A unified Semantic Web services architecture based on WSMF and UPML[1]

Jos de Bruijn*

Digital Enterprise Research Institute, Institute for Computer Science,
University of Innsbruck, Technikerstrasse 13, A-6020 Innsbruck,
Austria
Fax: +43-512-507-9872   E-mail: jos.debruijn@deri.org
*Corresponding author

Rubén Lara

Tecnología, Información y Finanzas, Españoleto, 19, 28010 Madrid,
Spain
Fax: +34-91-5200-167   E-mail: rlara@afi.es

Sinuhé Arroyo

Digital Enterprise Research Institute, Institute for Computer Science,
University of Innsbruck, Technikerstrasse 13, A-6020 Innsbruck,
Austria
Fax: +43-512-507-9872   E-mail: sinuhe.arroyo@deri.org

Juan Miguel Gomez

Digital Enterprise Research Institute, National University of Ireland,
Galway, University Road, Galway, Ireland
Fax: +353-91-512541   E-mail: juan.miguel@deri.org

Sung-Kook Han

Wongkwang University, 344-2 Shinyong-dong,
Iksan Chonbuk 570-749, South Korea
E-mail: skhan@wonkwang.ac.kr

Dieter Fensel

Digital Enterprise Research Institute, Institute for Computer Science,
University of Innsbruck, Technikerstrasse 13, A-6020 Innsbruck,
Austria

and

Digital Enterprise Research Institute, National University of Ireland,
Galway, University Road, Galway, Ireland
E-mail: dieter.fensel@deri.org

Abstract: Current semantic web services lack reusability and conceptual separation between services and goals. We propose a unified architecture based on the principles of WSMF and UPML. We introduce goal- and domain-independent web services. Reuse is achieved through the use of bridges and refiners for goal, web service and domain descriptions.

Biographical notes: Jos de Bruijn obtained his MSc degree in computer science at the Delft University of Technology in 2003. He is currently working as a Junior Researcher at the Digital Enterprise Research Institute (DERI) at the University of Innsbruck, Austria. He is Project Leader for the EU-funded SEKT project and a member of the SDK-cluster working groups on semantic web services (WSMO and WSML). His research interests include semantic web services (languages), ontology languages and ontology mediation.

Rubén Lara is R&D director at Tecnología, Información y Finanzas (TIF). He has been the Managing Director of the EU-funded project Knowledge Web (http://knowledgeweb.semanticweb.org). Rubén is a member of the SDK-cluster working group on semantic web services (WSMO and WSML), the most salient European initiative on semantic web services. He obtained his MS in computer science at Universidad Autónoma de Madrid in 2001, and received the First National Award in computer science by the Spanish Ministry of Culture and Science, as well as the Special Award in computer science at the Universidad Autónoma de Madrid.

Sinuhé Arroyo is a PhD Researcher in the area of semantic web and semantic web enhanced web services. He is Project Leader of the EU-funded projects DIP and Esperonto. Previous to his current position he was Project Leader at iSOCO in Madrid, Spain, responsible for the architecture and development of various applications in the field of the B2B applying semantic web technology. He obtained his MS in computer science at Universidad de Malaga, Spain, in 2000. He also holds a BS in software engineering obtained at the University Complutense de Madrid, Spain in 1997.

Juan Miguel Gomez is a Researcher at the Digital Enterprise Research Institute (DERI), where he started his PhD in the Leopold-Frazens Universitat in Innsbruck (Austria) and is completing it at the National University of Ireland, Galway (Ireland). He did his master thesis in software engineering in the Swiss Federal Institute of Technology (EPFL) in Lausanne (Switzerland) and studied his Msc in telecommunications engineering in the Universidad Politécnica de Madrid (UPM). He has been working in the SWWS project and his current research interests include the semantic web, semantic web services, eBusiness, business process modelling and formal methods for eCommerce.

Sung-Kook Han received his PhD degree in information science from Inha University, Korea. He has been a Professor in the Computer Engineering Department at Won Kwang University since 1984. He was a Visiting Researcher of DERI, Innsbruck, and he is currently a member of Semantic Web Forum Korea and Regional Director of the Korea Information Society. His research interests include semantic web, semantic web services, ontology and knowledge management systems.

Professor Dr Dieter Fensel (1960) has a Chair at the University of Innsbruck, Austria and in 2003 he became the Scientific Director of the Digital Enterprise Research Institute (DERI) at the National University of Ireland, Galway. His

current research interests include ontologies, semantic web, web services, knowledge management, enterprise application integration and electronic commerce. He has published around 150 papers as books and journal, book, conference and workshop contributions. He is co-author of several books in the areas of information integration, ontologies and the semantic web. He coorganised around 150 scientific workshops and conferences and is Associate Editor of several scientific journals.

1   Introduction

The World Wide Web has caused a paradigm change in the way people access information. However, the current web presents mainly unstructured information, which is only readable by a machine. To enable new intelligent applications such as content-based search, information brokering, and electronic commerce among several independent and heterogeneous partners, there is an emerging awareness that information on the web needs to be semantically annotated in order to assure machine-processability. This vision is called the semantic web (Berners-Lee et al., 2001), in which resources are explicitly annotated through the use of machine-processable semantic meta-data, enabling the development of new applications.

Nevertheless, the semantic web is only concerned with the provision of static content. Web services promise to lift the web to a new level by adding functionality to the web and realising the computer as a truly computational device. However, current web service technologies around SOAP[2], WSDL[3] and UDDI[4] are not sufficient to build a distributed, heterogeneous web service infrastructure, because they provide limited support for automation of service recognition, configuration and combination. Several approaches, extending these existing standards, have appeared.

The two main areas of extensions are in the area of adding explicit semantics to services; DAML-S (Ankolenkar et al., 2001) is in this context an initiative, which aims to provide explicit semantics for services; another extension, to which much attention goes, is the integration of web services in process modelling. Examples are BPML/WSCI (Arkin, 2003; Arkin et al., 2002), which tries to define business processes and distributed process orchestration, and BPEL4WS (Akkiraju et al., 2003), which models business processes and their interoperation. Because these approaches are focused on specific aspects of web services, an integrated approach is required to enable automatic location, composition, invocation, and interoperation of web services. Semantic web technologies are suitable to transform current web services into services supporting such automation. The resulting services, enriched semantically, are usually called semantic web services.

In this context, the web service modelling framework (WSMF) (Fensel and Bussler, 2002) is a fully-fledged framework for describing the various aspects related to semantic web services. While current web service technologies focus on specific parts of the different needs to realise semantic web services, WSMF addresses the overall problem, building on, adapting and applying semantic web technologies to provide rich descriptions of web services, enabling automatic discovery, composition, invocation and mediation of web services to enable communication between heterogeneous web services. Although WSMF provides a conceptual model for developing and describing semantic web services to allow their discovery, composition and interoperation, it remains to define in detail the

operational mechanisms. Reusability of web services is limited in WSMF because of the seemingly tight coupling between goals and web services (a web service contains an explicit link to a goal) and the domain-dependent description of web services.

To achieve reusability in web services, we consider some concepts related to the research area of problem-solving methods (PSM). Problem-solving methods are generic descriptions of reasoning processes and provide reusable architectures and components for implementing the reasoning part of knowledge-based systems (Benjamins, 1997; Fensel, 2000). The Unified Problem-Solving Method development Language (UPML) (Fensel et al., 2003) has been developed to describe and implement such architectures and components, facilitating their semi-automatic reuse and adaptation. Problem-solving methods presents a framework for developing knowledge-intensive reasoning systems based on libraries of generic problem-solving components. UPML is an architectural description language specialised for a specific type of systems providing components, adapters and their connection configuration. In order to realise an architecture of truly reusable web services, we propose to extend WSMF with notions developed in the area of problem-solving methods. Furthermore, we propose a conceptual architecture based on the IBROW Intelligent Broker architecture (Benjamins, 1997; Benjamins et al., 1998; Fensel, 1997a; Fensel and Benjamins, 1998b), which enables knowledge-component reuse over the World Wide Web, in order to make web service *Discovery*, *Composition*, and *Execution* a reality.

The paper is structured as follows: Section 2 introduces semantic web services, their major issues and web services modelling framework, WSMF. In Section 3 we provide an overview of problem-solving methods and its description language UPML. In Section 4, we identify extensions to WSMF based on concepts found in UPML. Related work is highlighted in Section 5. Finally, Section 6 presents conclusions and future work.

2   Semantic web services and WSMF

Current web vision and technologies are limited to the publishing, rudimentary retrieval and rendering of unstructured content. A web user can make any information publicly available and access a huge amount of content distributed over the web, but the web remains mainly static and the access to dynamic functionality is very limited. The semantic web adds machine-interpretable annotations to web content in order to provide intelligent access to heterogeneous, distributed information. However, the semantic web vision is still primarily about publishing and retrieving static content (Bryson et al., 2002).

In this context, the concept of web services has been defined and developed. Web services can be defined as software objects that can be assembled over the internet using standard protocols (Fensel and Bussler, 2002) to provide functionality rather than content. web services extend the web from a distributed source of information to a distributed source of service. Nevertheless, current technologies around web services, mainly SOAP, WSDL and UDDI, lack key features to support automation in location, combination and use of the functionalities provided by published web services.

To overcome these limitations, semantic web concepts are applied not only to the current web, but also to current web services, in order to minimise the human interaction required to use web services. The contribution of the semantic web to web services is twofold (Paolucci et al., 2002a). Firstly, it provides a shared conceptualisation and

terminology of the messages exchanged between different services. Secondly, ontologies provide explicit semantics that enable enhanced web services descriptions. These descriptions are not only machine-readable, but also machine-understandable and thus usable for web service automation. In this way, semantic web concepts are used to define intelligent web services, i.e. services supporting automatic discovery, composition and use. The joint application of semantic web concepts and web services to create intelligent web services has lead to what are usually called semantic web services (Lara et al., 2003; McIlraith et al., 2001).

Semantic web services have a big potential impact in areas like knowledge management, enterprise application integration (EAI) and e-Commerce, due to their capability to provide dynamic distributed computation, both within and across company boundaries. However, to make this potential impact a reality, several major challenges appear. One of the efforts aiming to overcome these challenges is the web services modelling framework (WSMF), which provides a service description framework to enable web services automation.

The major challenges that have to be faced to realise semantic web services are presented in Section 2.1. WSMF will be introduced in Section 2.2, which presents its main principles, and in Section 2.3, where its major components are related.

2.1   *Major challenges in semantic web services*

The key to dynamic and distributed computation is on-the-fly software creation through the use of loosely coupled, reusable software components, that is, web services (Fensel and Bussler, 2002). Such an automatic functionality provision cannot be possible without a strong automation support for the different steps involved in this process, namely: location of suitable services, combination (if required) of different services to provide the requested functionality, invocation of the selected (and composed) services, and interoperation between these services.

This scenario provides a set of major issues to make the semantic web services' vision a reality:

- *Automatic web service discovery*. Automatic web service discovery involves automatically locating web services that provide a particular functionality and that adhere to requested properties (McIlraith et al., 2001) expressed as a user goal, e.g. finding a service to buy a flight ticket between two given cities and which accepts a given credit card. To provide such automatic location, the discovery process should be based on the semantic match between a declarative description of the goal being sought, and a description of the service being offered. This problem requires not only an algorithm to match these descriptions, but also a language to declaratively express the capabilities of services (Paolucci et al., 2002a).

- *Automatic web service composition*. Service composition can be described as the combination of several services to provide a given functionality, e.g. combining a flight search and a flight book service to provide a flight ticket to the requester based on his criteria. Composition of web services requires something more than representing combinations of services where flow and bindings to the services are known a priori; it must allow for the combination of services to provide a given functionality when a request can not be fulfilled by using available services individually (Sirin et al., 2003).

● *Automatic web service execution*. Automatic web service execution involves the automatic invocation of an identified web service by a computer program or agent (The DAML services coalition, 2003). It consists of two different aspects. Firstly, details about the *grounding* technologies to be used to invoke the service and about the concrete invocation parameters must be available at run-time for the service requester in order to perform a real invocation. Secondly, as one of the main purposes of web services is the automation of application integration inside and across (i.e. e-Commerce) organisational boundaries, there is a need to support *interoperation* between different services, either between services in an organisation or between services in different organisations. In order to allow for this interoperation, the external behaviour of the service in terms of message interchange must be made public, and it has to be described using formal, explicit semantics.

In a nutshell, the major issues in semantic web services are related to the different steps involved in providing the functionality dynamically and automatically. These steps impose certain requirements on the description of semantic web services and on the architecture supporting such services.

In this context, the web service modelling framework (WSMF) (Fensel and Bussler, 2002) aims at providing an appropriate conceptual model for developing and describing services and their composition in order to support the major issues presented above. Its main principles are explained in the next section; its main components are presented in Section 2.3.

2.2   Principles of WSMF

The way a service is described determines to what extent automation can be realised by the different components of a given architecture. The goal of WSMF is to provide a fully-fledged description framework for web services to enable the required automation.

WSMF is based on two complementary principles, around which the design of the framework is organised (Fensel and Bussler, 2002):

● *Strong de-coupling* of its various components. This de-coupling includes information hiding based on the difference between internal business intelligence and public message exchange protocol interface descriptions (see Bussler (2001)).

● *Strong mediation* enabling anybody to speak with anybody in a scalable manner.

The first WSMF principle is twofold: firstly, it is concerned with the de-coupling of the different components needed to realise semantic web services; secondly, it is concerned with the difference between public service information and the internals of the service. This principle is a key feature to enable reuse of components in the architecture we envision in this paper.

The second principle overcomes the unrealistic assumption that every party in a wide and distributed environment uses the same terminology and interaction model. It includes the mediation of different terminologies (Fensel, 2003) as well as the mediation of different interaction styles (Bussler, 2001), thereby enabling communication between heterogeneous parties in an open and highly distributed environment like the web.

154   *de Bruijn et al.*

2.3   Major components of WSMF

WSMF consists of four main elements to provide appropriate service descriptions (Fensel and Bussler, 2002):

- ontologies, which provide the terminology used by other elements

- goal repositories, which define the problems that should be solved by web services

- web services descriptions, which describe various aspects of a web service

- mediators which solve interoperability problems.

A more detailed explanation of each element is given below.

2.3.1   Ontologies (Fensel, 2003)

Ontologies interweave human understanding of symbols with their machine-processability, gluing together two essential aspects which are exploited within WSMF, namely:

- ontologies provide a formal semantics for information, consequently allowing information processing by a computer

- ontologies provide a real-world semantics, which make it possible to link machine-processable content with meaning for humans based on consensual terminologies.

Ontologies provide the terminology that is used by other elements of WSMF specifications, with the purpose of enabling reuse of terminology as well as interoperability between components referring to the same or to linked terminologies.

2.3.2   Goal repositories

Goal repositories store requester goals. A goal specifies the objective a given requester might have when looking for a service. A goal specification consists of two elements:

- *Pre-conditions* describe what a web service expects for enabling it to provide its service, i.e. the requirements over the input. For example, a possible pre-condition to withdraw money from your bank account is to have money in the account.

- *Post-conditions* describe what a web service returns in response to its input, i.e. the relationship between the input and output. For instance, a book price would be the post-condition of a hypothetical book search service.

Goal definitions are kept separate from concrete web service descriptions in WSMF, in order to allow an *n2m* mapping between goals and actual web services. The reason for doing so is that a given service can fulfill different goals and, furthermore, different services can fulfil the same goal.

2.3.3   Web services descriptions

These give details about concrete web services. The framework describes them as black boxes, i.e. it does not model the internal aspects of how the service works, hiding private business logic aspects. This concrete service description relies on the previous two elements: ontologies provide the necessary terminology, and goal repositories express (more generic) goals that are fulfilled by the service.

The main elements in a WSMF web service description are the following:

- *Web service name*, i.e. a unique identifier for the service.

- *Goal reference*, which provides a link to the goal, which stores information about the purpose of the service.

- *Pre- and post-conditions*, which can be linked directly or indirectly to the pre- and post-conditions in a goal description. In this way, service pre- and post-conditions can respectively strengthen goal pre-conditions or weaken goal post-conditions. It is possible to use different terminologies for goals and service descriptions. This would, however, required mediation between these different terminologies.

- *Input* and *output* ports. Input data are passed to the service using the input port, while the output data are returned through the output port. The structure of the input and the output data is described in a formal way. The input and output ports allow for concurrently passing data to the service during the execution and for the provider to concurrently make results available to the requester without waiting for the service to have finished execution.

In addition to this basic web service description, WSMF considers other elements to describe services, always under the assumption that internal business logic is hidden. These elements include exception handling, compensation support and non-functional properties (e.g. geographical information and Quality of Service properties).

2.3.4   Mediation

Mediation is the process of enabling heterogeneous parties to automatically inter-operate. In heterogeneous contexts like the web, different services may use different terminologies and different interaction styles. Thus, mediation must be introduced to enable every service to communicate with any other. Several kinds of mediation are identified, namely:

- Vocabularies have to be mediated through ontology mapping (cf. Fensel, 2003; Klein, 2001), which ensures that heterogeneous web services can be invoked and composed.

- Process mediation deals with different interaction styles and conversation patterns.

- Mediation of message exchange protocols overcomes the differences in underlying messaging protocols between the communicating parties. Nevertheless, this type of mediation is out of the scope of this paper.

- Dynamic service invocation, which takes care of providing the invocation of appropriate services at run-time to fulfil a declarative goal specified by the requester at design-time (Fensel and Bussler, 2002).

The existence of mediators to overcome the inherent heterogeneous nature of web services requires the published services to expose their conversational interfaces using explicit semantics to allow data and process mediation. Data mediation is achieved by using ontologies to make the content of the exchanged messages understandable; dynamic service invocation is achieved by using web service proxies, which consist of a goal reference or a specification of pre- and post-conditions. These machine-understandable descriptions are used by the mediation element to allow for the successful communication between heterogeneous agents and services.

Together, the four elements presented in this section constitute the basic building blocks of the web services modelling framework. In our proposed unified architecture, we extend this framework to support the reuse of web services, relying on the strong decoupling of components and strong mediation between the components as introduced in WSMF. Notions from UPML and from the IBROW Intelligent Broker architecture will be used together with WSMF to define a unified conceptual architecture of truly reusable web services, as will be described in Section 4.

3   Problem-solving methods and UPML

Problem-solving methods (PSM) provide a reusable architectures and components for implementing the reasoning part of knowledge-based systems based on reusable libraries of generic problem-solving components (called *problem-solving methods*). The Unified Problem-Solving Method development Language (UPML) is an architectural description language for developing such architectures and components, facilitating their semi-automatic reuse and adaptation.

The following section provides an introduction to Problem-Solving Methods (PSM) and its relation with UPML. Section 3.1 presents a sketch of problem-solving methods; Section 3.2 introduces the UPML architectural components, and its origins within the IBROW project.

3.1   Problem-solving methods

Problem-solving methods describe reasoning components as patterns of behaviour that can be reused across applications. They are reusable building blocks that can be readily applied independently from the domain and implementation aspects (Benjamins, 1997; Fensel, 2000). Such building blocks provide an abstraction layer from different sources of 'noise': *implementation aspects*, *domain aspects* and *task aspects*.

Problem-solving methods decompose the reasoning task in a number of subtasks and elementary inference actions to solve each of the particular tasks. Methods describe *how* the goal of a particular task can be achieved. In order to realise their functionality, a way to express restrictions on the context of the problem-solving method is required. *Assumptions* provide such expressiveness by constraining the context of the problem-solving method. Assumptions restrict the functionality required from the problem-solving method (i.e. *weaken the goal*) or to reduce the complexity of a method by *strengthening the knowledge requirements* (Fensel and Benjamins, 1998a; Fensel and Straatman, 1998). They enable tractable problem-solving and economic system development of complex problems, by reducing the worst-case, or average-case complexity of computation, and decreasing the cost of the system development process by simplifying the problem that must be solved. And finally, they ensure a proper interaction of the problem solver with its environment (Fensel, 2000).

To enable the reuse of problem-solving methods, it is necessary to abstract the reasoning patterns from their application domain, however, depending on the domain knowledge, different methods may impose different assumptions over this knowledge that are not necessarily satisfied by all application domains. Hence, the application of a particular problem-solving method to a domain requires adaptation through explicitly specified adapters. Adapters bridge the gap between the requirements specified by a method's assumptions and the knowledge provided by the knowledge base of the application domain (Fensel and Motta, 2001).

3.2 UPML

The *Unified-Solving Method development Language* UPML (Fensel et al., 2003) presents a human-understandable high-level description language, to glue the different elements that make up the Knowledge-Based System (KBS)'s architecture. UPML provides *components*, *adapters*, *configuration* describing how the components should be connected using adapters, and *design guidelines* describing how to construct a Knowledge-Based System using the components and connectors that satisfy the architectural constraints defined by adapters (Fensel et al., 2003).

UPML was developed within the IBROW3[5] project (Benjamins et al., 1998; Fensel, 1997a), to allow for the semi-automatic reuse of problem-solving methods, by integrating PSM libraries over the internet. In the context of the IBROW3 project, an Intelligent Broker (IB) architecture was developed, enabling the location and composition of problem-solving methods, capable of obtaining knowledge components from the web, arranging them in a particular way and then solve the user's problem according to the stated requirements. The goal of the project was to enable the semi-automatic reuse of PSM libraries available over the internet.

3.2.1 The UPML architecture

The UPML architecture consists of the following components:

- *Task* defines the task that should be solved by the problem-solving method. A task specification can be viewed as a two parts concept. On one hand are the *goals* to fulfill, in the other *assumptions* about the domain knowledge and *preconditions* over the input. Goals specify objectives that should be fulfilled in order to provide a solution for a concrete problem. Preconditions are conditions on dynamic inputs, while assumptions are conditions on knowledge consulted by the reasoner that restrict the valid inputs. Assumptions ensure that the task can always be solved for a legal input (input for which the precondition holds) and often can be checked in advanced while preconditions cannot (Fensel et al., 2003).

- *Problem-solving methods* define the reasoning process and the types of knowledge necessary to solve a particular task. Problem-solving methods provide a decomposition of the task into more elementary sub-tasks, and the control flow between sub-tasks and elementary inference actions. The *competence* of a problem-solving method is specified independently from its operational behaviour, which allows verification that it fulfils a concrete task independent of its internal reasoning details. There are two types of problem-solving methods as described in UPML (Fensel et al., 2003), namely:

  - *problem decomposer* – decomposes a task to be solved into a set of sub-tasks

  - *reasoning resource* – solves a subtask of a problem provided by the problem decomposer. A reasoning resource performs a primitive reasoning step.

- *The domain model* provides *domain knowledge* as it is required by the problem-solving method and the definition of the task. A *domain* is made of three elements: a meta-level characterisation of properties of the domain, assumptions over the domain model, and the domain knowledge itself. Domain knowledge plays a pivotal role; it allows the defining of a task in a concrete application domain, and provides the means to carry out the steps of a concrete problem-solving method.

- *Ontologies* (Fensel, 2003; Gruber, 1993) provide the terminologies used in the definitions of the tasks, problem-solving methods and domain definitions. These elements are described in an independent fashion to enable their reuse across different domains, tasks and domains, and tasks and problem solving methods respectively.

- *Adapters* allow the independent definition of the task, problem-solving methods and domain models. Adapters are used to specify the relationships between the parts of the specification with respect to each other in a way that meets a concrete application problem. Adapters provide a means to relate the competence of a problem-solving method with a task and a domain model. There are two kinds of adapters:

  - *Bridges* model the relationships between two different components of the PSM formalism (e.g. between a domain and a task or a task and a problem solving method).

  - *Refiners* articulate the *refinement* of tasks, problem-solving methods and domain models (Fensel, 1997b; Fensel and Motta, 2001). They are used to achieve a stepwise adaptation through the application of sequences of refiners to the tasks, problem-solving methods and domain models.

  A refiner is also referred to as a one-dimensional adapter, while a bridge is the same as a two-dimensional adapter. Fensel and Motta (2001) describes a three-dimensional navigation space where the task, the problem-solving method and the domain model constitute the dimensions. Applying a one-dimensional adapter (refiner) constitutes of a step in one dimension in the space. Applying a two-dimensional adapter (bridge) constitutes a two-dimensional step in this space.

In this section we have provided a short introduction to the area of problem-solving methods and its architectural description language UPML. Problem-solving methods define reusable building blocks applicable independently from domain and implementation aspects that describe the reasoning components as patterns of behaviour allowing the decomposition of tasks into sub-tasks that can be readily solved. UPML is an architectural description language, aiming to describe heuristic reasoning components in a task- and domain-independent way through the use of ontologies to enable reuse across applications. We envision reusing concepts of UPML in the context of the web services modelling framework to enable the reuse of semantic web services.

4   A unified architecture for semantic web services

In order to support the reuse of web services, we propose to extend the web services modelling framework (WSMF) (Fensel and Bussler, 2002) with concepts taken from the area of problem-solving methods (Fensel, 2000), especially the Unified Problem-solving Methods development Language UPML (Fensel et al., 2003). The main extensions we propose are the introduction of (teleological) assumptions in goals and (ontological) assumptions (Fensel and Benjamins, 1998a; Fensel and Straatman, 1998) in web service descriptions. These two types of assumptions are further explained in Section 4.1.

We see a resemblance between the functional descriptions of web services and goals in WSMF on the one hand and problem-solving methods and tasks in UPML on the other. Table 1 provides an overview of the (functional) features currently present in WSMF with their counterpart in UPML and the features that are in UPML, but are, in our opinion, currently lacking in WSMF. Our proposed architecture, based on UPML, can be seen in Figure 1.

**Table 1**   Comparison of features in WSMF and UPML

Feature	*Sub-feature*	*WSMF*	*UPML*
Service competence description		X	X
	Name	X	X
	Pre-condition	X	X
	Post-condition	X	X
	Assumptions		X
Goal/task		X	X
	Name	X	X
	Pre-condition	X	X
	Post-condition	X	X
	Assumptions		X
Domain			X
	Assumptions		X
Bridges		(X)*	X
Refiners			X

Note: *WSMF has the concept of 'mediators', which bridge the gap between goals and web services and requesters and providers, respectively. However, WSMF lacks an explicit specification of what these mediators should look like.

In describing web services, we distinguish three kinds of descriptions. We distinguish the goal, the web service, and the domain descriptions. We argue that these descriptions should ideally be kept independent, even though there are some difficulties in maintaining independence, as explained in Section 4.2.2.

This independence allows us to specify refiners separately for the different components. Web services, goals and domains are structured in a process of stepwise refinement into a refinement hierarchy. Generic domain, goals and web service descriptions are refined into more specific descriptions using a process of step-wise refinement (cf. Fensel, 1997b). The refinement of descriptions provides an opportunity for structuring the different architectural components and for enabling reuse of the more generic descriptions.

To provide connections between the independently specified components, we introduce the concept of bridges and identify three kinds of bridges, connecting goals, web services, and domain descriptions.

160   *de Bruijn et al.*

Figure 1   Proposed architecture, based on the UPML architecture presented in Fensel et al.
(2003)
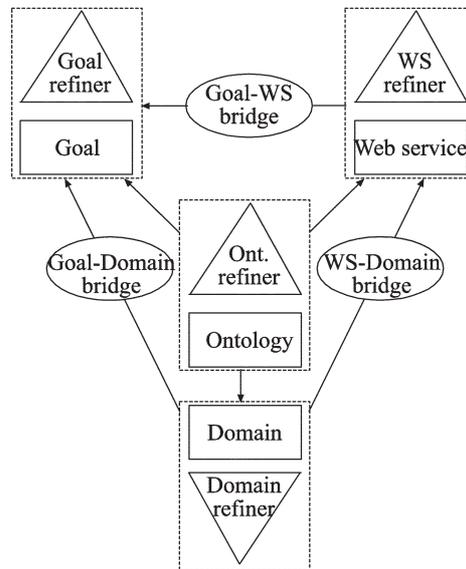
We furthermore propose to structure web service descriptions in two dimensions within
web service libraries. Web services are decomposed in a web service-goal decomposition
view; each web service is decomposed into sub-goals or *Web Service Proxies*. The second
dimension is the refinement dimension, in which refiners are used to structure the
descriptions of web services, goals and domains.

   In Section 4.1 we first discuss the role of assumptions in web services. We will then
provide detailed descriptions of web services, goals, and domains and the role of bridges
and refiners in the architecture in Section 4.2. We then describe the structure of web
service libraries in more detail in Section 4.3. Finally, we identify some limitations of our
approach in Section 4.4.

4.1   *The two-fold role of assumptions in web services*

Assumptions play an important role in the area of problem-solving methods (Fensel, 2000;
Fensel and Benjamins, 1998a). They are introduced mainly to gain efficiency (Fensel and
Straatman, 1998) in problem solving; by making assumptions, a problem-solving method
does not have to search the entire solution space, but only a (by the assumptions) limited
part of it. For example, in the area of parametric design, the entire design space is limited
by making assumptions, thereby reducing the complexity required from the problem-solving
method (Fensel, 2000). In problem-solving methods, there are two kinds of assumptions:
teleological and ontological assumptions (Fensel and Benjamins, 1998a). The former are
assumptions that weaken the goal to be achieved, while the latter specify requirements on
domain knowledge. Therefore, a problem can be reduced in complexity by either lowering
the requirements on the functionality (via teleological assumptions) or raising the
requirements on the domain knowledge (via ontological assumptions) so that part of the
complexity of the problem-solving method is transferred to the domain.

We propose to use these two types of assumptions for web services in order to facilitate automation of the tasks of web service *discovery*, *composition*, and *execution*. Teleological assumptions state what can be assumed to hold about the state of the world at the time of execution of the web service and, therefore, in fact weaken the functionality required from the web service, by restricting the states of the world for which execution is required. Ontological assumptions state knowledge requirements, in the sense of knowledge of the state of the world, for the web service to be able to execute. Note here that the knowledge necessary to meet the requirements stated in the assumption typically originates from the requesting agent or from another web service.

Assumptions made in the web service description must either be fulfilled in the goal description or in the domain description. In the former case, the assumptions are teleological assumptions (making assumptions on the goal); in the latter case, the assumptions are ontological assumptions (making assumptions on the domain knowledge), where part of the complexity of the service is replaced by complexity in the domain knowledge.

An example of a teleological assumption in a 'Buying' web service is that it is assumed that the invoker of the service only wants to buy a product from a particular vendor; this assumption should be reflected in the user goal. An example of an ontological assumption in a 'Buying' web service is the assumption that the requested product is in the product catalogue; here, the product catalogue would have to be part of the domain knowledge.

Assumptions have their limitations if the web service is operating in an open environment and non-trusted parties are allowed to invoke the service, since the (non-trusted) invoker is responsible for making sure that the assumptions hold. An example assumption is the requirement that a user is logged in. The provider of the service wants to be sure that a user is logged in. A solution would be for the provider to create a new composed web service that first invokes a login service and then the web service in question. The provider can now choose to make only the newly created composed web service available to the public. The assumption-checking has now been brought back to the closed environment; to the public, only the pre- and post-condition of the newly created composite service are exposed. The post-condition would state that unless a valid login is provided, the web service will not provide the function.

In WSMF, weakening of the goal and strengthening of the knowledge requirements is done by weakening the post-condition and strengthening the pre-condition, respectively. The web service description strengthens the pre-condition and weakens the post-condition that have been put forward in the goal description. In order for the requester of the service, who only knows the original goal description, to invoke the service, an (as yet undefined) process of mediation is required. This mediator must account for the functionality described in the goal that is not provided by the service. We propose to describe this gap in functionality explicitly in a bridge connecting the goal and the web service descriptions.

In the next section we describe how the above described assumptions are used in the various components of our architecture.

4.2   Describing web services, goals, and domains

Our proposed architecture consists of three main descriptive components, which are used in the discovery, composition and execution of web services. These descriptive components extend the goal and web service descriptions introduced in the web service

162   *de Bruijn et al.*
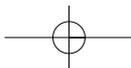
modelling framework. We furthermore introduce the independent concept of domain descriptions in web services. We cover here only the functional description of web services needed for discovery, composition and invocation.

We identify the following elements for the description of reusable web services, necessary for the automation of the discovery, selection, composition and invocation tasks:

- goal descriptions
- web service (capability) descriptions
- domain descriptions.

These descriptions are ideally, but not necessarily, independent. The independence of these components facilitates reuse of web services, goals and domains.

Key components in enabling the reuse of web services are bridges and refiners. Refiners (cf. refiners in UPML (Fensel et al., 2003)) allow for the step-wise refinement of (generic) goals, web service descriptions, and domains, thereby structuring the description space. Bridges (cf. bridges in UPML (Fensel et al., 2003)) allow the use of independently described web services, goals and domains, by providing connections between the descriptions, enabling mediation based on declarative specifications.

4.2.1   Goal descriptions

As in WSMF (Fensel and Bussler, 2002), a goal is described using a name, (an identification for the goal), and pre- and post-conditions; we add one component to the goal description, namely (teleological) assumptions, as described above. We furthermore add input and output roles in order to be able to specify pre- and post-conditions, since pre-conditions specify requirements over the input, while post-conditions describe the relationship between input and output (Fensel and Bussler, 2002). The assumptions specify what must hold for the state of the world for a web service to fulfil the goal. The teleological assumptions weaken the functionality required from the web service.

The goal is the required functionality of a web service from the point of view of the user. The assumptions introduced here are necessary to enable an unambiguous description of the goal; the assumptions say here under what circumstances the web service should execute. In this way, it reduces the complexity of the service required to fulfil this goal. Assumptions can, in fact, be seen as concessions the user is willing to make on the functionality of the service. An example assumption we could add to the (refined) goal of 'Buying' is that it is sufficient for the user if he/she can only buy the requested products at one particular vendor. Note that a web service which allows purchasing from different vendors still fulfils this goal.

The following are the essential functional properties contained in a goal description:

- *Name*. Each goal is uniquely identified by its name.
- *Goal ontology*. The goal ontology provides the vocabulary for specifying the pre- and post-condition and assumptions in the goal description. Furthermore, concepts from the ontology are used to specify the structure of input and output data. Through shared ontologies and/or explicit mappings between ontologies, these ontologies enable inter-operation between parties.

- *Input and output roles*. A goal description has one or more input and one or more output roles. These roles are necessary for the specification of pre- and post-conditions. The structure of input and output data is specified through links to ontology concepts.

- *Pre-condition*. The pre-condition specifies requirements on the input provided by the user.[6] For example, it would only make sense for a user to buy a product, if product description is provided.

- *Post-condition*. The post-condition describes the relationship between the input and the output. For example, when buying a product, the user would expect the product to be shipped to the user.

- *Assumptions*. Assumptions state what can be assumed to hold for the state of the world when the web service is executed. These assumptions weaken the goal, in the sense that the (from the web service) desired functionality is less than specified by the pre- and post-conditions; the problem space is reduced by the assumptions.

The pre-condition and the post-condition (together with the input and output roles) provide a declarative description of the functionality required by the user. The assumptions state that further requirements on the state of the world for the web service to behave as described by the pre- and post-condition. See Figure 2 for an example goal description.

Figure 2   Description of goal – GoodsExchange

goal
    name GoodsExchange
    pragmatics
      The goal of exchanging goods;
      2003-12-29;
    import ontology GoodsExchange
    roles
      note here that the input and output roles are all instances of ontology concepts; the colon here
    (':') indicates the instance-of relationship
        input gd:GoodsDescription; da:DeliveryAddress; rg:ReturnGoodsDescription;
        output de:Delivery;
    pre-condition
      all inputs are required
    assumptions
    post-condition
      if the described good is in stock, the good will be delivered and the return good (described
      with rg) will be delivered in return

In order to enable the reuse of goal descriptions, we propose to describe generic goals and to specialise these generic goals using refiners (Section 4.2.4), thereby structuring the description space of goals, which facilitates reuse and helps in discovering appropriate goals. Figure 3 describes a refiner for the generic 'GoodsExchange' goal of Figure 2 with the name 'Buying'. In the 'Buying' refinement the goods returned must consist of money and an assumption is introduced, which states that the payment information provided by the requester is valid. Note that these more specific goals are not completely described, but that a refiner is used to specify a new (virtual) goal from a more generic goal. This enables stepwise refinement of goals, where refiners are built on top of refiners, and reuse of the same generic goal for several more specific goals.

164   *de Bruijn et al.*

Figure 3   Goal-refiner – Buying

goal-refiner
   name Buying
   pragmatics
     Refines the goal GoodsExchange into the goal of Buying, where there is one producing and
     one consuming party;
     2003-12-29;
   import goal GoodsExchange
   import ontology Buying
   roles
     The input rg must not only be an instance of ReturnGoodsDescription, but also of
     PaymentInformation, which is a concept in the Buying ontology
       input rg:PaymentInformation;
       output
   pre-condition
   assumptions
     Payment information (rg) is valid
   post-condition

Goals should be described in a domain-independent manner and links between goals and domains should be created using goal-domain bridges (Section 4.2.5). These bridges, together with the corresponding goal and domain, create virtual domain-dependent goals.

4.2.2   Web service descriptions

Goal descriptions and web service descriptions correspond in their appearance. They both consist of a name, input/output roles, pre- and post-conditions, and assumptions. The difference between the two is that a goal is user-orientated (i.e. from the point of view of the requester) and a capability is web service orientated (i.e. from the point of view of the provider). A goal describes what a user wants to achieve, where a web service description states what a web service is capable of doing. Because of their conceptual difference and in order to enable reuse, goals and web services should be described in an independent manner (cf. task-independent problem-solving methods (Beys et al., 1996). In this way, web services can be reused across different goals. We recognise, however, that goal-independence is very hard to achieve for web services and that matching of goal and web service descriptions is a computationally hard task and would become an intractable problem for large volumes of web service descriptions. In order to mitigate this problem and because we have realised that it does not make sense to describe goals when there are no web services fulfilling it, we expect bridges to be present, which contain explicit syntactical references to goals and web service descriptions. These bridges help in discovering the web service for a goal through explicit mapping.

    The following are the main functional components of a web service (capability) description:

- *Name*. Each web service is uniquely identified by its name. This name can be a URI or, better, an IRI (International Resource Identifier).[7]

- *Web service ontology*. The web service ontology provides the vocabulary for specifying the pre- and post-condition and assumptions in the web service description, as well as the concepts for specifying the structure of input and output data.

- *Input and output roles*. The input and output roles are used for providing input to the web service and for returning output. The structure of input and output data is specified through links to ontology concepts.

- *Pre-condition*. The pre-condition specifies requirements over the input for the web service to be able to execute and return the expected results. For example, a book-buying service might require the ISBN number of an actual book as an input.

- *Post-condition*. The post-condition describes the relationship between the input and the output. For a book search service, a possible post-condition would be that it returns an ISBN number for a book if the input was a valid book title.

- *Assumptions*. Assumptions state what must hold for the state of the world for which the web service can successfully execute. These assumptions must be fulfilled by either the goal (teleological assumptions) or by the domain knowledge (ontological assumptions).

Note that if the domain knowledge is faulty and assumptions are assumed to hold, while they in fact do not (usually because of some user error or a fault in another web service), a web service will produce an error, or worse, behave in an unpredictable manner.

The pre-condition and the post-condition (together with the input and output) provide a declarative description of the functionality provided by the web service. The assumptions state requirements on the state of the world for the web service to behave as described by the pre- and post-condition. Note that, as in WSMF, input, through the input roles, is not restricted to parameters specified before the execution of the service. Input can be provided concurrently during execution of the service. Output is not restricted to after the execution of the service. Output data can be returned to the user as required, concurrently with the execution. See Figure 4 for an example web service description. Figure 5 presents a refiner for this generic web service.

Figure 4   Description of the generic Web service Transaction

```
web service
    name Transaction
    pragmatics
        A transaction between different parties;
        2003-12-29;
    import ontology Transaction
    specification
        roles
          the input consists of the initiating party (ip) and a number of participating parties; when the
          transaction is completed, a report is generated, stating the completion of the transaction
            input ip:TransactionParty; {pp: TransactionParty};
            output cr:TransactionCompletionReport;
        precondition
          The initiation party (ip) is required and there must be at least one participating party (pp).
        assumptions
        postcondition
```

Figure 5   Description of the Web service Purchasing

web service-refiner
    name Purchasing
    pragmatics
        There are only two parties in the transaction: a selling and a buying party;
        2003-12-29;
    import web service Transaction
    import ontology Purchase
    specification
      roles
        the initiating party (ip) is split up into two different inputs, namely the shipping information
        (si) and the paying information (pi). Note that the union of these two inputs is still a
        TransactionParty. The list of participating parties is reduced to one party, which is the selling
        party (sp). An additional input is the list of order lines, containing the ordered products with
        the requested quantities.
        The output is renamed to sp and is an instance of ShippingConfirmation. Note that the output
        of the service must still be an instance of TransactionCompletionReport.
          input ip → si:ShipInfo, pi:PayInfo; pp → sp:SellingParty; {ol:OrderLine};
          output cr → sp:ShippingConfirmation
      precondition
      The shipping and payment information are both required; there is exactly one selling party.
      assumptions
      pi contains valid payment information.
      postcondition
      If the selling party (sp) can deliver the requested products in the list of order lines, the product
      will be delivered and the shipping confirmation sp confirms the shipping of the requested
      products.

The order of message exchange should be specified in a conversational interface in order to allow for business logic mediation (Fensel and Bussler, 2002). A requester of a web service must know what input data are expected by the web service and what output data it can expect from the service. In order to allow for true business logic mediation, the requester should specify its conversational interface so that the mediator can mitigate any differences in business logics. We will not go into more detail here, as business logic mediation is not within the scope of this paper.

    A limitation, impeding the goal and web service independence, is the inherently distributed nature of the web service providers; furthermore, the operational specification typically contains private business logic of an organisation, which it does not want to share with the world. Because we cannot adapt the operational specification of a web service, we envision the goal-independent specification of web services at a more generic level. These generic web service descriptions are specialised, with the use of refiners, into more specialised descriptions, where, at a lower level, they describe actual web services. This helps to structure the space of web service descriptions. Note that the structuring of web service descriptions with web service-refiners is similar to the structuring of goal descriptions with goal-refiners.

4.2.3   Domain descriptions

WSMF does not allow for goals and web service descriptions to be reused across different domains. A goal and a web service are described using (potentially different) vocabularies with a connection (through an explicit link) between the two. We want to allow for a description of web services and goals independent of the specific domain. We recognise

that this might be hard to achieve in the context of web services, and we currently envision only describing generic goals and web services in a domain-independent way (Figure 6).

Figure 6 Description of the domain Books

domain
   name Books
   pragmatics
      The domain of Books at a particular vendor called 'myVendor';
      2003-12-29;
   import ontology Books, Selling
   knowledge
    myVendor is an instance of BookVendor.
     myVendor:BookVendor;
   meta-knowledge

The domain description does not only consist of static knowledge, but we also envision it to contain (semi-) dynamic knowledge. When, for example, an assumption in a web service description must be met when the web service is about to be invoked, the domain description should contain the knowledge necessary to prove the assumption. This knowledge can possibly come from the output of a web service that has just been invoked or from user input.

Other than in knowledge-based systems (which is the operating environment for problem-solving methods), on the web, the domain knowledge is not centrally distributed but stored at different, unknown locations. There is an open world and the knowledge of the world is limited.

The domain description consists of the domain vocabulary, in the form of an ontology. The other aspects are the meta-knowledge (in the form of axioms), which hold for the domain and the instances of the domain ontology, that is, the domain knowledge:

● *Domain ontology*. The domain ontology specifies the vocabulary used for the domain description.

● *Domain knowledge*. The domain knowledge consists of instances of the domain ontology. An example is a product catalogue or a table of country codes or the fact that a certain user is logged in (the output of a Login service).

● *Meta-knowledge*. These axioms are used to specify meta-facts (rules), which hold in the domain.

The agent holding the domain knowledge is usually the requester of the service. The requester needs to locate a goal and a domain (e.g. 'Buying' in the domain of 'Books') and a bridge between the two. If a bridge does not exist, one needs to be created. The goal, the domain and the bridge together create a new virtual domain-dependent goal. Using this domain-dependent goal, the requester can now search for an appropriate web service description that fulfils (part of) the goal.

4.2.4 Refiners

While in WSMF, web service descriptions are refinements of goal descriptions, we propose to refine goal and web service descriptions separately. By specifying the refinements separately, we allow for the description of generic goals and web services, which allows us to structure the space of web service and goal descriptions.

To allow for the stepwise refinement of goal, web service and domain descriptions, we introduce refiners. These refiners correspond to the refiners as they have been defined in UPML (Fensel et al., 2003), to the one-dimensional adapters for problem-solving methods described in Fensel and Motta (2001) and to the refinement adapter concept described in Fensel (1997b).

Goals, web services and domains are ideally described first in a very generic manner, after which refiners are used in order to create more specific descriptions.

When a goal or a web service is refined, no new goal or web service will be created. The refiner is a separate entity in that can be applied to different goals or web service descriptions, which, in the ideal case, enables reuse of refiners.

A goal or web service refiner consists of the following:

- A *name* to uniquely identify the refiner.

- An (optional) *ontology* used by the refiner, on top of the ontology used by the original goal or web service.

- A *reference* to the goal, web service or being refined. This reference is optional. We imagine the reuse of refiners for different descriptions, in which case we need connectors, connecting the descriptions with the refiners, enabling an *n2m* mapping between more generic and more refined services and goals, instead of a 1-to-1 mapping.

- Refinement of *pre-condition*, *post-condition* and *assumptions*. When the pre-condition is refined, the number of possible inputs is restricted, which is actually a strengthening of the pre-condition. When the post-condition is refined, the number of possible outputs is restricted, which weakens the post-condition. When assumptions are refined (strengthened), the number of states of the world for which the service can execute or the goal can be achieved, is restricted.

Refiners allow for the structuring of the description space and, when designed well, they allow for a high degree of reusability of descriptions, especially when there exists an *n2m* mapping between more generic and more specific descriptions (i.e. when refiners are reused).

4.2.5   Bridges

WSMF specifies an implicit bridge between goals and web service descriptions through the explicit link from a web service description to a goal: the *goal reference*. This direct link helps in the discovery task, because all web services that (partially or completely) fulfil a goal can be discovered analysing only syntactical links. We believe this puts too many constraints on the reusability of web service descriptions.

A second limitation of the WSMF problem is that (as defined in WSMF), the goal and web service description might use different vocabularies (i.e. a different ontology), which requires mediation and WSMF does not describe how the mediation between these different vocabularies should be described or executed.

For the purpose of relating goals and web services and the enabling of mediation and furthermore to enable domain-independence, we propose to use bridges (PSM-Task bridge in Fensel et al. (2003)) that connect web service descriptions with goals.

We identify three types of bridges; namely the goal-web service bridge, the web service-domain bridge and the goal-domain bridge. Bridges connect the independent descriptions of web services, goals and domain, thereby creating new (virtual) goal- and domain-dependent web services.

When a goal-domain bridge or a WS-domain bridge is applied to the descriptions, no new goal- or WS-description will be created. The bridges are separate modules, and the newly created domain-specific goal- and web service descriptions are created *virtually*; this enables the reuse of bridges and prevents maintenance problems for the original goal or web service description.

Goal-domain bridges enable the reuse of the web service and goal descriptions across domains. Web service-goal bridges enable the reuse of web services for different goals. Besides enabling reuse, the web service-goal bridge also enables data structure mediation. Data structure mediation has been identified as an important task in permitting inter-organisational communication (Fensel and Bussler, 2002). We do not describe business logic mediation here, because the description of conversational interfaces lies outside the scope of this paper.

4.2.5.1   *Goal-web service bridges*
The goal-web service bridge (see Figure 7 for an example) consists of the following:

- *Name*. A name uniquely identifying the bridge.

- *Web service and goal*. The web service and the goal for which the bridge is created.

- *Axioms*. The mapping between the web service and the goal is specified using a set of axioms.

- *Assumptions*. Assumptions introduced in order to make the bridge work.

Figure 7   Goal-Web Service bridge for goal Buying and Web service Purchasing

gws bridge
    name Buying @ Purchasing
    web service Purchasing
    goal Buying
    pragmatics
        This bridge connects the Buying goal and the Purchasing web service;
        2003-12-29;
    axioms
        GoodsDescription is (partially) mapped to OrderLine (every OrderLine is a GoodsDescription, but not the other way around) in order to relate the inputs of the goal and the Web service:
            OrderLine $\Rightarrow$ GoodsDescription;
        DeliveryAddress is mapped to ShipInfo (here equivalence is possible) in order to relate the inputs da and si:
            DeliveryAddress = ShipInfo;
        PaymentInformation is mapped to PayInfo in order to relate the inputs rg and pi
            PaymentInformation = PayInfo;
        ShippingConfirmation is partially mapped to Delivery (for an output this is no problem, because every output of the Web service still satisfies the output of the goal):
            ShippingConfirmation $\Rightarrow$ Delivery;
    assumptions
        There are no inputs for gd:GoodsDescription, which can not be translated to {ol:OrderLine}.

As was pointed out in WSMF (Fensel and Bussler, 2002), the content of the messages exchanges between the invoker and the executor needs to be made explicit. We assume that this is done by reference to ontology concepts in the input and output roles. Therefore, to enable data mediation, we need to specify ontology mappings in the goal-web service bridge. These mappings, consisting of explicit mapping rules between concepts and relations in different ontologies, enable transformation of instances between different representations. We need a language expressive enough to relate the concepts in two separate descriptions. Problems in the mapping of ontologies have been identified in Klein (2001) and there are several approaches available for the mapping of ontologies.

4.2.6   Ontologies

Ontologies (Fensel, 2003) provide the vocabulary for the goals, web services and domain descriptions. For now, the mappings between the different vocabularies are provided for in the bridges (e.g. Figure 8), but one could imagine that such mappings between ontologies exist outside of the bridges and are either imported or they exist in some ontology mapping repository.

Figure 8   Web Service-Domain bridge for domain Books and Web service Purchasing

```
wsd bridge
    name Purchasing @ Books
    web service Purchasing
    domain Books
    pragmatics
        This bridge connects the Purchasing Web service and the Books goal;
        2003-12-29;
    axioms
        The selling party sp is the myVendor from the domain.
          sp = myVendor;
        The range of ol.Product (the property Product of the OrderLine instances) is restricted to Book.
          ol.Product → Book;
        The classes in the different ontologies are related to each other
          Vendor::SellingParty;
          Book::Product;
```

Because ontologies are *formal consensual terminologies* (Fensel, 2003), they can be used to come to a common understanding between humans and machines. When an ontology is shared between different parties, this enables inter-operation. Furthermore, because ontologies are formal specifications, it is possible to create formal mappings, which enables automatic inter-operation between parties, using different, but interlinked, vocabularies.

A currently popular language for specifying ontologies on the semantic web is the web ontology language OWL (Dean and Schreiber, 2004).

One could imagine that ontologies can also be refined, just like the other components in our architecture. One possible way of doing this in OWL is by creating a new ontology and importing the more generic one with the owl:import statement. It is now possible to refine the different concepts and relationships in the more generic ontology.

4.2.7   Language requirements

For brevity, we used a semi-formal notation for the description of the example goal, web service and the bridges and refiners.

We envision using standardised semantic web languages for the ontologies and the descriptions of the domain, the goals, and the web services, such as OWL (Dean and Schreiber, 2004) for the description of the ontologies and a semantic web rule language for the description of the pre- and post-conditions, assumption and mapping axioms.

There is work under way in examining possibilities for specifying pre- and post-conditions for web services, among others in the SWWS[8] project and the DAML-S initiative (The DAML services coalition, 2003). There is currently the possibility of specifying pre- and post-conditions in DAML-S, but the modelling primitives provided are too restricted for our purposes, since DAML-S is restricted to OWL (OWL DL is equivalent to the SHOIN(D) Description Logic (Horrocks et al., 2003)). The main drawback of using a Description Logic language for the description of web services is that it is not possible to use variables. These variables are necessary to relate the input with the output in the post-condition of a web service or a goal. A logical language with variables is not necessarily less tractable than SHOIN(D). Datalog (Ullman, 1988), for example, is in the complexity class PTime, while SHOIN(D) is in NExpTime (Horrocks et al., 2003).

We are closely following the development of rule languages for the semantic web, such as RuleML (Boley et al., 2002), as these provide more expressiveness than Description Logics (Nardi et al., 2003) based languages such as OWL. Boley et al. (2001) made a first attempt at relating DAML+OIL (Connolly et al., 2001) (the predecessor of OWL) and RuleML. Grosof et al. (2003) show how to interoperate between DAML+OIL and RuleML by defining Description Logic Programmes (DLP), which translate Description Logic to Horn Logic. The current version of RuleML (0.8) has the expressivity of full Horn logic (Boley et al., 2001) and would therefore be expressive enough for our purposes. Recently (19 November 2003), a Semantic Web Rule Language (SWRL[9]) was proposed, combining RuleML with OWL. This initiative looks promising, but more research is required in order to find out whether this language is suitable for the description of web services.

4.3   Web services libraries

Fensel and Motta (2001) describe the structure of a problem-solving method library with a broad horizontal cover. We propose to apply the structures described there to describe a library of reusable web service descriptions, called a web services library (WSL). We must note here that web service descriptions do not correspond one-to-one with actual web service implementations. Abstract, generic web service descriptions are used to structure the web service descriptions and facilitate reuse of descriptions as well as discovery of web services. Connections between web service descriptions and actual implementations are made at a lower level in the refinement hierarchy.

Web service descriptions are organised in a hierarchical fashion in a web services library. However, the provider of the library does not show the decomposition of web services in the library to the users. The decomposition of a web service is private to the provider of the service. The web service is in fact a black box, as described in WSMF; only the public description is visible, while the internal workings are hidden from the invoker. A web service might invoke other web services during execution, but this is hidden from the invoker, so as to not expose the private business process of an enterprise. The invocation of other web services is, however, a very important aspect.

A possible extension of this back box approach would be a *grey box* approach, where the sub-goals or *web service proxies* (invocation of other Web services) are in the public description; the invoker of the service could then find services to fulfil these sub-goals and would not be dependent on the choices made by the web service provider.

If a web service needs other web services in order to live up to its capabilities, the web service becomes a composite web service. We will follow here the terminology employed in WSMF. A web service that does not invoke other services in its execution is an *elementary* web service. A web service that does invoke other services is a *composite* web service. Note that this distinction is only relevant to the provider of the service. The requester does not see the difference between an elementary and a composite service, the requester just sees a *web service* (except in the case of the aforementioned *grey box* approach).

We specify two views of a web services library (WSL):

- The *external view* consists of a set of public web service descriptions. We will assume that a WSL has some generic capability descriptions that are advertised to the public (possibly through a web services registry) and the actual web service descriptions, which are refinements of these generic capabilities and can be retrieved from the WSL. These generic capabilities are important for the first phase of the discovery process. It is then up to the provider of the WSL to decide which more specific web service descriptions publish in the second discovery phase. Note that these specific web service descriptions should be constructed from generic web service descriptions using web service-refiners.

- In the *goal-decomposition view*, each composite web service is decomposed into sub-goals, or *web service proxies*, as in WSMF. The actual data- and control-flow between the sub-goals is described in the implementation of the web service and is not important for this view. The importance of this view is the fact that it makes the dependencies between web services within the library explicit. If there is a web service present in the library that fulfils the goal, there will be an explicit link between this goal and the web service, through a bridge in the library. If no such web service is present in the library, the provider (or in the grey box approach the invoker) will have to search elsewhere to discover a web service that can take the place of the web service proxy.

A Goal-WS-bridge as illustrated in Figure 7 usually provides for a translation between the goal and the web service description, thereby creating a new virtual goal-dependent web service. Within a WSL, these bridges will be very simple and will often just state equivalence between goal and capability. When a goal needs to be matched with a capability in another WSL the bridge will provide a mapping between the goal and web service description. If an exact match is not possible, requirements on the domain knowledge (assumptions) may be introduced in the bridge in order to fill the gap between the goal and the web service.

As mentioned above, an important part of the external view consists of the generic capabilities exposed by the web services in the WSL. We envision these generic capabilities to be published in a web service registry (e.g. a UDDI registry with enhancements for semantic web services, cf. (Akkiraju et al., 2003; Paolucci et al., 2002b), enabling the discovery of web services libraries. Such a registry could employ syntactic

(through the existence of bridges between generic goals and generic capabilities) and semantic (through automatic theorem proving matching the generic user goal and the generic capability) matching in order to discover web service libraries that can fulfil a user goal. The user can then contact the WSL in order to discover more specific web service descriptions, which match (through a bridge) the more specific user goal.

4.4   Limitations of the approach

Currently no web languages are available to sufficiently describe the goals and web service capability descriptions needed in our approach (Section 4.2.7: language requirements). As research in the area of the description of web services progresses, we hope to see semantic web services description languages arise, which are expressive enough for our purposes. We believe that the SWRL proposal is a step in the right direction.

We did not focus on the specification of the conversational interface of web service or mediation of business logics. We recognise that it is necessary to specify the conversational interface of both the requester and the web service in order to allow for mediation between the different interaction styles.

Other aspects we left out, are security,[10] trust,[12] error handling, compensation and non-functional properties, such a cost and quality of service. These aspects are all mentioned in the web service modelling framework, WSMF. We consider our effort an extension of the functional description of web services in order to allow for reuse of web service and, therefore, left out these other (also important) aspects.

The architecture we have proposed has not been implemented. We, therefore do not yet know the practical implications of such an architecture, but once current language issues have been resolved, we plan to work towards implementing this architecture. Existing implementations of the current web services standards SOAP, WSDL and UDDI and implementations of rule engines such as TRIPLE and XSB do help in speeding up the implementation process.

5   Related work

Web services provide a valuable infrastructure to support the development of new business models and virtual organisations. Major aspects in supporting these models are automatic task driven discovery and composition of services. SOAP, WSDL and UDDI are important steps in the direction of a web populated by services. However, they only address part of the overall stack that needs to be available to lift the web to its full potential. SOAP provides the means to invoke the services but as different parties may use different protocols and vocabularies; we need means to align different terminologies. WSDL allows the description of services using semi-formal natural language terms but does not say anything about the actual functionality of the services (what they do) nor how it does it in a machine-processable way. UDDI facilitates the means to advertise web services through the use of keywords but does not support semantic description.

Current technology does not allow us to realise the automation process, therefore, extensions to the web service architecture are required in order to minimise user interaction so that the discovery, composition and execution of web services can be done

174   *de Bruijn et al.*

in a task-driven automated way. Below we mention some related approaches that aim to automate web service discovery, selection and composition, as well as arbitrary software components.

5.1   Approaches in web services discovery and selection

Several discovery approaches using semantic descriptions of web services have been proposed. Most of these approaches use ontologies and semantic matchmaking mechanisms. The matchmaking process is carried out by matching together goals and capabilities between client and web service. ATLAS (Paolucci et al., 2002a) is a DAML-based agent advertising language that enables agents and services to locate each other and interoperate. ATLAS has two types of matching methods: multiple criteria decision making (MCDM) and the conjunctive constraint method. The semantic matchmaking using RDF graphs (Trastour et al., 2001) uses semantic web related technologies. The matchmaking based on process ontologies (Klein and Bernstein, 2001) rather than frame-based approaches exploits the use of process model representation of service semantics and process ontologies to improve service discovery. DReggie (Chakraborty et al., 2002) based on DAML uses a Prolog reasoner for semantic discovery.

5.2   Approaches in web services composition

Web service composition is an emerging issue in the semantic web community. Efforts can be summarised into two types of service composition. The first, from the software engineering perspective, regards web services as service components. Approaches of the second type take the workflow point of view. Web service composition can be conceptualised as composition patterns as in software design patterns (Gamma et al., 1995). In Benatallah et al. (2001), a number of patterns for the definition and implementation of composite services are discussed. The patterns suggest a methodology for building a new service that tackles each of the issues occurring in service composition separately.

In service composition patterns, component services are described using interpretable information, i.e. an ontology, which provides means for automatic discovery and composition. On the other hand, workflow-based approaches such as web service orchestration (Peltz, 2003) and web service choreography (Arkin et al., 2002) describe the flow of messages exchanged between participating web services. Workflow management systems are capable of integrating business objects for setting up web services in an amazingly short time and with impressively little cost (Shegalov et al., 2001). In most workflow processes the discovery or selection of web services is done using a local repository when the required web services are not known a priori. Most of approaches based on workflow use static discovery and composition.

BPEL4WS (Akkiraju et al., 2003) and BPML/WSCI (Arkin, 2003; Arkin et al., 2002) are currently the most popular languages to describe workflow between composite web services. They have similar functionalities, both aiming at defining a language to describe process models, as well as public process interfaces and service choreography support, to provide conversational and interoperation means for web services; they focus on the static definition of business processes at design time, and do not provide enough means to support dynamic discovery, composition, invocation and interoperation the web services.

5.3 Approaches in software architecture

Web services can be viewed as distributed heterogeneous software components available over the internet. To compose these components, adaptation mechanisms are essential. As software component adaptation shares features with web service composition, many research efforts (Penix and Alexander, 1997; Wiederhold, 1992; Yellin and Storm, 1997) provide theoretic insights into web service composition. A new component-based way to construct software systems, called Invasive Software Composition (Aßmann, 2003) will provide efficient methods to integrate software components during composition and unify several software engineering techniques such as generic programming, view-based programming and aspect orientated programming. Inside the source code a number of code hooks is defined that enable 'plugging' pieces of code into these components, which allows the adaptation of components, where the actual execution code is changed. Penix and Alexander (1997) briefly mention the concept of subcomponent replacement. They present a hierarchical view of software architectures, where components in a software hierarchy can themselves be other software architectures consisting of components. With subcomponent replacement they mean replacing a component in a pre-defined architecture with another component, in order to enable reuse in different problem areas. Penix and Alexander (1997) and Penix et al. (1997) share our vision of goal-independent (or problem-independent) software components, although they do not provide a methodology for finding such problem-independent components.

6 Conclusions and future work

In this paper, we have presented the vision of semantic web services and the role of the web services modelling framework WSMF (Fensel and Bussler, 2002). We have extended WSMF using concepts from the area of problem-solving methods (Fensel, 2000) and specifically its architectural description language UPML (Fensel et al., 2003). We proposed these extensions in order to further facilitate the automation of discovery, selection, composition, and invocation and in order to enable reuse of web services across goals and domains.

There are some prerequisites in order to enable the use of the concepts presented in WSMF and extended in this paper. More research needs to be done especially in the area of the description of web services in order to be able to fully describe the functionality of web services. We have seen in Section 4.2.7 that we will need a more expressive language than the currently specified web ontology language OWL for the full specifications of web services, goals and domains and their refinements and the links between them.

Data mediation has much in common with, and in fact relies on ontology mapping (Ding and Foo, 2002; Fensel, 2003; Klein, 2001). Data in semantic web services is structured using ontologies and having explicit mapping rules between ontologies enables the transformation of these data from one representation to another. There are currently no semantic web-enabled languages that can be used for the purpose of ontology mapping, but there are promising intermediate steps, such as description logic programs (Grosof et al., 2003) and RuleML (Boley et al., 2001). Possible candidates for reuse as a semantic web rule language are F-Logic (Kifer et al., 1995) and F-Logic based solutions such as TRIPLE (Sintek and Decker, 2001) and FLORA-2 (Yang et al., 2003).

In this paper, we have not discussed business logic mediation, that is, mediation between different interaction styles (Bussler, 2001). In order to enable interaction between different parties with different interaction styles, each party has to make its interaction style explicit using a conversational interface, which describes the public process (Bussler, 2001) of either the requester or the provider of the web service. This conversational interface is used by the mediator in order to mitigate differences in interaction styles. Business logic mediation is a major issue in both the SWWS and the DIP project. Candidates for the specification of conversational interfaces are BPEL4WS (Andrews et al., 2003) and BPML/WSCI (Arkin, 2003; Arkin et al., 2002).

Our main contributions in this paper are the extension of WSMF to enable the reuse of web services and the introduction of web services libraries as a way to organise web services and to facilitate web service discovery and to facilitate the reuse of web service descriptions. Major challenges remain in identifying generic web service descriptions and generic goals and their respective refinement.

In this context, future research will have to show which existing generic tasks and generic problem-solving methods we can reuse in the area of web services. Some libraries of problem-solving methods have been described in Benjamins (1995), Motta (1999) and Motta and Lu (2000). We hope to be able to reuse these efforts in order to further develop libraries of web services. Efforts are ongoing in the context of the web service modelling ontology, WSMO (Roman et al., 2004) to realise the vision of semantic web services based on WSMF.

References

Akkiraju, R., Goodwin, R., Doshi, P. and Roeder, S. (2003) 'A method for semantically enhancing the service discovery capabilities of UDDI', *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, pp.87–92.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I. and Weerawarana, S. (2003) *Business Process Execution Language for Web Services, Version 1.1*, available from: ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf.

Ankolenkar, A., Burstein, M., Son, T.C., Hobbs, J., Lassila, O., Martin, D., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K. and Zeng, H. (2001) 'DAML-S: semantic markup for web services', *Proceedings of the 1st International Semantic Web Working Symposium (SWWS '01)*, Stanford, CA, USA.

Arkin, A. (2003) *Business Process Modeling Language, BPMI Proposed Recommendation*, available from: www.bpmi.org/.

Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S., Takacsi-Nagy, P., Trickovic, I. and Zimek, S. (2002) *Web Service Choreography Interface (WSCI) 1.0*, W3C Note 8, August 2002, W3C, available from: www.w3.org/TR/2002/NOTE-wsci-20020808.

Aßmann, U. (2003) *Invasive Software Composition*, Berlin: Springer-Verlag.

Benatallah, B., Dumas, M., Fauvet, M.C. and Rabhi, F. (2001) 'Towards patterns of web services composition', *Technical Report UNSW-CSE-0111*, The University of New South Wales.

Benjamins, V.R. (1995) 'Problem-solving methods for diagnosis and their role in knowledge acquisition', *International Journal of Expert Systems: Research & Applications*, Vol. 8, No. 2, pp.93–120.
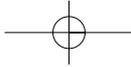
Benjamins, V.R. (1997) 'Problem-solving methods in cyberspace', *Proceedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems at the 15th International Joint Conference on AI (IJCAI '97)*, Nagoya, Japan, pp.1–18.

Benjamins, V.R., Plaza, E., Motta, E., Fensel, D., Studer, R., Wielinga, B., Schreiber, G., Zdrahal, Z. and Decker, S. (1998) 'IBROW3: an intelligent brokering service for knowledge-component reuse on the world-wide web', *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW '98)*, Banff, Canada.

Berners-Lee, T., Hendler, J. and Lassila, O. (2001) 'The semantic web', *Scientific American*, Vol. 284, No. 5, pp.34–43.

Beys, P., Benjamins, V.R. and Van Heijst, G. (1996) 'Remedying the reusability-usability trade-off for problem-solving methods', *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW '96)*, Banff, Canada, pp.2.1–2.20.

Boley, H., Grosof, B., Sintek, M., Tabet, S. and Wagner, G. (2002) *RuleML Design*, available from: www.dfki.uni-kl.de/ruleml/indesign.html.

Boley, H., Tabet, S. and Wagner, G. (2001) 'Design rationale of RuleML: a markup language for semantic web rules', *Proceedings of the 1st International Semantic Web Working Symposium (SWWS '01)*, Stanford, CA.

Bryson, J.J., Martin, D.L., McIlraith, S.A. and Stein, L.A. (2002) 'Toward behavioural intelligence in the semantic web', *IEEE Computer*, Vol. 35, No. 11, pp.37–44.

Bussler, C. (2001) 'The role of B2B protocols in inter-enterprise process execution', *Proceedings of Workshop on Technologies for E-Services (TES 2001) (in cooperation with VLDB2001)*, Rome, Italy.

Chakraborty, D., Perich, F., Avancha, S. and Joshi, A. (2002) 'An agent discovery architecture using Ronin and DReggie', *First GSFC/JPL Workshop on Radical Agent Concepts (WRAC)*, NASA Goddard Space Flight Center, MD.

Connolly, D., Van Harmelen, F., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F. and Stein, L.A. (2001) *DAML+OIL (March 2001) Reference Description*, available from: www.w3.org/TR/daml+oil-reference.

Dean, M. and Schreiber, G. (Eds) (2004) *Web Ontology Language Reference*, W3C Recommendation, 10 February 2004.

Ding, Y. and Foo, S. (2002) 'Ontology research and development, part 2 – a review of ontology mapping and evolving', *Journal of Information Science*, Vol. 28, No. 5, pp.375–388.

Fensel, D. (1997a) 'An ontology-based broker: making problem-solving method reuse work', *Workshop Proceedings Problem-solving Methods for Knowledge-based Systems in Connection with the 15th International Joint Conference on Artificial Intelligence (IJCAI '97)*, Nagoya, Japan.

Fensel, D. (1997b) 'The tower-of-adapter method for developing and reusing problem-solving methods', in V.R. Benjamins and E. Plaza (Eds) *Knowledge Acquisition, Modeling, and Management. Proceedings of the 10th European Workshop, EKAW '97*, *Lecture Notes in Artificial Intelligence 1319*, Berlin: Springer-Verlag, pp.97–112.

Fensel, D. (2000) *Problem-solving Methods: Understanding, Development, Description, and Reuse*, *Lecture Notes on Artificial Intelligence 1791*, Berlin: Springer-Verlag.

Fensel, D. (2003) *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, 2nd edition, Berlin: Springer-Verlag.

Fensel, D. and Benjamins, V.R. (1998a) 'The role of assumptions in knowledge engineering', *International Journal of Intelligent Systems (IJIS)*, Vol. 13, No. 8, pp.715–748.

Fensel, D. and Benjamins, V.R. (1998b) 'An architecture for reusing problem-solving components', *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI '98)*, Brighton, UK, pp.63–67.

178   *de Bruijn et al.*

Fensel, D. and Bussler, C. (2002) 'The Web Service Modeling Framework WSMF', *Electronic Commerce Research and Applications*, Vol. 1, No. 2, pp.113–137.

Fensel, D. and Motta, E. (2001) 'Structured development of problem solving methods', *Transactions on Knowledge and Data Engineering*, Vol. 13, No. 6, pp.913–932.

Fensel, D. and Straatman, R. (1998) 'The essence of problem-solving methods: making assumptions to gain efficiency', *The International Journal of Human Computer Studies (IJHCS)*, Vol. 48, No. 2, pp.181–215.

Fensel, D., Motta, E., van Harmelen, F., Benjamins, V.R., Decker, S., Gaspari, M., Groenboom, R., Grosso, W., Musen, M., Plaza, E., Schreiber, G., Studer, R. and Wielinga, B. (2003) 'The unified problem-solving method development language UPML', *Knowledge and Information Systems(KAIS) Journal*, Vol. 5, No. 1.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) *Design Patterns*, Addison-Wesley Pub.

Grosof, B.N., Horrocks, I., Volz, R. and Decker, S. (2003) 'Description logic programs: combining logic programs with description logic', *Proc. Intl. Conf. on the World Wide Web (WWW-2003)*, Budapest, Hungary.

Gruber, T.R. (1993) 'A translation approach to portable ontology specifications', *Knowledge Acquisition*, Vol. 5, No. 2, pp.199–220.

Horrocks, I., Patel-Schneider, P.F. and van Harmelen, F. (2003) 'From SHIQ and RDF to OWL: the making of a web ontology language', *Journal of Web Semantic*, Vol. 1, No. 1, pp.7–26.

Kifer, M., Lausen, G. and Wu, J. (1995) 'Logical foundations of object-oriented and frame-based languages', *Journal of the ACM*, Vol. 42, No. 4, pp.741–843.

Klein, M. (2001) 'Combining and relating ontologies: an analysis of problems and solutions', in A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt and M. Uschold (Eds) *Workshop on Ontologies and Information Sharing*, IJCAI '01, Seattle, WA.

Klein, M. and Bernstein, A. (2001) 'Searching for services on the semantic web using process ontologies', *Proceedings of the 1st International Semantic Web Working Symposium (SWWS '01)*, Stanford, CA.

Lara, R., Lausen, H., Arroyo, S., de Bruijn, J. and Fensel, D. (2003) 'Semantic web services: description requirements and current technologies', *Semantic Web Services for Enterprise Application Integration and e-Commerce workshop (SWSEE '03), in Conjunction with ICEC 2003*, Pittsburgh, PA.

McIlraith, S., Son, T.C. and Zeng, H. (2001) 'Semantic web services', *IEEE Intelligent Systems*, Special Issue on the Semantic Web, Vol. 16, No. 2, pp.46–53.

Motta, E. (1999) *Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design*, Amsterdam: IOS Press.

Motta, E. and Lu, W. (2000) 'A library of components for classification problem solving', *PKAW 2000: The 2000 Pacific Rim Knowledge Acquisition Workshop*, Sydney, Australia.

Nardi, D., Brachman, R. J., Baader, F. et al. (2003) *The Description Logic Handbook*, Cambridge University Press.

Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K. (2002a) 'Semantic matching of web services capabilities', *Proceedings of The 1st International Semantic Web Conference (ISWC 2002)*, Sardinia, Italy.

Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K. (2002b) 'Importing the semantic web in UDDI', *Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop, WES 2002*, Toronto, Canada, LNCS 2512, Berlin: Springer, pp.225–236.

Peltz, C. (2003) *Web Services Orchestration: A Review of Emerging Technologies, Tools, and Standards*, Hewlett Packard Co, available from: http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf.

Penix, J. and Alexander, P. (1997) 'Toward automated component adaptation', *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering (SEKE-97)*, Madrid, Spain.

Penix, J., Alexander, P. and Havelund, K. (1997) 'Declarative specification of software architectures', *Proceedings of the 12th IEEE International Conference on Automated Software Engineering (ASEC-97)*, Incline Village, NA.

Roman, D., Lausen, H. and Keller, U. (Eds) (2004). *Web Service Modeling Ontology (WSMO)*, WSMO Final Draft D2v1.0, available from www.wsmo.org/2004/d2/v1.0/.

Shegalov, G., Gillmann, M. and Weikum, G. (2001) 'XML-enabled workflow management for e-services across heterogeneous platforms', *VLDB Journal*, Vol. 10, No. 1, pp.91–103.

Sintek, M. and Decker, S. (2001) 'TRIPLE – an RDF query, inference, and transformation language', *Proceedings of the 1st International Semantic Web Conference (ISWC2002)*, Sardinia, Italy.

Sirin, E., Hendler, J. and Parsia, B. (2003) 'Semi-automatic composition of web services using semantic descriptions', *Web Services: Modeling Architecture and Infrastructure Workshop in Conjunction with ICEIS 2003*, Anger, France.

The DAML services coalition (2003) *DAML-S: Semantic Markup for Web Services (Version 0.9)*, available from: www.daml.org/services/daml-s/0.9/daml-s.pdf.

Trastour, D., Bartolini, C. and Gonzalez-Castillo, J. (2001) 'A semantic web approach to service description for matchmaking of services', *Proceedings of the 1st International Semantic Web Working Symposium (SWWS '01)*, Stanford, CA, USA.

Ullman, J.D. (1988) *Principles of Database and Knowledge-Base Systems, Volume I*, Computer Science Press.

Wiederhold, G. (1992) 'Mediators in the architecture of future information systems', *IEEE Computer*, Vol. 25, No. 3, pp.38–49.

Yang, G., Kifer, M. and Zhao, C (2003) 'FLORA-2: a rule-based knowledge representation and inference infrastructure for the semantic web', *Proceedings of the 2nd International Conference on Ontologies, Databases and Applications of Semantics (ODBASE)*, Catania, Italy.

Yellin, D.M. and Storm, R.E. (1997) 'Protocol specifications and component adaptors', *ACM Transactions on Programming Languages and Systems*, Vol. 19, No. 2, pp.292–333.

Notes

[1] This research was partially funded by the European Commission funded project SWWS, contract number IST-2002-37134, http://swww.semanticweb.org/.

[2] SOAP is a message layout specification that defines a uniform way of passing XML-encoded data. http://www.w3.org/TR/SOAP/.

[3] WSDL defines services as collections of network endpoints or ports. In WSDL the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. http://www.w3.org/TR/wsdl12/.

[4] UDDI provides a mechanism for clients to find web services. Using a UDDI interface, businesses can dynamically lookup as well as discover services provided by external business partners. http://www.uddi.org/

[5] http://www.swi.psy.uva.nl/projects/IBROW3/home-ibrow.html.

[6] Note that the user can be a human or a machine agent.

180   *de Bruijn et al.*

[7]   http://www.w3.org/International/O-URL-and-ident.html.
[8]   http://swws.semanticweb.org/.
[9]   http://www.daml.org/rules/proposal/.
[10] WS-Security standard: http://www-106.ibm.com/developerworks/library/ws-secure/.
[11] WS-Trust standard: http://msdn.microsoft.com/ws/2002/12/ws-trust/.