santa Clara, CA, USA, August, page(s): 530 – 537, 2008. ISBN: 978-0-7695-3279-0

[32] Krummenacher R.; Norton B.; Simperl E.; Pedrinaci C. SOA4All: Enabling Web-scale Service Economies. IEEE Computer Society, 3rd IEEE International Conference on Semantic Computing (ICSC), page(s): 535 – 542, 2009. ISBN: 978-1-4244-4962-0

[33] NESSI: Networked European Software and Services Initiative. [www-document] [Retrieved: October 14, 2009]. Available at: http://www.nessi-europe.com/Aboutus/AnintroductiontoNESSI/Presentation/tabid/152/Default.aspx

[34] Service Web 3.0 [www-document] [Retrieved: October 14, 2009]. Available at: http://www.serviceweb30.eu/cms/

[35] Alonso G.; Casati F.; Kuno H.; Machiraju V. Web services: Concepts, Architectures and Applications. Springer Verlag, page(s): 354, 2004. ISBN: 3-540-44008-9

[36] Sheth A.; Nagarajan M. Semantics-Empowered Social Computing. IEEE Internet Computing, Vol. 13, Issue 1, page(s): 76-80, 2009. ISSN: 1089-7801

[37] Jorge Cardoso. On the Move to Semantic Web Services. Proceedings of World Academy of Science, Engineering and Technology, Vol. 8, October 2005. ISSN: 1307-6884

[38] Handschuh Siegfried. Semantic Annotation of Resources in the Semantic Web. Semantic Web Services: Concepts, Technologies, and Applications. Springer Berlin Heidelberg, page(s): 135-155, 2007. ISBN: 978-3-540-70893-3

[39] Dragan Gasevic, Dragan Djuric, Vladan Devedzic. Model Driven Architecture and Ontology Development. Springer Berlin Heidelberg New York, page(s): 311, 2006. ISBN: 3-540-32180-2

[40] Erl, Thomas. Service-oriented architecture: Concepts, Technology, and Design. Prentice Hall, page(s): 792, 2005. ISBN: 0-13-185858-0

[41] Aier, S; Offermann, P.; Schönherr, M.; Schröpfer, C. Implementing Non-functional Service Descriptions in SOAs. Trends in Enterprise Application Architecture. Springer LNCS, Berlin, page(s): 40-53, 2007. ISBN: 978-3-540-75911-9

[42] Berners-Lee T.; Hendler J. and Lassila O. The Semantic Web. Scientific American, Vol. 284, Nr. 5, page(s): 29-37, 2001. ISSN: 0036-8733

[43] Klusch Matthias. Semantic Web Service Description. CASCOM: Intelligent Service Coordination in the Semantic Web. Springer-Verlag, page(s): 31-57 2008. ISBN: 978-3-7643-8574-3

[44] Ariel Pashtan. Mobile Web Services. New York: Cambridge University Press, page(s): 284, 2005. ISBN: 0-521-83049-4

[45] Lausen H.; Lara R; Polleres A.; Bruijn.; Roman D. Discovery. Semantic Web Services: Concepts, Technologies, and Applications. Springer Berlin Heidelberg, page(s): 211-244, 2007. ISBN: 978-3-540-70893-3

[46] Michael P. Papazoglou. WEB services: principles and technology. Pearson-Prentice Hall, page(s): 782, 2008. ISBN: 978-0-321-15555-9

[47] Kashyap V.; Bussler C.; Moran, M. The Semantic Web: Semantics for Data and Services on the Web. Springer-Verlag Berlin Heidelberg, page(s): 79-135, 2008. ISBN: 978-3-540-76452-6

[48] Bijan Parsia. Querying the Web with SPARQL. Springer Berlin-Heidelberg, page(s): 53-67, 2006. ISSN: 1611-3349

[49] Web Ontology Language (OWL). Ivan Herman. W3C. [www-document] [Retrieved: January 2, 2010]. Available at: http://www.w3.org/2004/OWL/

[50] DARPA Agent Markup language (DAML) [www-document] [Retrieved: January 2, 2010]. Available at: http://www.daml.org/

[51] SPARQL Query Language for RDF. W3C Recommendation 15 January 2008. [www-document] [Retrieved: January 2, 2010]. Available at: http://www.w3.org/TR/rdf-sparql-query/

[52] Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001 [www-document] [Retrieved: January 13, 2010]. Available at: http://www.w3.org/TR/wsdl

[53] OASIS . UDDI Version 3.0.2. UDDI Spec Technical Committee Draft, Dated 20041019. [www-document] [Retrieved: January 11, 2010]. Available at: http://www.uddi.org/pubs/uddi_v3.htm

[54] SOAP Version 1.2. W3C Recommendation (Second Edition) 27 April 2007. [www-document] [Retrieved: January 13, 2010]. Available at: http://www.w3.org/TR/soap/

[55] Patil A.; Oundhakar S.; Sheth A.; Verma K. METEOR-S Web service Annotation Framework. In 13th conference on World Wide Web. ACM Press 2004. ISBN: 1-58113-844-X

[56] Web Service Semantics –WSDL-S, Version 1.0, W3C Member Submission, Akkiraju, R. et al, November 2005 [www-document] [Retrieved: January 2, 2010]. Available at: http://www.w3.org/Submission/WSDL-S.

[57] LSDIS. METEOR-S: Semantic Web Services and Processes. [www-document] [Retrieved: January 16, 2010]. Available at: http://lsdis.cs.uga.edu/projects/meteor-s/

[58] Larvet, P.; Christophe, B.; Pastor, A. Semantization of Legacy Web Services: From WSDL to SAWSDL. Third International Conference. Internet and Web Applications and Services, ICIW '08, page(s): 130 – 135, 2008. ISBN: 978-1-60558-990-9

[59] Ferndriger S. et al. Enhancing Semantic Web Services with Inheritance. Lecture Notes in Computer Science, Vol. 5318. Proceedings of the 7th International Conference on the Semantic Web. Springer-Verlag, page(s): 162 – 177, 2008. ISBN: 978-3-540-88563-4

[60] Sheth, A.P.; Staab, S.; Dean, M.; Paolucci, M.; Maynard, D.; Finin, T.; Thirunarayan, K. The Semantic Web. Semantic Web Standards. The Semantic Web - ISWC 2008. 7th International Semantic Web Conference, ISWC 2008, Vol. 5318, Springer, page(s): 935, 2008. ISBN: 978-3-540-88563-4

[61] Fensel D.; Lausen H.; Bruijn J.; Stollberg M.; Roman D.; Polleres A.; Domingue J. Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer-Verlag Berlin Heidelberg, page(s): 63-81 2007. ISBN: 978-3-540-34519-0.

[62] European Semantic Systems Initiative. ESSI Cluster. [www-document] [Retrieved: January 13, 2010]. Available at: http://www.essi-cluster.org/news/press-releases/

[63] He, H.; Haas, H.; Orchard, D. Web Services Glossary. Web Services Architecture Usage Scenarios, Technical report, W3C, 2004. [www-document] [Retrieved: January 17, 2010]. Available at: http://www.w3.org/TR/ws-gloss/

[64] Corcho O.; Losada S.; Benjamins R. Mediation. Semantic Web Services: Concepts, Technologies, and Applications. Springer Berlin Heidelberg, page(s): 287-308, 2007. ISBN: 978-3-540-70893-3

[65] Nagarajan, M.; Verma, K.; Sheth, A.P.; Miller, J.; Lathem, J. Semantic Interoperability of Web Services – Challenges and Experiences. Web Services, ICWS '06. page(s): 373 – 382, 2006. ISBN: 0-7695-2669-1

[66] Henocque L.; Kleiner M Semantic Web Services: Concepts, Technologies, and Applications. Springer Berlin Heidelberg, page(s): 245-286, 2007. ISBN: 978-3-540-70893-3

[67] Cabral L.; Domingue J.; Motta E.; Payne T.; Hakimpour F. Approaches to Semantic Web Services: An Overview and Comparisons. The Semantic Web: Research and Applications. First European Semantic Web Symposium, ESWS 2004. Springer Berlin Heidelberg, page(s): 225-239, 2004. ISBN: 978-3-540-21999-6

[68] Cardoso, J. and Sheth Amit P. Semantic Web Processes: Semantics Enabled Annotation, Discovery, Composition and Orchestration of Web Scale Processes. In Fourth International Conference on Web Information Systems Engineering. WISE 2003 page(s): 375, 2003. ISBN: 0-7695-1999-7

[69] Cardoso, J. and Sheth Amit P. Semantic Web Services, Processes and Applications. Springer Science-Business Media, page(s): 161-193, 2006. ISBN: 978-0-387-34685-4

[70] D30v0.1 Aligning WSMO and WSDL-S. WSMO Working Draft 5 August 2005 [www-document] [Retrieved: January 28, 2010]. Available at: http://www.wsmo.org/TR/d30/v0.1/

[71] Dieter Fensel, Mick Kerrigan, Michal Zaremba. Implementing Semantic Web Services: The SESA Framework. Springer-Verlag Berlin Heidelberg, page(s): 285-301, 2008. ISBN: 978-3-540-77019-0

[72] SAWSDL Recommendation. Semantic Annotations for WSDL Working Group [www-document] [Retrieved: February 4, 2010]. Available at: http://www.w3.org/2002/ws/sawsdl/

[73] OWL-S: Semantic Markup for Web Services. W3C Member Submission 22 November 2004 [www-document] [Retrieved: February 4, 2010]. Available at: http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

[74] David Martin et al. Bringing Semantics to Web Services: The OWL-S Approach. Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, California, USA, July 6-9, (2004). Springer-Verlag, page(s): 26-42, 2005. ISSN: 0302-9743

[75] Fensel D.; Lausen H.; Bruijn J.; Stollberg M.; Roman D.; Polleres A.; Domingue J. Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer-Verlag Berlin Heidelberg, page(s): 26-42, 2007. ISBN: 978-3-540-34519-0

[76] Semantic Web Services Language (SWSL). W3C Member Submission 9 September 2005. [www-document] [Retrieved: January 17, 2010]. Available at: http://www.w3.org/Submission/SWSF-SWSL/

[77] Richard Hull. Towards a Unified Model for Web Services Composition. Advances in Computer Science – ASIAN 2005. Springer Berlin-Heidelberg 2005. ISBN: 978-3-540-30767-9

[78] Dieter Fensel, Mick Kerrigan, Michal Zaremba. WSMO and WSML. Implementing Semantic Web Services: The SESA Framework. Springer-Verlag Berlin Heidelberg, page(s): 43-65, 2008. ISBN: 978-3-540-77020-6

[79] D14v0.2. Ontology-based Choreography and Orchestration of WSMO Services. WSMO Final Draft 3rd February 2006 [www-document] [Retrieved: January 2, 2010]. Available at: http://www.wsmo.org/TR/d14/v0.2/

[80] D4.2v01 Formal Comparison WSMO/OWL-S. DERI Working Draft 15 March 2004 [pdf document] [Retrieved: January 28, 2010]. Available at: http://www.wsmo.org/papers/deliverables/d4.2v01_20040315.pdf

[81] Lara R.; Roman D.; Polleres A.; Fensel D.; A Conceptual Comparison between WSMO and OWL-S. Web Services. European Conference, ECOWS 2004, September 27-30, 2004. Springer Berlin / Heidelberg, 2004. ISBN: 978-3-540-23202-5

[82] Michael Kifer. Service Discovery with SWSL-Rules. Position paper for the W3C Workshop on Frameworks for Semantics in Web Services, June 2005. [pdf document] [Retrieved: January 27, 2010]. Available at: http://www.w3.org/2005/04/FSWS/Submissions/62/kifer.pdf

[83] Semantic Web Services Framework (SWSF) Overview. W3C Member Submission 9 September 2005. [www-document] [Retrieved: January 18, 2010]. Available at: http://www.w3.org/Submission/SWSF/

[84] Urbieta A.; Azketa E.; Gomez I.; Parra J.; Arana N. Bridging the Gap between Services and Context in Ubiquitous Computing Environments Using an Effect- and Condition-Based Model. 3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008. Springer Berlin / Heidelberg, page(s): 43-65, 2008. ISBN: 978-3-540-85866-9

[85] Martin D.; Burstein M.; McDermott D.; McIlraith S.; Paolucci M.; Sycara K.; McGuinness D.; Sirin E.; Srinivasan N. Bringing Semantics to Web Services with OWL-S. World Wide Web Journal, Vol. 10, No. 3, Springer Netherlands, 2007. ISSN: 1386-145X

[86] Liyang Yu. Introduction to the Semantic Web and Semantic Web Services. Chapman & Hall/CRC, 2007. ISBN: 978-1-58488-933-5

[87] David Martin. OWL-S: Semantic Markup for Web Services. [www-document] [Retrieved: February 4, 2010]. Available at: http://www.ai.sri.com/daml/services/owl-s/1.2/overview/

[88] OWL-S Walk-Through [www-document] [Retrieved: February 6, 2010]. Available at: http://www.ai.sri.com/daml/services/owl-s/1.2/OWL-S-walkthru.html

[89] Congo.com fictitious B2C site [www-document] [Retrieved: February 7, 2010]. Available at: http://www.ai.sri.com/daml/services/owl-s/1.2/CongoService.owl

[90] Martin D.; Paolucci M.; Wagner M. Bringing Semantic Annotations to Web Services: OWL-S from the SAWSDL Perspective. The Semantic Web. 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007. Springer Berlin / Heidelberg, page(s): 340-352, 2007. ISBN: 978-3-540-76297-3

[91] Martin D.; Paolucci M.; Wagner M. Grounding OWL-S in SAWSDL. Service-Oriented Computing – ICSOC 2007, Springer Berlin / Heidelberg, page(s): 416-421, 2007. ISBN: 978-3-540-74973-8

[92] Kopecky, J.; Vitvar, T.; Bournez, C.; Farrell, J. SAWSDL: Semantic Annotations for WSDL and XML Schema. Internet Computing, IEEE, Vol. 11, Issue: 6, page(s): 60 – 67, 2007. ISSN: 1089-7801.

[93] Iqbal, K.; Sbodio, M.L.; Peristeras, V.; Giuliani, G. Semantic Service Discovery using SAWSDL and SPARQL. Semantics, Knowledge and Grid, 2008. SKG '08. Fourth International Conference, page(s): 205 – 212, 2008. ISBN: 978-0-7695-3401-5

[94] LSDIS. SAWSDL: Semantic Annotations for WSDL [www-document] [Retrieved: March 22, 2010]. Available at: http://lsdis.cs.uga.edu/projects/meteor-s/SAWSDL/

[95] Kopecky, J.; Vitvar, T.; Gomadam, K. 2008. d38v0.1 MicroWSMO: semantic description of Restful services. Deliverable, Conceptual Models for Services Working Group. WSMO Working Draft – 19th February 2008 [pdf document] [Retrieved: March 22, 2010]. Available at: http://wsmo.org/TR/d38/v0.1/20080219/d38v01_20080219.pdf

[96] Dave Lambert and Neil Benn. WP3: Standardisation, Networking, and Community Building. D3.1 Standardisation Activity Report. January 18, 2010 [pdf document] [Retrieved: March 23, 2010]. Available at: http:// www.serviceweb30.eu/cms/index.php/resources/doc_download/100-d31-standardization-activity-report-m24

[97] T. Vitvar, J. Kopecky, M. Zaremba, and D. Fensel. WSMOLite: Lightweight Semantic Descriptions for Services on the Web. In ECOWS ’07: Proceedings of the Fifth European Conference on Web Services, 2007. IEEE Computer Society. ISBN: 978-0-7695-3044-4

[98] Stojanovic L. and Motik B. Ontology evolution within ontology editor. OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management EKAW 2002: 30th September 2002; Siguenza, Spain 2002, page(s): 53-62

[99] Gennari J.; Musen M.; Fergerson R.; Grosso W.; Crubézy M.; Eriksson H.; Noy N.; Tu S. The evolution of Protégé-2000: An environment for knowledge-based systems development. International Journal of Human-Computer Studies, 58(1):89–123, 2003. ISSN: 1071-5819

[100] Knublauch, H.; Fergerson, R.W.; Noy, N.F., et al. The Protégé OWL plugin: An open development environment for semantic web applications. Proceedings of the 3rd International Semantic Web Conference (ISWC), Hiroshima, Japan, November 2004 ISBN: 978-3-540-23798-3

[101] Jena Ontology API [www-document] [Retrieved: March 27, 2010]. Available at: http://jena.sourceforge.net/ontology/

[102] Jena: A Semantic Web Framework For Java. [www-document] [Retrieved: March 27, 2010]. Available at: http://jena.sourceforge.net/

[103] Srinivasan N.; Paolucci M.; Sycara K. CODE: A Development Environment for OWL-S Web services. Technical Report CMU-RI-TR-05-48, Robotics Institute, Carnegie Mellon University, October 2005. [www-document] [Retrieved: March 22, 2010]. Available at: http://www.ri.cmu.edu/pubs/pub_5159.html

[104] Htoo, N.Z.C.; Nyunt, T.T.S. Semantic Web Services offer discovery using OWL-S IDE. Signal Processing and Communication Systems, 2008. ICSPCS 2008, page(s): 1 – 5, 2008. ISBN: 978-1-4244-4243-0

[105] Kalyanpur A.; Parsia B.; Sirin E.; Cuenca Grau B.; Hendler J. Swoop: A Web Ontology Editing Browser. Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 4, Issue 2, page(s): 144-153, 2006. ISSN: 1570-8268

[106] Brewster C. et al. User-Centred Ontology Learning for Knowledge Management. Lecture Notes In Computer Science, Vol. 2553, Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems-Revised Papers, Springer-Verlag, page(s): 203 – 207, 2002. ISBN: 3-540-00307-X

[107] Daniel Elenius. The OWL-S Editor – A Domain -Specific Extension to Protégé. [pdf document] [Retrieved: March 28, 2010]. Available at: http://protege.stanford.edu/conference/2005/submissions/abstracts/accepted-abstract-elenius.pdf

[108] Scicluna, J., Abela, C., Montebello, M.: Visual modelling of OWL-S services. In: Proceedings of the IADIS WWW/Internet 2004 Conference (ICWI). Madrid, Spain, October 2004

[109] The Mindswap OWL-S API [www-document] [Retrieved: March 25, 2010]. Available at: http://www.mindswap.org/2004/owl-s/api/

[110] Timm, J.T.E.; Gannod, G.C. Grounding and Execution of OWL-S Based Semantic Web Services. International Conference on Services Computing, 2008. SCC '08. IEEE, Volume 2, page(s): 588 - 592

[111] Il-Woong Kim; Kyong-Ho Lee. Describing Semantic Web Services: From UML to OWL-S. Web Services, 2007. ICWS 2007. IEEE. Page(s): 529 – 536 ISBN: 0-7695-2924-0

[112] Amanchi, Santhosh K.; Durbha, Surya S.; King, Roger L.; Bheemireddy, Shruthi; Younan, Nicolas H.; Mobile computing and Sensor Web Services for coastal buoys. Geoscience and Remote Sensing Symposium, 2009 IEEE International,IGARSS, Vol. 5, page(s): V-465 - V-468, 2009. ISBN: 978-1-4244-3394-0

[113] Matthias Wagner, Massimo Paolucci. Enabling Personal Mobile Applications through Semantic Web Services. Future Networking Lab DoCoMo Communications Laboratories Europe, Munich, Germany [pdf document] [Retrieved: April 2, 2010]. Available at: http://www.w3.org/2005/04/FSWS/Submissions/53/DoCoMo-Pos-Paper.pdf

[114] Mostefaoui, G.K.; Pasquier-Rocha, J.; Brezillon, P. Context-Aware Computing: A Guide for the Pervasive Computing Community. International Conference on Pervasive Services, 2004. ICPS 2004. IEEE/ACS. page(s): 39 – 48, 2004. ISBN: 0-7803-8577-2

[115] Yu P; Cao J.; Wen W.; Lu J. Mobile Agent Enabled Application Mobility for Pervasive Computing. Third International Conference, UIC 2006, Lecture Notes in Computer Science, Ubiquitous Intelligence and Computing, page(s): 648-657, 2006. ISBN: 978-3-540-38091-7

[116] Broll G.; Siorpaes S.; Paolucci M.; Rukzio E.; Hamard J.; Wagner M.; Schmidt A. Supporting Mobile Service Interaction through Semantic Service Description Annotation and Automatic Interface Generation. Semantic Desktop and Social Semantic Collaboration Workshop, 5th International Semantic Web Conference ISWC 2006, Athens, GA, USA, November 5, 2006. ISSN: 1613-0073

[117] Baousis V.; Zavitsanos E.; Spiliopoulos V.; Hadjiefthymiades S.; Merakos L.; Veronis G. Wireless Web Services using Mobile Agents and Ontologies. In Proceedings of the IEEE International Conference of Pervasive Services (ICPS), Santorini, Greece, July, 2005. ISBN: 1-4244-0237-9

[118] Srinivasan, M.; Paolucci; Sycara K. Adding OWL-S to UDDI, Implementation and Throughput. First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004) San Diego, California, USA, 2004

[119] Verma, K.; Sivashanmugam, K.; Sheth A.; Patil A.; Oundhakar S.; Miller J. METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services. Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) Kluwer Academic Publishers pp. 17-39. ISSN: 1385-951X

[120] Serena Pastore. The necessity of semantic technologies in grid discovery. Journal of Networks, vol. 3, no. 4, April 2008 ISSN 1796-2056

[121] Klusch M.; Fries B.; Sycara K. OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services. Web Semantics: Science, Services and Agents on the World Wide Web, Volume 7, Issue 2, April 2009, Pages 121-133. ISSN: 1570-1263

[122] Bergenti F.; Caceres C.; Fernandez A.; Fröhlich N.; Helin H.; Keller O.; Kinnunen A.; Klusch M.; Laamanen H.; Lopes A.; Ossowski S.; Schuldt H.; Schumacher M. Context-aware Service Coordination for Mobile e-Health Applications. European Conference on EHealth (ECEH06), Switzerland, October 12-13, 2006. Proceedings of Lecture Notes in Informatics, Vol. P-91, GI-Edition, Germany. [pdf document] [Retrieved: April 2, 2010]. Available at: http://liawww.epfl.ch/~schumach/publications/Bergenti%20et%20al%20-%20Context-aware%20Service%20Coordination%20for%20Mobile%20e-Health%20Applications.pdf

[123] Broll G.; Siorpaes S.; Paolucci M.; Rukzio E.; Hamard J.; Wagner M.; Schmidt A. Supporting Mobile Service Usage through Physical Mobile Interaction. Pervasive Computing and Communications, 2007. PerCom '07. Fifth Annual IEEE International Conference, page(s): 262 – 271, 2007. ISBN: 0-7695-2787-6

[124] Hendrik Berndt. Towards 4G technologies. Wiley Series in Communications Networking & Distributed Systems, page(s): 295, 2008. ISBN: 978-0-470-01031-0

[125] Wagner M.; Noppens O.; Liebig T.; Luther M.; Paolucci M. Semantic-based Service Discovery on mobile Devices. In Proceedings of ISWC'05 Demo Track. 2005 [pdf document] [Retrieved: April 2, 2010]. Available at: http://www.informatik.uni-ulm.de/ki/Noppens/publications/wagner-et-al-demo-iswc05.pdf

[126] Stefan Zander. A Context-Aware Approach for Integrating Semantic Web Technologies onto Mobile Devices. ESWC 2009 Heraklion: Proceedings of the 6th European Semantic Web Conference on The Semantic Web. Springer-Verlag 2009. ISBN: 978-3-642-02120-6

[127] Bernhard Schandl, Stefan Zander. Adaptive RDF graph replication for Mobile semantic web applications. The First International Workshop on Managing Data with Mobile Devices. University of Vienna, Department of Distributed and Multimedia Systems [pdf document] [Retrieved: March 30, 2010]. Available at: http://www.ubicc.org/files/pdf/3_380.pdf

[128] Sauermann L.; Bernardi A.; Dengel A. Overview and Outlook on the Semantic Desktop. In Proceedings of the 1st Semantic Desktop Workshop, volume 175, Galway, Ireland, November 2005. CEUR Workshop Proceedings.

[129] µJena – Software [www-document] [Retrieved: March 15, 2010]. Available at: http://poseidon.elet.polimi.it/ca/?page_id=59

[130] Jena contributions page. Micro Jena [www-document] [Retrieved: March 15, 2010]. Available at: http://jena.sourceforge.net/contrib/contributions.html

[131] Woerndl, W.; Hristov, A. Recommending Resources in Mobile Personal Information Management. Third International Conference on Digital Society, 2009. ICDS '09, page(s): 149 – 154, 2009. ISBN: 978-0-7695-3526-5

[132] Sauermann, L.; Grimnes, G.A.; Kiesel, M.; Fluit, C.; Maus, H.; Heim, D.; Nadeem, D.; Horak, B.; Dengel, A. Semantic Desktop 2.0: The Gnowsis Experience. Proc. 5th International Semantic Web Conference, Springer LNCS 4273, 2006. ISBN: 978-3-540-49029-6

[133] Nummiaho, A.; Vainikainen, S.; Laakko, T. Utilizing Existing Semantic Knowledge Bases for Creating Annotations on Mobile Devices.
Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, 2009. MDM '09. Page(s): 554 – 559, 2009. ISBN: 978-0-7695-3650-7

[134] Budi Darma Laksana, Cherry Galatia Ballangan. A light-weight MVC (model-view-controller) framework for smart device application. [pdf document] [Retrieved: March 17, 2010]. Available at: http://puslit2.petra.ac.id/ejournal/index.php/inf/article/viewFile/16485/16477

[135] Muller, J.; Lenhart, T.; Henrici, D.; Hillenbrand, M.; Muller, P. Developing Web Applications for Mobile Devices. First International Conference on Distributed Frameworks for Multimedia Applications, 2005. DFMA '05. Page(s): 346 – 350. ISBN: 0-7695-2273-4

[136] Eckstein R.; Loy M.; Wood D. Java Swing. O'Reilly, 1998. ISBN: 978-0-596-00408-8

[137] E. Althammer, W. Pree. Design and Implementation of a MVC-based Architecture for E-commerce Applications. International Journal of Computers and Applications, ACTA Press, Calgary, Canada, Vol. 23, no. 3. page(s):173-185, 2001. ISSN: 1206-212X