

## Semantic and syntactic service descriptions at work in geo-service chaining

Rob Lemmens<sup>1</sup>, Carlos Granell<sup>2</sup>, Andreas Wytzisk<sup>1</sup>, Rolf de By<sup>1</sup>,  
Michael Gould<sup>2</sup>, Peter van Oosterom<sup>3</sup>

<sup>1</sup> International Institute for Geo-Information Science and Earth Observation (ITC)  
Enschede, The Netherlands

lemmens@itc.nl

<sup>2</sup> Universitat Jaume I

Castellón, Spain

carlos.granell@lsi.uji.es

<sup>3</sup> Delft University of Technology

Delft, The Netherlands

### SUMMARY

*In this paper we demonstrate the integrated use of semantic and syntactic service descriptions for service chaining, by combining a prototype that deals with service discovery and abstract composition, and one that supports concrete composition and execution of services. Current XML-based service description languages, such as OWL-S and WSDL-S, allowed us to build a geo-service-reuse architecture based on common ontologies and shared service descriptions.*

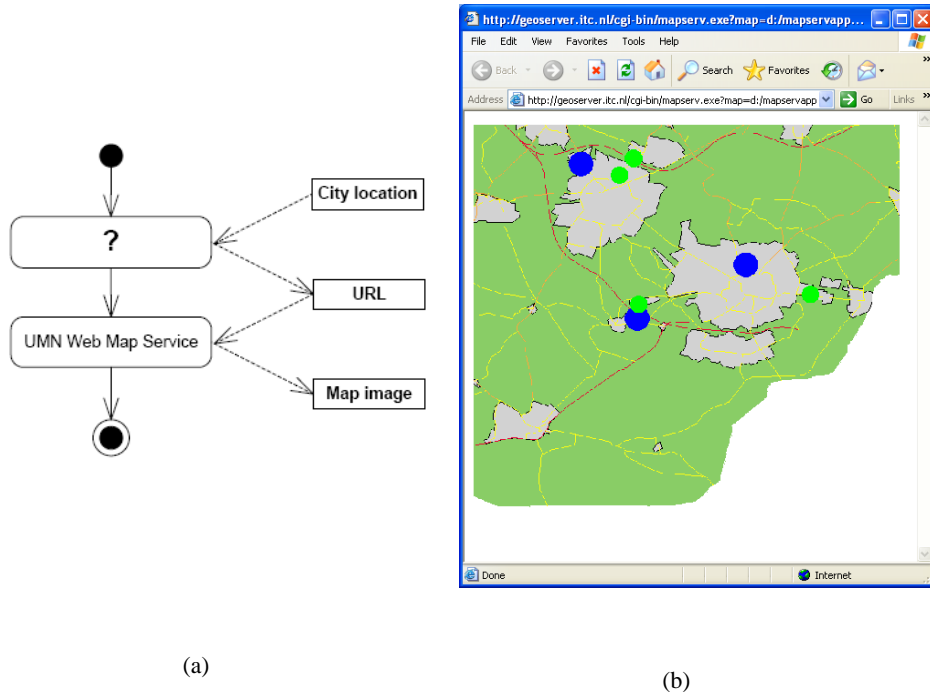
**KEYWORDS:** *Service chaining, service descriptions, discovery, reuse, WSDL-S, OWL-S*

### INTRODUCTION

The development of modular software processes is driven by the need for process distribution over computer networks and software reuse. The integrated exploitation of those processes in distributed systems can be facilitated by *service chaining*, which involves service discovery, abstract composition (identifying service chain functionality), concrete composition (identifying service chain messaging) and execution, typically in this order. The first two receive much attention from research in semantics, the latter two deal with syntactic issues. Currently, most approaches in geo-service chaining address semantic and syntactic issues separately. In this paper, we identify their relations and demonstrate an integrated approach, by combining two prototypes that were developed independently. One deals with service discovery and abstract composition ('GeoMatchMaker'), and the other supports concrete composition and execution of services ('Integrated Component Designer'). In the next section we start with a description of a risk mapping use case that is used throughout the paper.

### USE CASE

A simple use case was created for testing combined service discovery and composition. The aim of this use case is to create a service chain that generates a map with information about potentially hazardous objects such as ammonia and fireworks storages, and centre this map around a location provided by a human user. The starting point is the last service in the 'RiskMap' chain: an OGC-compliant Web Map Server, implemented with the University of Minnesota WMS software. Instead of having to provide this service with a URL to display a map, we want to let the end user enter the name of a city and let the service chain do the rest. The elements of the service chain and the aimed output are depicted in respectively Figure 1a and b.



**Figure 1:** a. UML activity diagram representing the RiskMap service chain; b. Output map of the last service in the potential chain, shown in a Web Browser. The map shows the area around the city of Enschede with fireworks depots (small circles) and ammonia storage locations (large circles).

**SYNTACTIC SERVICE DESCRIPTIONS**

The main goal of syntactic service descriptions, such as the Web Service Description Language (WSDL), is to describe interfaces of web services for invocation purposes. Our interest in WSDL is mainly focused on the abstract part of a WSDL description —operation and input/output messages— in terms of service discovery and chaining. Implementation details such as binding and port tags, also described in WSDL, will be needed during the service execution. At this stage, we make use of WSBPEL (WSBPEL, 2005) that expresses in a XML-based language how a set of web services are to be invoked. Both specifications are expected to become recommendations under their respective committees (W3C and OASIS), yet they treat web services at the syntactical level, which is insufficient for creating meaningful descriptions of web services.

**SEMANTIC SERVICE MODELLING**

To improve semi-automatic discovery methods, services have to be described with formal languages that allow for machine reasoning. Current research issues include ontology design, client interfacing and reasoning (see also the work of Klien et al., 2006). The Web Ontology Language (OWL) is a recommended specification of W3C and facilitates the creation of web-based ontologies. OWL draws upon the formal theory of Description Logics, which has roots in first-order predicate logic and provides highly expressive concept-forming constructs (Baader et al., 2003). OWL-Services (Martin et al., 2004), or OWL-S in short, is an upper-ontology based on OWL that models the characteristics of Web services and that can be used to create semantically enriched Web Service descriptions.

OWL-S provides three modelling constructs at the top level, i.e., the service profile (what the service does), the service grounding (how the service can be accessed) and the service model (how to use the service in terms of semantic content, including its workflow). OWL-S provides classes that can be instantiated by a service provider to create specific service descriptions. Because OWL-S is an upper-ontology, it obviously does not provide domain ontologies. These have to be established by service communities themselves. An integrating framework, implemented in the work of this paper, is presented below.

### Ontology framework

Service descriptions are provided in a semantic framework, which is the combination of ontologies and their relationships, as well as the way they are used to describe software artefacts (e.g., services, data sets, etc.). The proposed framework is depicted in Figure 2. At the basis are three types of formal ontology:

- A **feature concept ontology** formally defines the conceptualisations of real world phenomena and the relationships between them. For example, ‘building’ is a feature type that is (partially) defined by its thematic attributes, and spatial attributes.
- A **feature symbol ontology** formally defines the abstract elements that make up a feature in an object/field model, based on the ISO19109 standard (ISO/TC211, 2003a). This model distinguishes three abstraction levels, i.e., meta-level, implementation level and data level.
- A **geo-operation ontology** formally defines types of operations in terms of their behaviour and is based on OWL-S. Each operation type is characterised by the behaviour of one out of a set of well-known atomic GIS operations (inspired by the ISO19119 service taxonomy (ISO/TC211, 2003b)) and its typical input and output parameters. The parameter types are represented by symbol ontology elements (at the information meta level) and by ‘OP’ classes with similar names and class structure as in the information application level. The *typeBijection* relation is used as bridge between abstraction levels.

### Service descriptions

Semantic service metadata can be accommodated in the ontology by an operation class representation and/or a representation with individuals (class instances). A (partial) class definition of a gazetteer operation (which serves as a candidate operation for our RiskMap chain) in a Description Logic axiom is shown below. The prefixes refer to the hosting ontology. ‘LocSpat’ stands for the operation that reads a location attribute type (e.g. an address) and produces a spatial attribute type (e.g., a geometric object), typical to what a gazetteer operation performs.

$$\begin{aligned} \text{opera:LocSpat} &\sqsubseteq \\ &\text{opera:AcrossAttributeTypes} \sqcap \\ &(\exists \forall \text{ opera:hasInputParameter.symbol:GF\_LocationAttributeType}) \sqcap \\ &(\exists \forall \text{ opera:hasOutputParameter.symbol:GF\_SpatialAttributeType}) \end{aligned}$$

with:

$\sqsubseteq$  ‘is subclass of’  
 $\exists \forall$  conjunction of ‘there exists at least one’ and ‘for all’  
 $\sqcap$  ‘intersected with’  
 $.$  separator between role and role-filler



## ANNOTATION APPROACHES

Having explained syntactic and semantic service descriptions, this section outlines two approaches to link syntactic and semantic descriptions.

### OWL-S grounding

OWL-S provides abstract constructs for input and output parameters of processes. It does not explicitly describe the concrete I/O messages, but rather specifies, in a so-called *grounding*, how they must be linked to parameters in a concrete message mechanism. In the OWL-S specification version 1.1, WSDL is used as the grounding mechanism. For each OWL-S process, a mapping is created between each I/O parameter of the OWL-S process model and its corresponding target parameter in the WSDL document. Further, other parameters, such as operation name and a URI, pointing to the actual WSDL document, are specified. The use of an OWL-S processor, such as the OWL-S Virtual Machine, allows, based on the combination of OWL-S process and grounding, for the control of interaction between web services (Paolucci et al., 2004).

### WSDL-S

The Web Service Description Language (WSDL) solely represents the syntactical behaviour of web services, lacking semantic expressivity. Akkiraju et al. (2005) have proposed WSDL-S, which annotates web services by enriching WSDL descriptions with semantic tags. In our approach, we have borrowed some features of WSDL-S to semantically annotate operations and parameters.

```

- <wSDL:message name="getCoordinatesResponse">
  <wSDL:part name="output" element="xsd1:ResponseType"
    wssem:modelReference="Ontology0#ADL_Point"/>
</wSDL:message>
- <wSDL:message name="getCoordinatesRequest">
  <wSDL:part name="name" element="xsd1:RequestType"
    wssem:modelReference="Ontology0#ADL_CityName"/>
</wSDL:message>
- <wSDL:portType name="ADLGazClient">
  <wSDL:operation name="getCoordinates"
    wssem:modelReference="Ontology0#LocSpat"/>
</wSDL:portType>

```

Figure 3: WSDL-S snippet for ADL Gazetteer service.

Figure 3 denotes the annotated WSDL-S snippet for ADL Gazetteer. Specifically, the input and output message *part* and *operation* tags of WSDL are annotated via the WSDL-S *modelReference* attribute to describe what they mean. For example, the annotation for the operation *getCoordinates* refers to the concept *LocSpat* in the geo-operation ontology, which formally defines an operation that returns a spatial attribute type, based on a location. Part tags are annotated in the same manner.

## SERVICE CHAINING

This section presents the integrated approach for service chaining using syntactic and semantic descriptions (as illustrated in Figure 4). Regarding semantic descriptions, we assume that a set of common geo-ontologies is shared by all participants. It is also assumed that services have been annotated by service providers of such geo-ontologies. Moreover, annotated services found by the discovery process are directly consumed by the composition process to build a concrete composition. As new compositions are published again in the web services repository, not only single services are discovered but also compositions, thus increasing the service reuse. The following subsections explain in detail the proposed architecture of Figure 4.

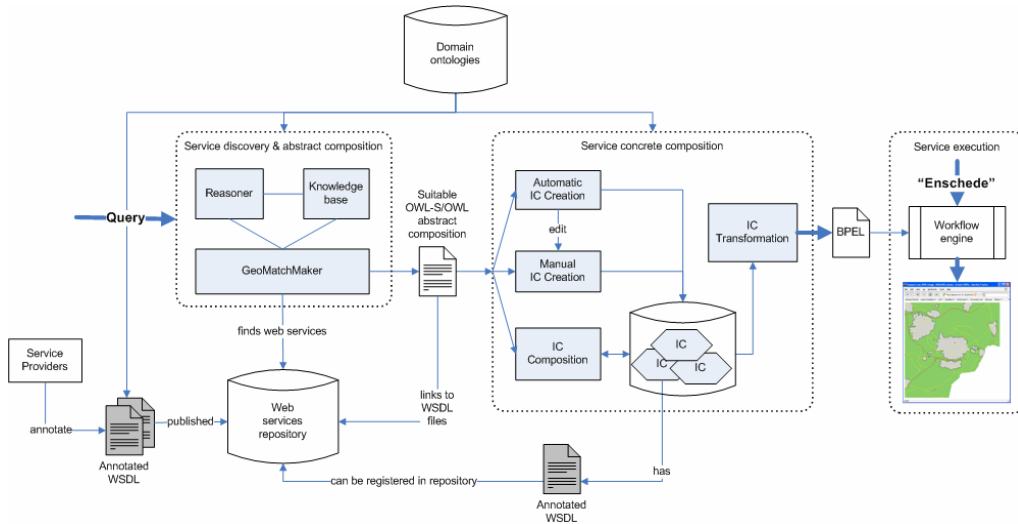


Figure 4: Integrated architecture for service chaining.

### Discovery & abstract composition

Geo-service discovery in general involves the identification of service advertisements that may match a service request.

Consider the service chain with  $n$  services:  $chain(S^1, \dots, S^n)$

In service chaining, we seek for cross matches between the output parameters of a service and the input parameters of a subsequent service and evaluate the behavioural aspects of the combination. For searching a service  $S^{i+1}$  that follows a given service  $S^i$ , a requesting ontology concept  $R$  (representing the testing service  $S^i$ ) is tested against an advertising concept  $A$  (representing a candidate service  $S^{i+1}$ ). Depending on whether we use class-based or individuals-based definitions of the operations, there are four possibilities to test the concept match. Classes are denoted with upper case, individuals with lower case, below,

$conceptmatch(R_{out}(S^i), A_{in}(S^{i+1}))$	type I	(between classes)
$conceptmatch(R_{out}(S^i), a_{in}(S^{i+1}))$	type II	(between class $R$ and individuals $a$ )
$conceptmatch(r_{out}(S^i), A_{in}(S^{i+1}))$	type III	(between individuals $r$ and class $A$ )
$conceptmatch(r_{out}(S^i), a_{in}(S^{i+1}))$	type IV	(between individuals)

Concept match type I matches class descriptions only. This is done by TBox reasoning. Concept match types II, III, IV are performed with individuals by ABox reasoning. In a knowledgebase, the TBox (T stands for ‘Terminology’) holds declarations of concepts and the ABox contains assertions (hence the term ‘ABox’), specific to individuals (instances of the concepts) (Baader et al., 2003). Differences between TBox and ABox reasoning in the context of this paper have been discussed in (Lemmens et al., 2004). In our GeoMatchMaker prototype, type II matches have been performed with the (RacerPro<sup>8</sup>) reasoner (see Figure 4) by inferring all candidate individuals  $a$  in the knowledge base that instantiate  $R$ . More details on the reasoning process can be found in (Lemmens, 2006). For brevity reasons, we elaborate only on the search of a service that follows the first service (for which

<sup>8</sup> <http://www.racer-systems.com/>

we have selected the ADL Gazetteer service). Figure 5 shows the results in terms of a set of matching services. These are services that create a bounding box around the geometric point, generated by the gazetteer. They are further evaluated by precisizing the requesting concept until one is left. After selecting the BBoxCreate service, there is one service left to complete the chain. This is a service that must build a GetMapRequest from the bounding box. Information, such as feature selection and coordinate system metadata, which are needed by the GetMapRequest, are also added in this service.

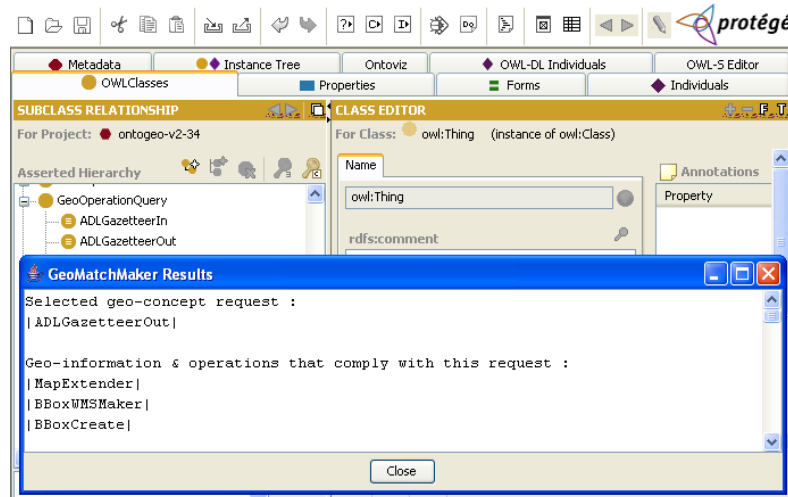


Figure 5: Output of the GeoMatchMaker prototype (discovery part).

The GeoMatchMaker prototype integrates the Protégé ontology editor<sup>9</sup> and provides an interactive environment to compose the service chain. The chain can be exported for execution purposes in different forms, such as an OWL-S document, which supports nine control flow patterns. The structure of the service chain, modelled in OWL-S is depicted in Figure 6a. The boxes represent instances of OWL-S process concepts. Amongst them are the discovered geo-operations (*ADLGazetteer*, *BboxCreate*, *MakeGetMapRequest*) and supporting control constructs (*Sequence*, *Perform*, etc). The sequence pattern can be recognised by following the ‘first-rest’ control flow and is portrayed as a UML activity in Figure 6b.

### Concrete composition & execution

Our service composition approach stems directly from the definition of the software composition systems, which are predominantly characterised by a component model and a composition method (Szyperski, 1998). The component model defines how to describe components to enhance their reusability. The composition method describes the mechanisms used for composing such components. In our research, we provide the integration component concept (component model) and the service composition method.

Integrated components are the fundamental building blocks for service composition by reusing and combining simpler components to form complex ones. Access to an integrated component is controlled by two functional interfaces: the public interface openly expresses an integrated component’s functionality, whereas the private interface encapsulates how an integrated component performs the functionality expressed by the public interface. On other hand, the service composition

<sup>9</sup> <http://protege.stanford.edu/>

method allows us to create, compose, reuse, and transform eventually integrated components into executable processes. More details can be found in (Granell, 2006).

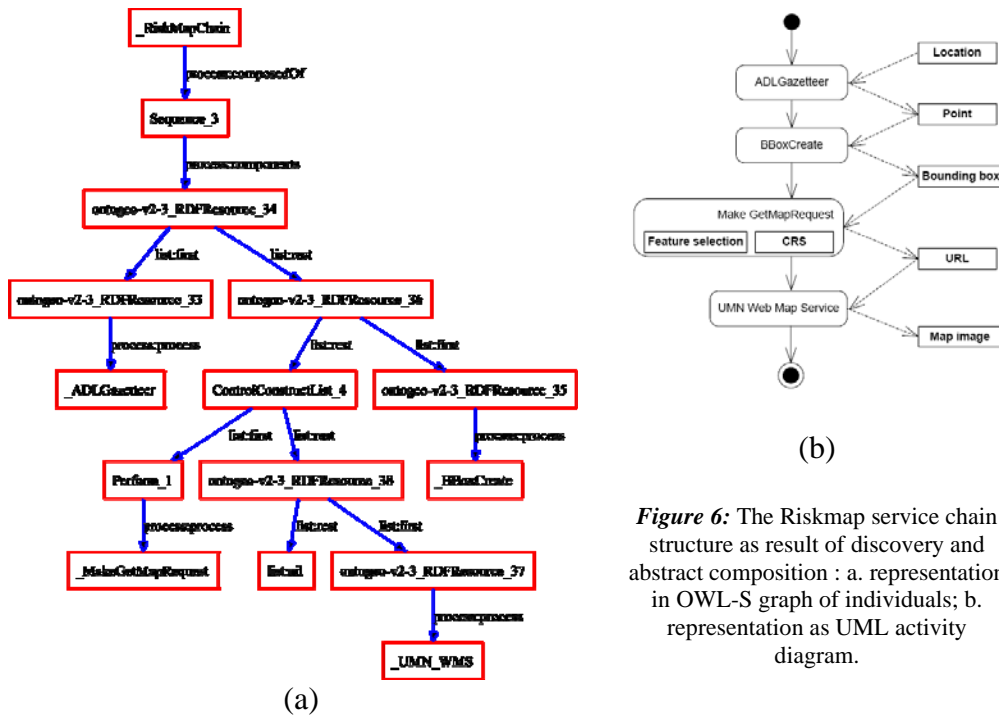


Figure 6: The Riskmap service chain structure as result of discovery and abstract composition : a. representation in OWL-S graph of individuals; b. representation as UML activity diagram.

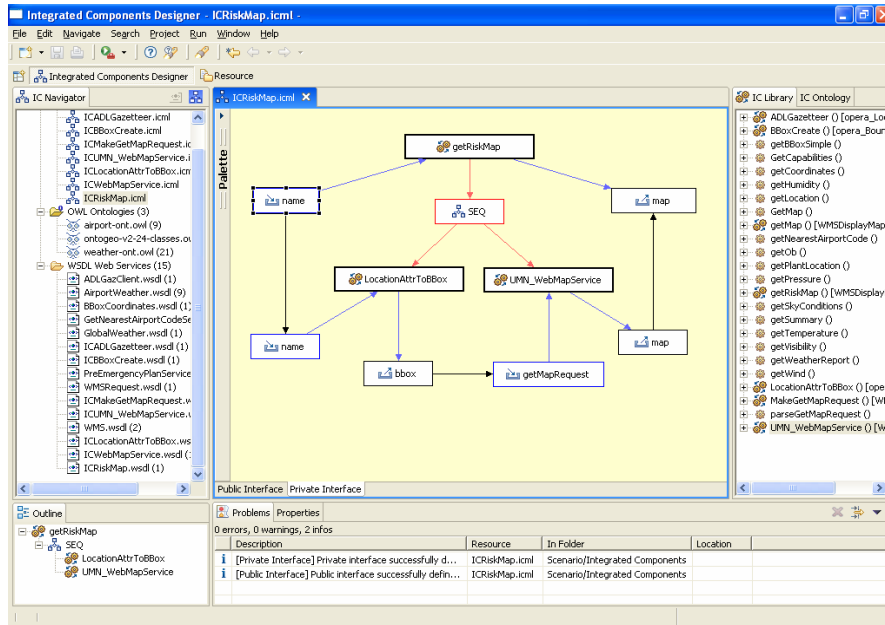


Figure 7: Snapshot of the target composition called GetRiskMap.

The centre and right hand side of Figure 4 depicts the concrete service composition and execution. Service discovery produces an OWL-S document that contains an abstract chain, i.e., a suitable web services list for composition (Figure 6b). As the service composition is carried out in terms of integrated components, the first step consists of creating integrated components from such a list. For that, we offer three different possibilities (Figure 4). The first one automatically creates the corresponding integrated component from an annotated WSDL file (Automatic IC Creation, Figure 4). The second possibility allows users to manually generate a new integrated component by annotating it with the concepts taken from shared geo-ontologies. In both cases, such new integrated composition is created from a single web service. Yet, as one goal of this work is to improve service reuse, the service discovery can also discover existing service compositions to be used in new compositions. In this case, the creation process is not necessary because the integrated component already exists. Therefore, the composition process (IC Composition box in Figure 4) is responsible for constructing complex integrated components by incrementally reusing existing ones taken from the repository.

Figure 7 depicts a snapshot of the Integrated Component Designer applied to the use case. In particular, it shows the private interface editor for defining the combination of integrated components to create the target `getRiskMap` composition. Indeed, such composition combines (reuses) two other integrated components already available —`LocationAttrToBBox` and `UMN_WebMapService`— by using a sequential pattern. Each of them is a composition itself. The former contains the first two services in the abstract chain, `ADL Gazetteer` and `BBoxCreate`, forming an intermediate composition that takes a city location as input and produces a bounding box. The latter integrates the last two services, `MakeGetMapRequest` and `Minnesota WMS`, encapsulating a full get map request. In this way, modelling integrated components by means of our composition method become the basis for service reuse (Granell et al., 2005).

The user might execute the desired composition through the transformation process (IC transformation in Figure 4). This process serialises an integrated component description representing our use case composition into a WSBPEL process document (WSBPEL, 2005). The right hand side of Figure 4 depicts the service execution, which takes the WSBPEL process and produces the risk map. We have used the Oracle BPEL Process Manager<sup>10</sup> as a workflow engine for service execution.

## DISCUSSION AND CONCLUSION

One of the strengths of the presented integrated approach is the use of common ontologies for the different steps in service chaining. Web-based ontologies provide a formal, yet flexible mechanism to describe web services. However, it is obvious that our GeoMatchMaker prototype does not support automatic discovery, rather than semi-automatic (human controlled) discovery. Another limitation lies in the exchange of workflow information between the prototypes. Currently, there does not exist a single common format that holds workflow elements, ontology concepts and WSDL parameters. However, this can be implemented by a relatively simple style sheet transformation in future work.

From our implementation experiences, the WSDL-S approach has been implemented with fewer effort than the OWL-S grounding. Although OWL-S supports the whole range of discovery-composition-chaining, there are fewer enactment engines for it, compared to other standards, such as WSDL and BPEL. From a practical point of view, a hybrid solution is therefore still preferred.

The main contribution of this work regarding service composition is to improve service reuse by defining a framework for designing, modelling, and reusing web services compositions through the concept of integrated components. Reusable services and compositions offer developers not only service description reuse but also knowledge reuse as taken from previous solutions and experiences applied to similar problems.

## ACKNOWLEDGEMENTS

The work described here has been supported in part by EU project SST4-2004-01225724 (AWARE) and Fundació Bancaja-Caixa Castelló.

## BIBLIOGRAPHY

- Akkiraju, R., Farrel, J., Miller, J., Nagarajan, M., Schmidt, M-T, Sheth, A., Verma, K., 2005 Web Services Semantics – WSDL-S, Technical Note W3C, April 2005. <http://www.w3.org/2005/04/FSWS/Submissions/17/WSDL-S.htm>
- Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., Eds., 2003 The Description Logic Handbook; Theory, Implementation and Applications. Cambridge University Press.
- Granell, C., Gould, M., Grønmo, R. Skogan, D., 2005 Improving Reuse of Web Service Compositions. In Proc. of the 6<sup>th</sup> EC-Web Conference, Copenhagen, Denmark. Springer-Verlag, LNCS 3590: 358-367.
- Granell, C., 2006 An Integrated Component-based Approach for Improving Service Reuse, PhD Thesis, to be published.
- ISO/TC211, 2003a Final Draft International Standard 19109 Geographic information - Rules for application schema International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics.

---

<sup>10</sup> <http://www.oracle.com/technology/products/ias/bpel/index.html>

- ISO/TC211, 2003b Final Draft International Standard 19119 Geographic information - Services International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics.
- Klien, E., Lutz, M. and Kuhn, W., 2006 Ontology-Based Discovery of Geographic Information Services - An Application in Disaster Management. *Computers, Environment and Urban Systems* Volume 30, Issue 1, January 2006, Pages 102-123.
- Lemmens, R. and Vries, M. de, 2004 Semantic Description of Location Based Web Services using an Extensible Location Ontology. In *Proceedings of Münster GI-days 2004: Geoinformation and mobility*, IfGI prints 22, pp. 261–276.
- Lemmens, R.L.G., 2006 Semantic Interoperability of Distributed Geo-Services, PhD Thesis, to be published.
- Martin, D.; Burstein, M.; Hobbs, J.; Lassila, O.; McDermott, D.; McIlraith, S.; Narayanan, S.; Paolucci, M.; Parsia, B.; Payne, T.; Sirin, E.; Srinivasan, N. & Sycara, K., 2004 OWL-S: Semantic Markup for Web Services, World Wide Web Consortium.
- Paolucci, M., Soudry, J., Srinivasan, N., and Sycara, K., 2004 A Broker for OWL-S Web Services. In *Proceedings First International Semantic Web Services Symposium 2004 AAAI Spring Symposium Series*.
- Szyperski, C., 1998 *Component Software. Beyond Object-Oriented Programming*. Addison-Wesley, New York.
- WSBPEL, 2005 OASIS Web Services Business Process Execution Language Version 2.0. Committee Draft, 01 September 2005. [http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=wsbpel](http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel)