

## D2.1 State of the Art and Requirements on Semantic Querying, Discovery, and Composition

Manuel Mazzara<sup>1</sup>, Alexander Urbanec<sup>1</sup>, Joerg Hoffmann<sup>2</sup>, and Adina Sirbu<sup>2</sup>

*<sup>1</sup>Distributed Systems Group  
Information Systems Institute  
Vienna University of Technology  
Austria  
{lastname}@infosys.tuwien.ac.at*

*<sup>2</sup>Digital Enterprise Research Institute  
Leopold – Franzens Universität Innsbruck  
Austria  
{firstname.lastname}@deri.org*

February 28, 2007



### **Abstract**

This document provides an overview on the state of the art for semantic query, composition and discovery. We also discuss the syntactical counterpart presenting the related technologies and the most relevant proposals of standard. In addition we think that it is very important to put the emphasis on commonalities and differences between the syntactical and semantic approaches and, for this reason, we present what has been developed so far to make them work together taking the best form both. This contribution is intended to be very important in trying to make the Web Services and the Semantic Web communities collaborate.

### **Abstract**

Das vorliegende Dokument gibt einen Ueberblick ueber den aktuellen Stand der Entwicklung in den Gebieten semantischer query, composition und discovery. Weiters behandeln wir verwandte syntaktischen Technologien und die wichtigsten Standardisierungsvorschlaege in diesem Forschungsfeld um einen besseren Gesamtueberblick und eine leichtere Einordnung dieses Bereichs zu ermoeeglichen. Insbesondere erscheint es uns wichtig Gemeinsamkeiten und Unterschiede zwischen Anstze dieser beiden Welten (syntaktisch und semantisch) zu beleuchten, sowie Strategien zu betrachten, um diese zu verbinden und so das Beste aus beiden Feldern in einem gemeinsamen Ansatz zu nutzen. Durch diese Herangehensweise soll die vorliegende Arbeit nicht nur einen allgemein berblick ber aktuelle Forschung bieten, sondern auch zu einem wichtigen Beitrag werden, um Web Service und Semantic Web Communities naeher zusammen zu bringen.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Service Oriented Computing . . . . .	3
1.2	Service-Oriented Architecture . . . . .	4
1.3	The need for Composition and Discovery . . . . .	4
<b>2</b>	<b>Syntactic and Semantic Approaches</b>	<b>5</b>
2.1	Syntactic Approach and Composition . . . . .	6
2.1.1	Technological Background . . . . .	6
2.1.2	Composition Requirements . . . . .	8
2.1.3	Composition . . . . .	8
2.1.4	Composition Engines . . . . .	9
2.2	Semantic Approach and Discovery . . . . .	9
2.2.1	Why Semantics? . . . . .	9
2.2.2	From RDF to WSMO . . . . .	12
2.2.3	Relationships between WSDL and OWL-S . . . . .	13
2.2.4	Discovery . . . . .	14
2.2.5	Semantic Query . . . . .	15
2.2.6	Semantic Web Service Reasoner and Validator . . . . .	16
2.2.7	Requirements on Semantic Discovery and Composition . . . . .	16
<b>3</b>	<b>Approaches towards an Integration</b>	<b>17</b>
3.1	OWL-S/WSMO to/from BPEL Mapping . . . . .	18
3.2	Semantically Enriched Signatures - Contracts . . . . .	18
<b>4</b>	<b>Our Approach</b>	<b>19</b>
4.1	Semantic Discovery . . . . .	20
4.2	Semantic Composition . . . . .	21
4.3	Interface Language . . . . .	21
4.4	From the Interface Language to BPEL . . . . .	22
4.5	BPEL Validation . . . . .	22
4.6	Advantages over Previous Methodologies . . . . .	23
<b>5</b>	<b>Conclusions</b>	<b>23</b>

## **Outline**

This outline is intended to partially integrate the index, giving the reader some additional information on the content of the sections in such a way he could read only those parts relevant for his specific purposes.

In the first section a general introduction on Service Oriented Computing and Web Services technologies is given. Here we explain the need for composition and discovery in B2B integration. In the second section we describe the syntactic and semantic approaches and, for each of them, we present the main proposals so far discussed in literature. We try to give also a partial answer to the main objections that usually arise when presenting the advantages of Semantic Web in certain contexts. To do this we discuss the potential advantages we see for banks and insurances. Once we understood the main ideas behind these two different worlds, in the third section we try to take the best from both presenting some hybrid approaches where the other two can somehow collaborate. Section four is dedicated to the novel approach to integration we intend to realize in the context of SemBiz. Finally, in the last section we add some conclusive remarks.

# 1 Introduction

In this section we give a general introduction on the context in which the SemBiz project finds its background, i.e. the notion of Service Oriented Computing and its candidate implementation by the use of the so-called Web Services technologies. From this discussion the need for composition and discovery will emerge considering B2B integration and the problem of crossing organizational boundaries which are two of the main issues related to Web Services. We will briefly discuss also this aspect.

## 1.1 Service Oriented Computing

Service Oriented Computing (SOC) is a paradigm for distributed computing and e-business processing that finds its origin in object-oriented [2] and component computing [49]. One of the main goals of SOC is enabling developers in building networks of integrated and collaborative applications, regardless of both the platform where the application or service runs (e.g., the operating system) and the programming language used to develop them.

In this scenario, Web Services technology is simply a widespread accepted instantiation of SOC providing a platform on which it is possible to develop applications taking advantage of the already existing Internet infrastructure. This does not mean that the set of technologies we are introducing here is the only one which make possible to realize SOC: regarding the many layers there have been indeed different proposal and approaches to make the vision an actual implementation. We are simply describing the more successful and accepted set of technologies on which already the majority of services are running. The success of these technologies is mainly due to the fact that they rely on the already existing Web protocols, i.e. they run on a machine that was already set up, well known and widely used.

In the overall intention, the main goal of SOC should be to facilitate the integration of newly built and legacy applications both within and across organizational boundaries avoiding difficulties due to different platform, heterogeneous programming languages, security firewall, etc... The basic idea behind this orientation is to allow independently developed applications (using different languages, technologies etc...) to be exposed as services and then interconnected exploiting the already set up Web infrastructure with relative standards: (HTTP [15], XML [61], SOAP [47] and WSDL [5]).

In the original vision of the Web the notion of semantic was essentially absent. This has been something that came out later. In the same way, also in the Web Services scenario the further step would be to introduce some notion of semantic into these technologies in order to facilitate automated discovery and composition. This semantic notion is very innovative, although almost not used yet in real life. One of the purpose of this deliverable is also to underline the importance and the utility (and of course feasibility) of this innovation.

A Web Service, as working today, supports its operations by associating any functionality to specific access points available over the network in such a way that they can be exploited, in turn, by other services. Many specific definitions of Web Services are available in literature, some are similar with subtle, not always significant, differences. Here, for simplicity, we prefer to give the definition taken directly by the W3C [55]:

“A Web Service is a software system identified by a URI, whose public interfaces

and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols”.

Many things have been said around Web Services and also this hype of revolutionary flavor has been spreaded around. Our opinion is that nothing of revolutionary really exists behind Web Services, even in the semantic vision. It is an evolutionary technology, not just suddenly existing. There is no revolution here, Web Services have to be seen as an evolution based on already existing Internet Protocols. In some sense, they are simply the expected next step.

## 1.2 Service-Oriented Architecture

The service oriented paradigm of computation can be abstractly implemented by the-so called service-oriented architecture (SOA) [3]. This attempt can be considered the latest of a long series in software engineering addressing the reuse of software components.

Historically, the first major step of this evolution has been the development of the concept of *function*. Using functions, a program is decomposed into smaller subprograms and writing code is focused on the idea of the application programming interface (API). An API, practically, represents the contract to which a software component has to commit. The second major step was the development of the concept of *object*. An object is a basic building block which contains both data and functions within a single encapsulated unit. With the object-oriented paradigm the notions of classes, inheritance and polymorphism are introduced. In this way classes can be viewed as a lattice. The next step in this evolution has been introduced with the advent of SOC. The *Web Services Programming Model* consists of three components: service consumers, service providers and service registrars. A service registrar acts as an intermediary between the provider and the consumer, so they are able to find each other. A service provider simply publishes the service. A service consumer tries to find services using the registrar; if it finds the desired service it tries to set up a contract with the provider in order to consume such a service and thus do business. Picture 1 shows the model and the state of the art protocols choices (which means the most common but not the only ones) for implementing interactions between components, i.e. SOAP, WSDL and UDDI.

Note that, although in this picture we are committing to specific protocols, nothing forbids the use of different options. In particular, this should be clear when discussing the semantic scenario which is of course not excluded in this representation. In particular we will see how the use of UDDI can be connected with semantic discovery.

## 1.3 The need for Composition and Discovery

Web Services technology promises to facilitate B2B integration by replacing proprietary interfaces and data formats with a standard Web-messaging infrastructure. Web messaging is sufficient for some simple application integration needs; however, it does not adequately support the complete automation of critical business processes. The first generation of Web Services technology has largely focused on the Web-messaging foundation supported by SOAP (Simple

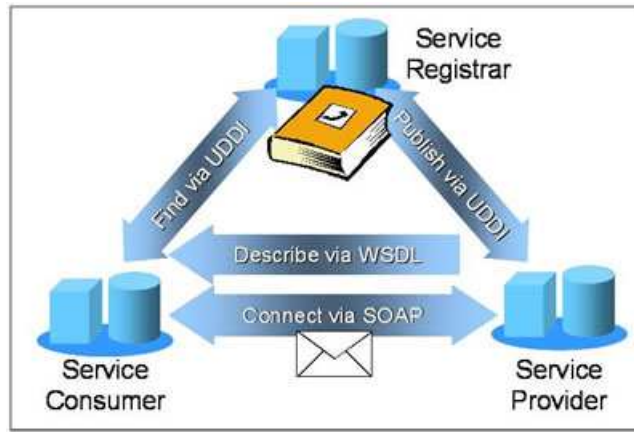


Figure 1: Web Services Programming Model

Object Access Protocol) and WSDL (Web Services Definition Language). WSDL is used to describe a Web Service in terms of its ports (addresses implementing this service), port types (the abstract definition of operations and exchanges of messages), and bindings (the concrete definition of packaging and transportation protocols, such as SOAP, are used to inter-connect two conversing end points). Although this foundation has the ability to specify critical information and requirements relating to the business process context, it does not support business processes that cross organizational boundaries. To truly integrate business processes across enterprise boundaries, merely supporting simple interaction using standard messages and protocols is insufficient. Business interactions require long-running interactions driven by an explicit process model. This raises the need for Web Services composition languages and automated discovery.

## 2 Syntactic and Semantic Approaches

With the goal of giving technological support for the service programming model, different approaches have been developed. We classify them in two different and very important categories (syntactic and semantic) and, for each one, we will present the main proposals between the different protocols, languages and standard so far discussed in literature.

Since the syntactical approach presently finds its main advantage in having concrete and easy to use compositional tools, we prefer to discuss together this approach and the compositional issues. Thus, we merge these two descriptions in a single section hoping to make clearer the overall scenario by means of this synergy. In the same way, we want to introduce semantics and discovery together, since it would be semantic discovery the main advantage of this approach. For semantic discovery we mean *discovery mechanisms of services based on semantic criteria*.

A complete synopsis between all the proposals and concepts is not reasonable here since it would make very unclear the presentation. Indeed, every community or working group basically tries to push its "toys" in an interesting scenario due to the strong industry involvement and interests. This does not mean that every paper or proposal is mature in the same way.

We decided to classify things and give ideas and relationships between these, at least in our present understanding. This has been an hard work, more complicated than simply listing acronyms and Web sites. We want to give the general idea of the most relevant activities providing adequate references for all the details. The interested reader can go into the details surfing the appropriate resources we indicate. After this overview, we discuss also those strategies that try to take the best from both the approaches.

## **2.1 Syntactic Approach and Composition**

The interesting thing in the Web Services Programming Model is the fact that a service can itself use several other services and each of these services will be, in turn, based on the same model. This means to have a recursive use of the model where, notably, the overall scenario will be transparent to the final consumer. In this way, Web Services technologies provide a mechanism to build complex services out of simpler ones: this practice is called *Web Services Composition*. A composition consists in the aggregation of Web Services by programming the relative interactions and has the ability to reuse the created aggregations [19]. To program a complex cross-enterprise task or a business transaction, for example, it is possible to logically chain discrete Web Service activities into inter-enterprise business processes. Composing Web Services and pushing between the collaboration between different partners can be an opportunity for companies to share cost and responsibility instead of redesigning and reimplementing already existing functionalities. However, while the technologies intended to connect Web Services on a point-to-point basis can be considered well established, modeling complex business scenario involving a large number of participant and defining - often long running - workflow and dataflow between the different services - owning to different domains - is still an interesting research direction.

Different organizations are presently working on composition proposals from the syntactic side. Indeed from the industry point of view this approach has been largely understood and accepted, while the semantic one is still lacking some concrete support (although it promises interesting developments). Form this fact the need emerges to find an interesting roadmap to put the best ideas together getting the best advantages of both the approaches.

With this approach the interface of a service is defined in an Interface Definition Language called WSDL which is very close do the CORBA IDL ([7], [16]) in some sense. Basically the service is seen like an RPC with the relative "signature" (i.e. the syntax of messages entering and leaving the service). The order of messages exchange between the services is instead defined with other languages, e.g. WS-BPEL or similar. In this scenario is it not yet clear which standards will arise but BPEL (Business Process Execution Language) is a good candidate. The approach lacks of a semantic definition of components, since WSDL is indeed only a syntactical interface definition.

### **2.1.1 Technological Background**

What made Web Services interesting and successful since the beginning has been also the fact that they could rely on a working platform. We very briefly describe the main characters of this scenario.

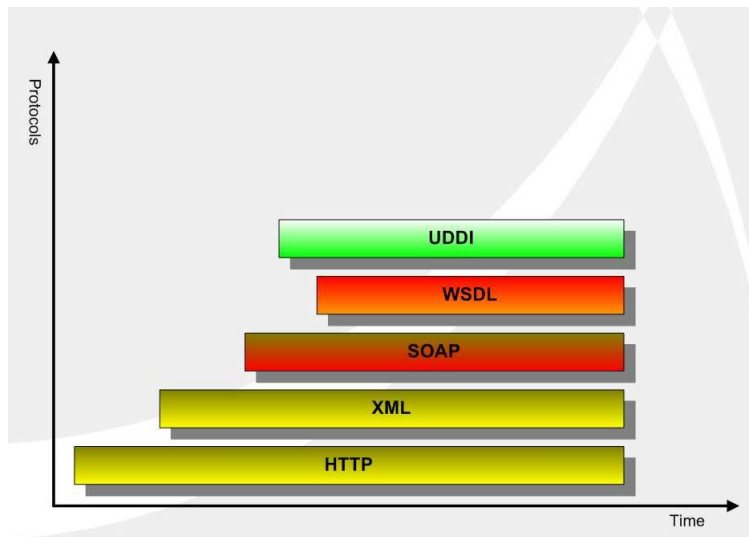


Figure 2: Syntactical Web Services Protocols Stack

At the beginning there was HTTP (HyperText Transfer Protocol) [15]. It is a well-known communication protocol with which it is possible to send information from one point to another point on the Internet. The information we send over the wire can be structured using XML (eXtensible Markup Language) [61]. The XML metalanguage defines the format of the information and it is a basic foundation for the upper layers. Then, the Web Services framework here is essentially based on three technologies, at least in its initial phase: SOAP, WSDL and UDDI. SOAP [47] (Simple Object Access Protocol) is a protocol defining how services have to interact with each other. It is based on XML and describes the documents structure to be used for invoking services. Basing the specification on XML makes it possible to overcome the problems that arise when integrating different operating systems, object models and programming languages. In order to achieve flexibility, SOAP abstracts away from the underlying transmission protocol (typically HTTP, SMTP, FTP, ...). WSDL [5], instead, describes the operations supplied by services, including expected parameters and return values. Finally, since it is necessary to have a central market place where publishing WSDL documents to allow other parties to find and use them, UDDI (Universal Description, Discovery and Integration) [53] has been introduced.

The complete stack is shown in Fig.2 but the picture deserves some comments with respect to the presence of UDDI as registry technology. Indeed, leading companies like Microsoft, IBM and SAP have recently considered unsuccessful the original intent of the project and the general feeling is that they could shut down any support on further developing and usage. At this stage, anyway, the final destiny of UDDI it is not very clear to us and also how the motivations for its failure can be justified would deserve a long discussion. Indeed, out of this situation some proprietary solutions are coming out. We decided to include UDDI in the stack as a leading technology for discovery since it has been for years at the center of the discussion as a reference point. The interested reader can consider the UDDI evolution looking at WSIL [57].

### 2.1.2 Composition Requirements

Composition proposal should satisfy all the following main requirements relevant for the workflow-based composition of services:

**Flexibility:** it can be achieved by providing a clear separation between the process logic and the Web services invoked. This separation can usually be achieved through an *orchestration engine* that handles the overall process flow. With this flexibility, an organization can easily swap out services as business needs change.

**Basic and structured activities:** An orchestration language must support activities for both communicating with other Web services and handling workflow semantics. One can think of a basic activity as a component that interacts with something external to the process itself. In contrast, structured activities manage the overall process flow, specifying which activities should run and in which order.

**Recursive composition:** A single business process can interact with multiple Web services. However, a business process can itself be exposed as a Web service, enabling business processes to be aggregated to form higher level processes.

**Persistence and correlation:** The ability to maintain state across Web services requests is an important requirement, especially when dealing with asynchronous Web services. The language and infrastructure should provide a mechanism to manage data persistence and correlate requests in order to build higher-level conversations.

**Exception handling and transactions:** Orchestrated Web services that are long-running must also manage exceptions and transactional integrity. For example, resources cannot be locked in a transaction that runs over a long period of time.

### 2.1.3 Composition

The most important composition proposals in the syntactical setting in the past have been IBM's WSFL [26] and Microsoft's XLANG [60]. XLANG was a block-structured language with basic control flow structures such as `sequence`, `switch` (conditional), `while` (looping), `all` (parallel) and `pick` (choice based on timing or external events). WSFL instead was not limited to block structures and allowed for directed acyclic graphs (DAG). Iteration was only supported through exit conditions - that is, an activity iterates until its exit condition is met. These two proposals have finally converged to the Web Services Business Process Execution Language [1] (WS-BPEL or BPEL for short) which is presently a working draft by OASIS. Another recent proposal in phase of standardization by the World Wide Web Consortium (W3C) is WS-CDL [18]. Both allow the definition of workflow-based composition of services with some similarities and some differences. Describing in details a synopsis between these two proposals is beyond our scope, you can find an interesting synopsis in [17].

### **2.1.4 Composition Engines**

BPEL is a layer on top of WSDL. While the information contained in WSDL documents defines message types and port types - representing the operations supported by the service and the interaction modalities - BPEL itself specifies the flow of actions to perform (using precisely the information contained in WSDL documents). A BPEL document is an XML-based document that can be executed by an orchestration engine which is the central coordinator. The engine will read the BPEL document and will invoke the necessary Web Services in the specified order. The process itself will be offered as a Web Service and can be invoked in the same way. Many BPEL engines have been already released so far, some of them are commercial like the Oracle BPEL process Manager [32] while some other are open source projects [31].

## **2.2 Semantic Approach and Discovery**

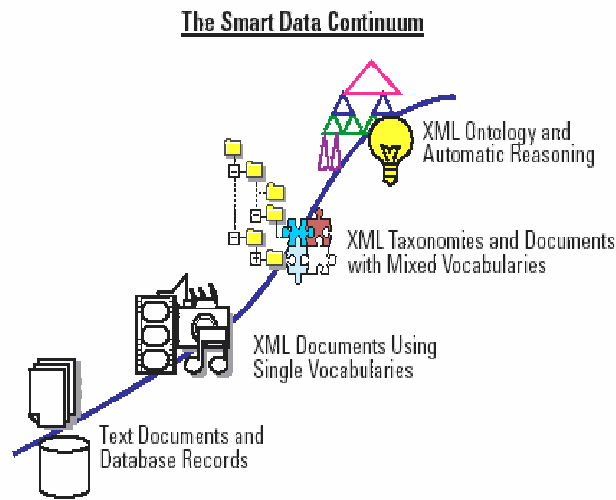
The semantic approach finds its root in a different community, the Semantic Web community. Semantic Web services are the fruitful combination of Semantic Web and Web service technologies. The purpose of Semantic Web services is to overcome the limitations of current Web services by adding explicit semantics to them. The exploitation of such semantics would then resolve the interoperability issues and automate the Web service usage process. OWL Service Ontology (OWL-S) [34] and Web Service Modeling Ontology (WSMO) [59] are the major initiatives for providing semantic annotations on top of a Web service infrastructure.

This part introduces some of the most important semantic-approach related technologies, starting with a short overview of the general meaning of "semantic". After this we present a motivating example and then a specific introduction to RDF and XTM - currently the most well known languages for the description of basic semantic data-structures. In the following RDF-S, OWL and WSML are presented as ways of describing higher-level semantic information and, finally, the usage of this concepts in the context of Web Services is shown by the introduction of OWL-S and WSMO, two ontologies for Semantic Web service description.

### **2.2.1 Why Semantics?**

Semantics are meaning, meaning of data and information. While XML was introduced many years ago as a machine-readable language to represent structured data, the motion of Semantic Web aims at annotating this data so that computers are able to use this information in a better and more efficient way. Semantic languages like RDF allow us to describe data-entities as well as their relationships, and schema languages like RDF-S and OWL are used to add an additional layer on this information to further define the general data-model that lies beyond these instances. Using this kind of self-contained semantic information and relationships instead of relying on pure syntactical structures and hard-coded interpretations takes the focus from the application-level back to the data and allows machines to perform more meaningful and automated tasks (refer to Fig.3)

Before presenting the state of the art in this context, let us consider a motivating example which could appear simple but which basically shares much of the logic with real case studies (or at least with some subset of bigger case studies). The example wants basically to show the need for semantic discovery.



The trend is to put the "smarts" in the data, not in the applications.

Figure 3: Smarter Data instead of Smarter Applications

Fig.4 describes a scenario in which your basic requirement is providing a service where the input data is the telephone number for which you want to know the current bill as the output. Let us suppose that the only way to get such information is to retrieve two different pieces of information remotely located in different places. Firstly you have to identify the service provider related to that input number and secondly you need to get an "USERCODE" which represent the user itself <sup>1</sup>. Finally, the third service provides the desired information given both the code and the provider as input.

In the case you want to implement a "bigger" service including the described functionalities it will be necessary to "coordinate" (or "compose", if you prefer) this three smaller services with distributed responsibilities. The main problem you meet here is basically the fact that in  $WS_1$  the input data, as it would be identified in a WSDL document, is called "USERNUM" while in the second it is simply "NUMBER", although the intended meaning is exactly the same - i.e. they are precisely the same thing from a semantic point of view but not from the syntactical one. This is precisely the case in which it is of main importance adding some "still syntactical" annotations to give anyway more information on the semantics of the data. Consider that we call this *semantic information* but it should be clear that it is still a syntactical one but it is a meta-information on the basic syntax (information about the information).

Basically every possible application area of semantic web could be exploited by industry companies: automated agents utilizing data, knowledge management, search and advertising scenarios, exchange of data, interoperability and composition of complex systems... The interested reader can find a complete account on this in [64]. There are, anyway, some sectors that are considered less suitable: banks and insurances. Here we want to briefly discuss the potential advantages we see even in these areas. Specifically we want to answer, with our specific point

<sup>1</sup>In this hypothetical example we could suppose that the service providing the correspondence between phone number and "USERCODE" cannot holds the actual name for some reason related to privacy

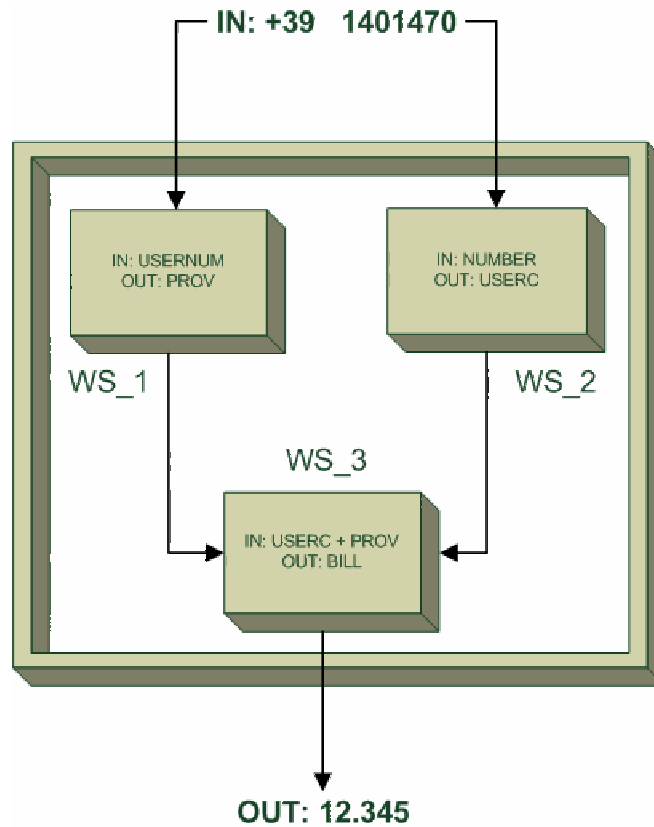


Figure 4: Motivating Example

of view, to the main objections that usually arise when referring to this scenario.

**Banks** A reasonable objection could be that, in general, banking seems not to be a really suitable area for semantic web and it should not be able to take a large advantage out of this. One of the motivations is the fact that banks usually have very standardized data, closed interfaces and fixed data structures. Especially in bank to bank interactions it seems that there will be hardly any need for semantic descriptions since everything is precisely defined. Generally for security reasons, the networks are closed and not employing open standards and the processes are fixed, predefined and clearly structured. Consider, for example, an international money-transfer: which motivation can you find for a semantic description?

Indeed, we agree on this point of view but this does not mean that banking cannot be an interesting target for the semantic web. All the limitations we listed above hold only for the core business of banks. Nowadays, there are a lot of other areas and potential business in which banks are interested that are more open and might be suitable for the goal. In a future of competition between banks for attractive conditions the area of investments and research are becoming more and more important. Here open information comes into the game. For example, when buying stocks, people want information about companies, stock history, shares etc. All this could be semantically described and exchanged/shared by banks, brokers, enterprises etc...

Consider if the annual reports of an enterprise would be available using the "semantic web": it would be much easier for banks to use the data without manual interaction! Finally, for internal purpose (knowledge management etc.) semantic technologies could be also used.

**Insurances** For insurances considerations similar to the ones we discussed for banks still hold. But, since they have more interactions on a data level than banks, it is easy to find more aspects where semantic web could be useful. For example insurances need and use data about people (legal status, health report, employment history etc.), cars, buildings ... (depending on what type of insurance company we are talking about). Furthermore, insurance companies have interfaces to hospitals, doctors, car companies, police and so on, and hence there would be some areas where data could be semantically enriched and exchanged to limit manual transformation and the need for parsers and converters tailored on single input or output formats. So, the advantages of semantic web would be centered very much in the area of information gathering and knowledge management.

### 2.2.2 From RDF to WSMO

At the beginning there was **RDF** (Resource Description Framework) [39] which is a recommendation from the W3C for creating meta-data structures defining data on the Web. Initially, it has been designed in the context of Semantic Web to provide a method for classifying and describing resources on the World Wide Web in order to improve both searching and navigation which indeed was - and still is - the main goal of Semantic Web. Yet, by using the general concept of URIs as unique and globally valid identifiers for these resources, RDF does not necessitate the target of URIs to actual (physically) exist and therefore allows as well the description of abstract concepts, ideas or real-life entities. Doing so, RDF is very simple in its basic structure: every RDF-Statement is an object-attribute-value triple, connecting a given entity with another entity (be it a literal value or a resource) by using a directed relationship. The RDF model is completely detached from any concrete syntax but, in general, the W3C-recommendation RDF/XML is used as the syntax for representing RDF-information.

**Topic Maps** shares with RDF a common goal: defining a standardized language which can be used to describe data in such a way that every application implementing the standard is able to understand and use it in a well-defined and self-contained way. Anyway, developed as an ISO-Standard, Topic Maps have a slightly different orientation than RDF. Its origin lies in the domain of library science where it was developed as a modern successor of a back-of-the-book index to make information findable. Hence, they applied a greater focus on the general structure of data and, while RDF-resources simply have properties that have values (either literals or other resources), Topics in Topic Maps (the equivalent to RDF resources) are described using a much broader set of pre-defined characteristics (like names, variants and roles played in relationships). Besides, the characteristics of relationships themselves are more complex in Topic Maps than in RDF since they may involve not two, but any number of participants. Yet, as a result of these differences, Topic Maps are more heavyweight (due to their more explicit structure) and besides less formalized and much less used than RDF (although [9] and [10] illustrate that there are current motions to make up especially the formalization deficit).

Similar to what XSD (XML Schema Definition) [62] does for XML, **RDF-Schema** [40] is used to declare a general vocabulary for RDF, describing involved resources and relationships. Beyond others, range and value restrictions for properties as well as a simple hierarchy of nodes

(super and sub-types) are supported in RDF-S, allowing to build a basic underlying structure of the involved data. Like in XSD, any RDF-data-instance can then be validated against a given RDF-Schema document.

Taking this concept a step further and built on top of RDF, **OWL** (Web Ontology Language) [33] can be roughly viewed as its extension with a higher level vocabulary to describe and constitute the overall structure that lies beyond RDF data. It incorporates additional features like relationships between resource-types (classes), cardinality-support, equality (between resources and relationships) and characteristics of properties (e.g. defining a property to be symmetric instead of unidirectional). While OWL is syntactically only consisting out of RDF statements (like XSD is pure XML), it is the foundation to build a general data model and hence for achieving data-interoperability between applications and a common understanding on what a given piece of information actually "means". From OWL also started the **OWL-S** (Ontology Web Language for Services) [34] initiative which aims to define an OWL-based Web Service ontology supplying to service providers the semantic desiderata, as discussed in the previous section. Here composition is based on pre- and post-conditions plus inferencing like in the usual semantic setting.

The OWL-S Process Model is fundamental for composition and it is really relevant to understand differences and relationships between BPEL and OWL-S and all the difficulties that can be met in making them cooperating. The process model is built as a workflow of different processes where each of this is constituted by four components: *inputs*, *outputs*, *preconditions* and *effects*. Results specify what effects are generated given a particular condition. Processes in the workflow are related to each other by a control flow allowing the specification of the causal relations between them (for example sequence, parallel, synchronization points, conditional, non-deterministic selection).

Together with this proposal, another leader initiative is running on in the field of Semantic Web Services: **WSMO** (Web Service Modeling Ontology) [59]. WSMO refines and extends WSMF (Web Service Modeling Framework) [25] to a meta-ontology for Semantic Web services. WSMF defines a rich conceptual model for the development and the description of Web services based on two main requirements: maximal decoupling and strong mediation. WSMO aims at describing all relevant aspects related to general services which are accessible through a Web interface, with the ultimate goal of enabling the automation of tasks involved in both intra and inter-enterprise integration of services, such as discovery and composition. The major differences between the OWL-S and WSMO approaches are discussed in [23]. **WSML** (Web Service Modeling Language) [58] is a formalization of the WSMO ontology, providing a language within which the properties of Semantic Web services can be described. WSML is a family of formal representation languages with its roots in Description Logics and Logic Programming.

### 2.2.3 Relationships between WSDL and OWL-S

WSDL is an XML-format describing services as a collection of endpoints sending or receiving messages. This description is done abstractly, before being bound to the specific protocols and formats (e.g. HTTP, SOAP, MIME etc.) used to implement the actual Web Service and to define the actual endpoint. It only provides for a very basic description of messages by employing simple XML-Schema types and it does not provide for directly adding richer or

structured semantics (like it could be done in OWL-S) and is aimed on single or synchronous interactions. OWL-S, on the other hand, moves the focus on describing not only the interfaces, but the service as a whole, its behavior and its messages.

In general, WSDL and OWL-S should be seen as complementary [43]. OWL-S, for example, explicitly includes an ontology for Grounding that employs WSDL [37] and can be used to define the mapping between entities of an OWL-S document and selected WSDL-elements. In this process, OWL-S atomic processes correspond to WSDL operations while inputs and outputs of OWL-S abstract processes to WSDL-message parts. OWL classes are used as abstract types of the messages in WSDL (for a much deeper analysis of the relationship between OWL-S and WSDL you may refer to [38]).

#### 2.2.4 Discovery

In the previous section we largely discussed an hot topic like composition. However, especially when dealing with the composition and integration of Web Services into large-scale processes, another issue that will surely arise is (automated) discovery. And this would be the point in which semantics should play the main role since BPEL in itself only allows for manually specifying the coordination among multiple Web services, expressed in WSDL.

In this section, we discuss the work done in automatic Web service discovery based on semantic descriptions of functionalities of Web services and user requests. Other Web service discovery means that offer a low level of automation, such as the use of UDDI [53] registries alone for locating Web services, will not be discussed.

Several existing approaches to dynamic Web service discovery rely on OWL-S or DAML-S (as the predecessor of OWL-S). For instance, [44] and [46] propose approaches based on DAML-S for matching service advertisements and requests. Both approaches rely on subsumption reasoning to determine if a match exists between the information defined in the service profile and the information given in the request.

In [44], only the information transformation aspect of the service is considered, i.e. only inputs and outputs are taken into account in the discovery process. A matching of a service profile and a request goal (also modelled as a profile) occurs when all the outputs of the goal are matched by (possibly a subset of) the outputs of the Web service capability, and all the inputs of the Web service capability are matched by (possibly a subset of) the inputs of the goal. Different degrees of match are determined for the checking of inputs and outputs depending on the subsumption relationship that holds between the pairs of outputs (resp. inputs), which in turn result on a ranking of the matching services.

One of the problems of this approach is that every requester output has to be checked against all the outputs of every service, the number of necessary output subsumption tests being therefore  $n*m*s$ , where  $n$  is the number of outputs in the request,  $m$  is the number of outputs in the service capability, and  $s$  is the number of available services. The same applies for input checking. Moreover, only inputs and outputs are considered, thus only modelling the communicative aspects i.e. information flow of the service, excluding the effects on the real world. Some limitations are derived from the use of DAML+OIL, as it does not offer a formalism to express rules. Because of this lack of rules, we cannot define how the outputs of the service relate to its inputs and therefore the functionality of the service is not completely captured. The direct consequence is that we cannot guarantee that the service will actually fulfill the requester goal.

The approach in [46] uses as well DAML-S and DL subsumption reasoning to perform discovery. Here, the whole profile (including inputs, outputs, preconditions and effects) is defined as a subconcept of the DAML-S service profile, and then classified in the subsumption hierarchy defined by the advertised service profiles. The kind of matches considered are different from [44]. This approach can be used for service discovery if the inputs and preconditions are removed. However, [46] does not solve some of the limitations encountered in [44]: the lack of rules for DAML+OIL results in an incomplete description of the service functionality.

Solutions for integrating such algorithms for capability matching of DAML-S service profiles with the existing Web service standards are proposed in both [43] and [42]. The first cited approach [43] maps service profiles of DAML-S to UDDI-records and stores this information within the registry utilizing the generic UDDI concept of a tModel [51]. Semantic information can thus be integrated in the UDDI registry and used for publishing and matching services. [45] takes this concept to the next level by presenting an approach to enhance the UDDI-registry with a so called "matchmaker-module" that can process the mapped and integrated OWL-S descriptions for handling requests (refer to Fig.5)

The second cited approach [42] has a similar purpose, but is consistent with the use of industry standard WSDL rather than requiring DAML-S. It involves relating WSDL constructs to DAML+OIL ontologies in the Web service descriptions and then providing an interface to UDDI that allows querying based on ontological concepts. The annotation of registries and its specialization in domains helps to deal with a potentially huge number of published services. However, this annotation does not relate inputs, outputs, preconditions and effects, therefore providing a not sufficiently accurate description of the service functionality. Furthermore, this description is too WSDL centered, while approaches such as OWL-S and WSMO take a more flexible approach, not imposing any language for the grounding of Semantic Web services.

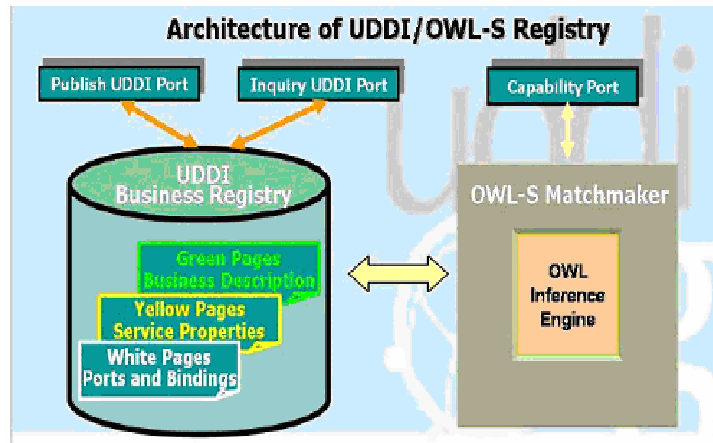


Figure 5: OWL-S Matchmaker

### 2.2.5 Semantic Query

The more complex and distributed data becomes, the bigger is the need for powerful and efficient querying. While the idea of structured queries is almost as old as IT itself (e.g. database-

query languages like SQL), over the last years a high amount of effort was put in developing specialized query languages for open standards like XML. XQuery [63], the most widely used, was specified by the World Wide Web Consortium to allow for flexible and simple extraction of data out of XML documents and any other XML-based data store (see [54] for more details on this issue). However, when dealing with self-contained knowledge-representations (and this is without doubt one of the main goals of both RDF and Topic Maps), it is one of the main requirements to have a well-defined and standardized syntax for (possibly complex) queries not only on a syntactical, but also on a semantic layer. While the W3C has already started some standardization work in this area [35], a multitude of different semantic query languages were developed all over the world in the last few years. As it would not be possible to provide a reasonable overview of their characteristics and differences within the confines of this paper, you may refer to [14] that offers a very good comparison of six different RDF Query Languages (RQL, SeRwL, TRIPLE, RDQL, N3 and Versa). Here we only focus on a very recent development in this area that seems promising: the SPARQL Query Language for RDF (recursively for "SPARQL Protocol and RDF Query Language").

**SPARQL** has become a candidate recommendation by the W3C in April 2006 and is currently available as a working draft in the 4th release [48]. Basically, SPARQL is based on graph pattern matching where one or two of the three positions of the typical RDF triplet are left empty to be filled with variables. In general, SPARQL is used like XML Query or SQL, so to get a result set of elements out of some - more or less detailed - restrictions. As OWL is written in RDF, SPARQL can also be applied to query OWL-S ontologies and hence Semantic Web Service descriptions.

Similar to the one for RDF, there is currently a movement by ISO in the area of Topic Maps for creating a single standard, and the language actually selected to be the basis for this "TMQL" (Topic Map Query Language) is **Tolog** [11]. Originally inspired by Prolog, Tolog supports standard Horn clauses, but also NOT and OR-operators and other SQL-known features such as projection, counting and sorting. Like SPARQL, it is based on matching predicates against the data source (in this case the Topic Map) and binding the results in variables (but also including support for the natural structures of Topic Maps and allowing for more flexible predicate definition).

### 2.2.6 Semantic Web Service Reasoner and Validator

There have been some attempts to contextualize the literature on Process Reasoners in the Semantic Web Services scenario. In [41] you can find the main ideas and useful pointers. The focus in this paper is on OWL while in [28] DAML-S is the reference. An interesting work developed before the advent of Semantic Web but that can give inspiration is [8].

### 2.2.7 Requirements on Semantic Discovery and Composition

The document described so far, in details, the state of the art for Semantic Web Services. As it should be clear automatic discovery and composition of services represents today one of the most interesting open challenge and understanding features and requirements is the first step to contribute to this incoming scenario.

Automatic Web service discovery and composition consist in automatically locating and aggregating services providing specific functionalities and they need a high-level and abstract way to semantically describe services plus opportune reasoning procedures for the actual matchmaking. Here we briefly overview the main requirements related to this context, for a complete account refer to [22].

**Services Combination:** Semantic Composition requires more than simply describing the flow between a priori-known services, like BPEL and other related languages presently do. It should also provide the possibility of combining services to get out a specific functionality even if a request cannot be satisfied using available services.

**Need for declarative description and algorithms:** The discovery process needs to be based on some kind of semantic match between a declarative description of the desired service and a description of the actual services. So it is necessary to have both languages to express this description and algorithms for performing the actual match.

**Capability Description:** The description is the actual capability of the service and it should be described separately from the service description (to relate specific refinements to the same generic high-level capability, to allow a service to present two different capabilities, etc...).

**Multiple Availability of a Capability:** Different services can provide the same capability.

**Capability Information:** A capability has to include different information to allow dynamic discovery and composition: at least pre and post-conditions, a textual description, a reference to the services and a proper identifier to allow references to it.

**Abnormal Situations:** A description has to allow the inclusion of compensation information.

### 3 Approaches towards an Integration

Once we understood the main ideas behind these two different worlds (emerging from different background and communities) the interesting challenge is trying to take the best of both defining an opportune framework where these machines can somehow collaborate.

Anyway, consider that this connection is not trivial since, for example, BPEL and WSML are related to completely different domains and approaches even with somehow different goals: the first finds its origin in the *workflow technology* background while the second has its root in the *semantic Web* initiative where the main tool is the use of *ontologies*. How to make these two things collaborating and why? Are there advantages in such a hybrid approach? And which ones?

In this section we describe two work methodologies that have been at least partially discussed in literature for integrating the two approaches and we present advantages and disadvantages of each of these. Sometimes these works we present have been not explicitly contextualized as integrating approaches in literature. One of the really complicated things of our work has been

to find them understanding that they could fit in the bigger picture we want to build here. In this sense, we believe that our contribution is not merely a review of the state of the art but a first attempt to find hidden connections between heterogeneous works developed in apparently different contexts. In the next section we will present a novel integrating approach that we intend to adopt for the SemBiz project.

### **3.1 OWL-S/WSMO to/from BPEL Mapping**

The first thing one could think to relate the syntactic and the semantic world is to provide some kind of translation like when we want to switch from a language in which we do not feel comfortable to another one far better for us. So we could think to provide a mapping - basically translations rules - from OWL-S (or WSMO) to BPEL (or vice versa).

However, it is not yet very clear how to map OWL-S to BPEL (and vice versa) since the second aims to represent the composition of Web Services (basically the interaction) while the first provide only a representation of the Process Model of one single service, leaving the composition problem to a different layer. Nevertheless, tools mapping OWL-S to BPEL exist [6] since OWL-S allows the description of the dynamic behavior of a service by means of its process model. Of course the final mapping is strongly limited by the fact that the original OWL-S process is the description of a single service and not of a net of services to be composed.

Instead, for what concern WSMO, as far as we know the process model is still under development. Once it is released it will be possible to consider also a mapping between BPEL and WSMO with the same basic limitations of OWL-S.

### **3.2 Semantically Enriched Signatures - Contracts**

The idea on which the second methodology is based is the fact that WSMO could somehow complement BPEL. Indeed, using semantic technologies Web Services can more easily be discovered and composed into bigger processes basically adding semantic annotation for automatic discovery of suitable services.

As explained above the syntactical approach is characterized by the fact that a service is simply described by means of a "signature" describing the syntax of messages entering and leaving the service itself (basically something very similar to CORBA IDL, RPC [30] and so on...). This signature is described in a famous language called WSDL and it lacks, as said, a semantic definition.

An alternative methodology to the mapping described above could be to enrich the signature definition of semantic information, i.e. adding to WSDL ontological information (OWL-S or WSMO for example). Generally the better approach would be to enrich this information not the BPEL process itself but the WSDL signature since it is exactly that one the place in which the interface is described and that one would be the place where any kind of information related to the messages should be placed (basically this is information encapsulation). Some proposal exists in this sense like WSDL-S [56].

Additionally to this information we could theoretically add a description of the protocol of interaction of the service (sometimes called a behavioral type) which will be useful for verification issues (we will discuss verification in the next section). This kind of approach has been developed in [4].

Summarizing, what we should need for our discovery mechanism, composition and verification would be a semantically and behaviorally enriched signature - we will call it a *contract* - that need to be composed by:

- An ontology annotates *signature*, i.e. WSDL + ontological information
- A protocol of interaction (*behavioral types*)

where the first would allow to identify the correct services in the discovery phase while the second would be for verification purposes<sup>2</sup>. The disadvantage of this approach is the fact that we need to modify the WSDL structure which is presently well known and used by many already offered services.

## 4 Our Approach

For our work, we have identified a novel connection between the Semantic and the Syntactic worlds. We bring together Semantic Discovery, Semantic Composition, and Process-Level (Syntactic) Validation and Execution; at the interface of the two worlds, we use a simple interface language for parallel and sequential workflows. The overall process is capable of fully automatically turning a BPMO process WSMO-style capability into a concrete, validated, BPEL implementation. The process assumes that the existing BPMO services – those stored in the service repository – each come with a WSMO-style capability description, and a corresponding BPEL implementation.

The process proceeds in two major phases. The first phase acts on the Semantic side. The BPMO capability is interpreted as a WSMO goal. Then, in a first step, Semantic Discovery is used to find the available services, in the repository, which are relevant to that goal. This discovery step is more or less a standard WSMO-style discovery, using semantic matching to identify the relevant services. The second step on the semantic side is then to perform the actual composition. Using the BPMO capability, and the BPMO capabilities of the discovered services, as input, the composition step automatically constructs a sequential workflow of BPMO services that, based on the capability information, is suitable to accomplish the goal. To realize this composition step, we will adapt techniques from AI Planning. In a final step on the Semantic side, the sequential workflow is post-processed into a parallel workflow, by removing unnecessary ordering relations. It is beneficial to do this offline, rather than online during the actual composition step, since parallel composition is known – from AI Planning – to scale very badly. Once the semantic phase has completed, the output is a parallel workflow of BPMO services. This is handed over to the Syntactic side in the form of a simple interface language, allowing for sequential and parallel composition. The BPMO services are, in the interface language, replaced by IDs (URLs or some other method) from which the syntactic side can identify the corresponding BPEL. Technically, the interface language will take a simple XML-based format.

On the syntactic side, the received workflow is “fleshed out” by replacing the service IDs with the corresponding BPEL, and fitting the pieces together, hence obtaining a BPEL implementation of the workflow. Then validation tasks are performed. These will proceed by, first,

---

<sup>2</sup>Theoretically services could also be discovered on a behavioral basis but this is still a very fuzzy direction which goes behind the scope of this work

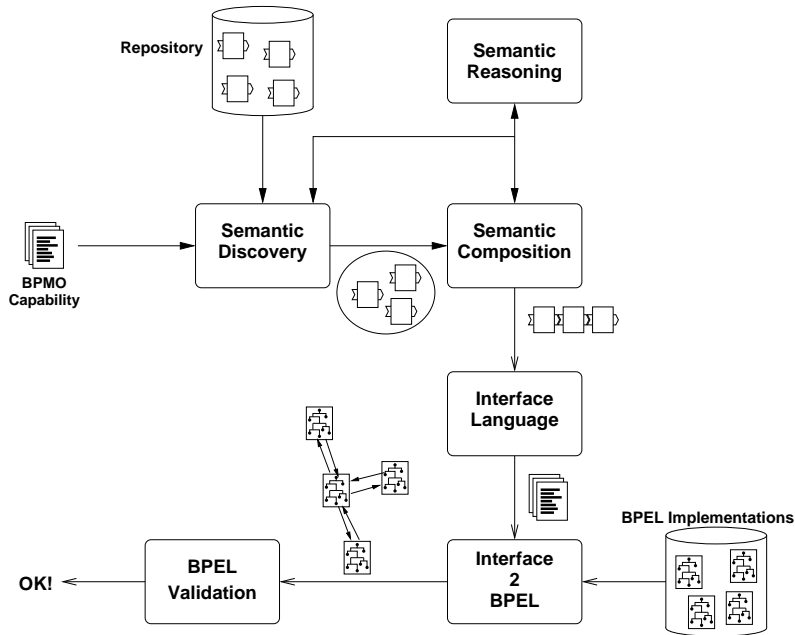


Figure 6: Overview of the different phases

formalising the BPEL in a  $\pi$ -calculus formalism; then, validation tools from that area can be applied. The existing tools and approaches are not sufficient for this purpose, and so these methods will be appropriately extended within SemBiz. The latter is a major research contribution on this side of the overall process.

Upon termination of the process, the final BPEL process is a validated implementation of the original BPMO capability. The complexity of this transformation is suitably distributed over the individual steps in the process chain, hence resulting in a manageable complexity at each single step.

#### 4.1 Semantic Discovery

For automatically locating business processes described using BPMO, we build on the experience gained from designing WSMO Discovery [24]. WSMO Discovery distinguishes between goal discovery, Web service discovery and service discovery. Web service discovery consists of matching the abstract description of the service being sought, with the abstract descriptions of the services being offered, in order to detect which of them can potentially be used to get the desired service. For this, the concrete user input is generalized to more abstract goal descriptions, and the concrete services and their descriptions are abstracted to the classes of services a Web service can provide.

There exist different approaches to discovery, with different accuracies. Syntactic approaches have the advantage that they can quickly filter a possibly large set of available Web services. On the other hand, semantic approaches match Web services based on the capabilities they offer, therefore delivering results of higher precision. The main advantage of the semantic

approaches is that the background knowledge, encoded using ontologies, is also taken into account during the discovery process.

The semantic discovery will first attempt to locate the BPMO capabilities that fully satisfy the user request. Once again, we build on the experience from WSMO Web service discovery, and consider the possible degrees of a match that can occur.

More precisely, from the Web services that fully match a user request, the most relevant are the *exact matches*, meaning that the functionality advertised in the Web service is semantically identical to the one requested by the goal. Less relevant to the request, but more plausible in real world settings, are *plugin matches*. Here, the functionality of the Web service is more general than the functionality of the goal, meaning that the Web service will deliver all the objects which are requested but it may also deliver objects irrelevant for the requester.

If no Web services fully satisfying the goal are found, the discovery engine searches for Web services partially matching the request. *Subsumption matches* are those Web services with a functionality more specific than the one requested. Such Web services are not able to deliver all the objects which are requested, but are guaranteed not to deliver irrelevant objects. *Intersection matches* represent the weakest degree of match, and imply that there exists a commonality between the Web service and goal.

## 4.2 Semantic Composition

The partially matching BPMO capabilities are passed on to the functional level composition, which operates also on the semantic descriptions. Composition at this level does not result into an executable process, rather it identifies the set of capabilities appropriate for composing and their interrelation. This type of composition should therefore be considered an initial step whose purpose is to decrease the complexity prior to process level composition.

At this level, a solution is built by formulating the composition problem as a search in a transition system. Most of the existing approaches use compilations into standard planning formalisms, and therefore distort the problem. Our approach is to address exactly the BPMO service composition, and solve the problem in its natural form.

In order to deal with partial matches, the background ontologies, which encode the behavior of the underlying domain, must be taken into account. Incorporating background theories into the computation of transitions introduces the well known frame and ramification problems, making composition a computationally very hard task. Furthermore, we will allow composition with intersection matches, which is especially hard. The challenge here is thus to identify special cases with simpler semantics and also reduced complexity.

## 4.3 Interface Language

Let us consider the motivating example presented above. To implement this kind of scenario it is necessary to find a way to detect and manage the fact that "USERNUM" and "NUMBER" are basically the same thing. This is exactly the role of semantic Web Services, Ontologies and all the research field in which OWL-S and WSMO are involved.

Indeed, using an approach similar to the one in fig.5 it is possible to identify the set of relevant services suitable to reach the overall goal which should be described in term of some

logic-related language expressive enough to include at least first order logic. It should be also possible to detect that  $WS_1$  and  $WS_2$  can be executed in parallel (not necessarily in sequence) but before  $WS_3$ , while  $WS_3$  need to be executed afterward (basically it is something like a petri net dependency or  $\pi$ -calculus rendez-vous).

The output of the semantic discovery and composition mechanisms - i.e. the process of identifying the suitable processes for the desired goal and getting out some form of "aggregated" service - is in general an intermediate representation of the process looking like, for example:

$$(WS_1|WS_2).WS_3$$

where the intended meaning of this expression would be that you have firstly to execute  $WS_1$  and  $WS_2$  concurrently (in parallel, like UNIX Processes or Java threads) while  $WS_3$  must be executed only afterward.

This is a simple "workflow pattern" that could be represented but, usually, in composition we find further and more complex behaviors that need to be expressed. Taking inspiration from the literature for concurrent systems, from the concurrency theory and from business case studies we suggest to model at least the following behaviors:

- **Sequential Composition:** it contains one or more services that are executed in the order in which they are listed. The sequence completes when the final service in the sequence has completed.
- **Parallel Composition:** the most fundamental semantic effect of grouping a set of services in this way is to enable concurrency. This composition completes when all of the contained services have completed.
- **Conditional Composition:** it is used to perform the execution of one of several branches depending on an external event. A branch is selected if the event associated with it occurs. After it has accepted an event, the other events are no longer accepted.

#### 4.4 From the Interface Language to BPEL

Providing this kind of intermediate representation allows us to get out of this a BPEL abstract process (but it is also possible to imagine any different kind of interesting output, i.e. other composition languages) which can become executable with some additional information (basically WSDL) and where any semantic annotations is no longer available (and useful) at this stage since "consumed" during the first phases of semantic discovery and composition (let us use these terms here although they can result ambiguous because used in many contexts with slightly different meanings).

#### 4.5 BPEL Validation

In the case of BPEL any verification on compositions has to check that the basic services can work together by means of opportune interactions. The focus in OWL-S/WSMO instead is on

verifying whether the particular service has the properties that the client expects in order to use it. In this sense it is not evident how to reuse the results of work on verifying BPEL to the verification of OWL-S. For this reason it would preferable performing the verification step on the BPEL level, gaining experience from the wide literature.

Furthermore, the advantage of this intermediate representation is that it allows also to be mapped to some formal machinery (independent from the source language which could be again not necessarily BPEL) with a well-defined semantics. This would be not directly necessary for the execution but to perform state-of-the-art static verification and analysis on processes, e.g. deadlock-freedom analysis and useless-code elimination [52]. For example, in the case the intermediate representation would be mapped to some  $\pi$ -calculus related language it would be possible to answer to many questions using already developed tools that, for theoretical reasons, cannot be complete but which can anyway be really useful in concrete cases.

A foundational unifying framework based on the  $\pi$ -calculus that could be applied in this sense has been developed in [27] and [29]. It is an orchestration language able to meet composition requirements and to encode the whole BPEL itself. This works together contribute with a powerful and expressive language, with a solid semantics, that allows formal reasoning and processes equivalence proofing. These results can be used both for a better understanding of the BPEL semantics and behavior, in general misunderstood, and for developing conceptual and software tools able to detect process equivalences leading to flow design simplification and orchestration engines and compilers lightening.

#### **4.6 Advantages over Previous Methodologies**

In general with an OWL-S/WSMO to BPEL mapping (or similar) we are limited by the fact that the original process description regards a single service and not a net of services to be composed. Adopting the intermediate representation solution is a way to avoid this issue. Furthermore, the undesired thing concerning the annotation of WSDL was the fact that we needed to modify the WSDL structure which is presently well known and used by many already offered services. With the last solution also this problem seems solved. Finally with the use of an intermediate representation we can easily access to execution (possibly with different frameworks) and verification facilities.

### **5 Conclusions**

This document contributes with an overview on the state of the art for semantic query, composition and discovery. The industry involvement and the raising general interests on the topic allowed the proliferation of many papers and proposal with different levels of maturity since every community or working group is trying to push their stuff into the scenario. The discussion here does not pretend to be exhaustive, a complete synopsis between all the proposals and concepts would make the presentation basically unreadable and just a list of ugly and non understandable acronyms. This was not our purpose, so we decided to classify things and give ideas and relationships between these, at least as far as we understood. This work required a lot of effort, it was really more complicated than simply listing acronyms and Web sites. We wanted to give the general and simple idea of the most relevant activities providing adequate references for the interested reader who wants to go into the details.

Our classification has been based on two different and very important approaches: the syntactic and the semantic. Since we think that it is very important to put the emphasis on commonalities and differences between the two approaches, we also discussed hybrid strategies that try to take the best from both. We believe that this contribution will be very important in trying to make the Web Services and the Semantic Web communities collaborate. Specifically, in the third and fourth sections we described three work methodologies for integrating the two approaches and we presented advantages and disadvantages of each of these. While the first two of these methodologies have been at least partially discussed in literature, the last one can be considered our novel contribution to SemBiz. Sometime the works we presented in section three have been not explicitly contextualized as integrating approaches in literature. One of the really complicated thing of our work has been to find them understanding that they could fit in the bigger picture we wanted to build in this document. In this sense, we believe that our contribution is not merely a review of the start of the art but a first attempt to find hidden connections between heterogeneous works developed in apparently different contexts. In this document we tried also to motivate the need for Semantic Web Services and we coped with the complex issue of trying to give a partial answer to the main objections that usually arise when presenting the advantages of Semantic Web in certain contexts. To do this we discussed the potential advantages we see for banks and insurances.

At the end we hope to have put many complex things in an easier light making them understandable and emphasizing connections that were not explicitly visible before. We hope also to have found, in the third section, a fruitful research direction that can be followed in the next future.

## References

- [1] T. Andrews, F. Curbera et al. *Web Service Business Process Execution Language*, Working Draft, Version 2.0, 1 December 2004.
- [2] G.Booch *Object-Oriented Analysis and Design with Applications*, Addison-Wesley
- [3] S.Burbeck - *The Tao of e-Businness Services*, IBM Corporation, 2000 [[www-4.ibm.com/software/developer/library/ws-tao/index.html](http://www-4.ibm.com/software/developer/library/ws-tao/index.html)]
- [4] S. Carpineti, G. Castagna, C. Laneve et L. Padovani - *A formal account of contracts for Web Services*. In WS-FM, 3rd Int. Workshop on Web Services and Formal Methods, no. 4184, LNCS, pag. 148-162, Springer, 2006.
- [5] E. Christenses, F. Curbera, G. Meredith, S. Weerawarana. *Web Services Description Language (WSDL 1.1)* [[www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl)], W3C, Note 15, 2001.
- [6] Conversion From OWL-S To BPEL Online Demo [<http://www.laits.gmu.edu:8099/OWLS2BPEL/>]
- [7] CORBA - *Common Object Request Broker Architecture* [[www.omg.org/technology/documents/formal/corba](http://www.omg.org/technology/documents/formal/corba)]

- [8] B. Drabble - *EXCALIBUR: A Program for Planning and Reasoning with Processes*, Journal of Artificial Intelligence, vol.62, no. 1, 1993
- [9] P. Durusau, S. Newcomb, R. Barta - *Topic Maps Reference Model*
- [10] L. Garshol - *Q: A model for topic maps*, Proceedings of Extreme Markup Languages, 2005
- [11] L.M. Garsohl - *Tolog - A Topic Map Query Language*, Proceedings of the First International Workshop on Topic Map Research and Applications Germany, October 2005, LNCS, Volume 3873, 2006
- [12] P. Haase, J. Broekstra, A. Eberhart, R. Volz *A comparison of RDF query languages*, Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004
- [13] Michael N. Huhns and Munindar P. Singh. *Service-Oriented Computing: Key Concepts and Principles*. IEEE Internet Computing , January 2005, pages 75-81
- [14] I. Herman - 2004 Presentations: *Towards the Semantic Web* [<http://www.w3.org/2004/Talks/0209-Helsinki-IH/Overview.pdf>]
- [15] *HTTP - HyperText Transfer Protocol Specification* [[www.w3.org/protocols](http://www.w3.org/protocols)].
- [16] *OMG IDL: Details* [[http://www.omg.org/gettingstarted/omg\\_idl.htm](http://www.omg.org/gettingstarted/omg_idl.htm)]
- [17] M. Juric, P. Sarang, B. Mathew - *Business Process Execution Language for Web Services, 2nd Edition*, Packt Publishing.
- [18] N. Kavantzias, D. Burdett, G., Y. Lafon. *Web Services Choreography Description Language Version 1.0*, 12 October 2004.
- [19] R. Khalaf, S. Tai, and S. Weerawarana - *Web Services, the next step: A framework for robust service composition*, CACM, Special Issue on Service-Oriented Computing, M.P. Papazoglou and D. Georgakopoulos, Eds., October 2003.
- [20] N. Kobayashi - *Type-based information flow analysis for the pi-calculus*, Acta Informatica, 42(4-5):291-347, 2005
- [21] N. Kobayashi - *A New Type System for Deadlock-Free Processes*, Proceedings of CONCUR 2006, Springer LNCS 4137, pp.233-247, 2006
- [22] R. Lara, H. Lausen, S. Arroyo, J. de Bruijn and D. Fensel - *Semantic Web Services: Description Requirements and Current Technologies*, In International Workshop on Electronic Commerce, Agents, and Semantic Web Services, 2003
- [23] R. Lara, A. Polleres, H. Lausen, D. Roman, J. de Bruijn, and D. *A Conceptual Comparison between WSMO and OWL-S* [[http://www.wsmo.org/2004/d4/d4.1/v0.1/20050106/d4.1v0.1\\_20050106.pdf](http://www.wsmo.org/2004/d4/d4.1/v0.1/20050106/d4.1v0.1_20050106.pdf)]
- [24] U. Keller, R. Lara, A. Polleres, I. Toma, M. Kiefer, and D. Fensel WSMO discovery. Working Draft D5.1v0.1, WSMO, November 2004. [<http://www.wsmo.org/TR/d5/d5.1/v0.1/>]

- [25] D. Fensel and C. Bussler - The Web Service Modeling Framework WSMF, *Electronic Commerce Research and Applications*, 1(2), 2002.
- [26] F. Leymann - *Web Services Flow Language (WSFL 1.0)* [[www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf](http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf)].
- [27] R. Lucchi and M. Mazzara - *A pi-calculus based semantics for WS-BPEL*, *Journal of Logic and Algebraic Programming*, vol. 70, 2007
- [28] D.J. Mandell and S.A. McIlraith - *Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation*, *Proceedings of the Second International Semantic Web Conference*, 2003
- [29] M. Mazzara and I. Lanese - *Towards a Unifying Theory for Web Services Composition*, *Proceedings of WS-FM'06, 3rd International Workshop on Web Services and Formal Methods*, LNCS vol. 4184. Springer Verlag, 2006
- [30] A. Munro - *Moving To Distributed Processing Standards ... Remote Procedure Call* [<http://www.ja.net/documents/NetworkNews/Issue44/RPC.html>]
- [31] Open Source BPEL Engines [<http://alek.xspaces.org/2005/01/27/free-bpel-engines>]
- [32] *Oracle BPEL Process Manager* [<http://www.oracle.com/technology/bpel/index.html>]
- [33] *OWL - Web Ontology Language* [<http://www.w3.org/2004/OWL/>]
- [34] *OWL-S - Semantic Markup for Web Services* [<http://www.w3.org/Submission/OWL-S/>]
- [35] *OWL Web Ontology Language Overview* [<http://www.w3.org/TR/owl-features/>]
- [36] *OWL-S' Relationship to Selected Other Technologies* [<http://www.w3.org/Submission/OWL-S-related/>]
- [37] *Ontology for Grounding* [<http://www.daml.org/services/owl-s/1.1/Grounding.owl>]
- [38] *Describing Web Services using OWL-S and WSDL* [<http://www.daml.org/services/owl-s/1.0/owl-s-wsdl.html>]
- [39] *RDF - Resource Description Framework* [<http://www.w3.org/RDF/>]
- [40] *RDF Vocabulary Description Language 1.0: RDF Schema* [<http://www.w3.org/TR/rdf-schema/>]
- [41] E. Sirin and B. Parsia - *Planning for Semantic Web Services*, In *Semantic Web Services Workshop at 3rd International Semantic Web Conference*, 2004
- [42] K. Sivashanmugam, K. Verma, A. Sheth, J. Miller - *Adding semantics to Web Services standards*, In *Proceedings of the 1st International Conference on Web Services (ICWS)*, 2003
- [43] M. Paolucci, T. Kawamura, T. Payne, K. Sycara - *Importing the semantic Web in UDDI*, In *Proceedings of E-Services and the Semantic Web Workshop*, 2002

- [44] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In I. Horrocks and J. Handler, editors, 1st Int. Semantic Web Conference (ISWC), pages 333347. Springer Verlag, 2002.
- [45] N. Srinivasan, M. Paolucci, K. Sycara - *Adding OWL-S to UDDI, implementation and throughput*, First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC), 2004
- [46] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In Proceedings of the 12th International Conference on the World Wide Web, Budapest, Hungary, May 2003.
- [47] *SOAP - Simple Object Access Protocol* [[www.w3.org/TR/soap](http://www.w3.org/TR/soap)]
- [48] *SPARQL Query Language for RDF* [<http://www.w3.org/TR/rdf-sparql-query/>]
- [49] C. Szyperski. *Component Software: Beyond Object-Oriented Programming*, Addison Wesley Professional
- [50] *TopicMaps.Org* [<http://www.topicmaps.org/>]
- [51] *Understanding tModel* [<http://www.codeproject.com/soap/understandingTModels.asp>]
- [52] *TyPiCal: Type-based static analyzer for the Pi-Calculus* [<http://www.kb.ecei.tohoku.ac.jp/koba/typical/>]
- [53] *XML - Extensible Markup Language 1.0* [<http://www.uddi.org/>]
- [54] P. Wadler - *XQuery: a typed functional language for querying XML*, Advanced Functional Programming, 2002
- [55] World Wide Web Consortium (W3C) [<http://www.w3.org>]
- [56] *Web Service Semantics - WSDL-S* [<http://www.w3.org/Submission/WSDL-S/>]
- [57] *Web Service Inspection Language* [<http://www.ibm.com/developerworks/webservices/library/ws/wsilspec.html>]
- [58] *WSMO - Web Services Modelling Language* [<http://www.wsmo.org/wsml/>]
- [59] *WSMO - Web Services Modeling Ontology* [<http://www.wsmo.org/>]
- [60] S. Thatte - *XLANG: Web Services for Business Process Design* [[www.gotdotnet.com/team/xml/wsspecs/xlang-c](http://www.gotdotnet.com/team/xml/wsspecs/xlang-c)] Microsoft Corporation, 2001.
- [61] *XML - Extensible Markup Language 1.0* [<http://www.w3.org/TR/2000/REC-xml-20001006>]
- [62] *XML Schema Definition* [<http://www.w3.org/XML/Schema>]
- [63] *XQuery 1.0: An XML Query Language* [<http://www.w3.org/TR/xquery/>]
- [64] Y. Zhao and K. Sandahl - *Potential Advantages of Semantic Web for Internet Commerce*, In proceedings of the 5th International Conference on Enterprise Information Systems, 2003