Transactions in GIS, 2007, 11(6): 849–871

Research Article

Enhancing Geo-Service Chaining through Deep Service Descriptions

Rob Lemmens
Department of Geo-Information Processing
International Institute for Geo-Information Science and Earth Observation (ITC)
The Netherlands

Andreas Wytzisk
Department of Geo-Information Processing
International Institute for Geo-Information Science and Earth Observation (ITC)
The Netherlands

Rolf de By
Department of Geo-Information Processing
International Institute for Geo-Information Science and Earth Observation (ITC)
The Netherlands

Carlos Granell
Department of Information Systems (LSI)
University Jaume I
Castellón, Spain

Michael Gould
Department of Information Systems (LSI)
Universitat Jaume I
Castellón, Spain

Peter van Oosterom
OTB Research Institute, Section GIS-technology
Delft University of Technology
Delft, The Netherlands

Abstract
We demonstrate the integrated use of semantic and syntactic service descriptions, called *deep* service descriptions, for service chaining by combining two prototypes: one that deals with geoservice discovery abstract composition (called 'GeoMatchMaker'), with one that supports concrete composition and execution of geoservices services (called 'Integrated Component Designer'). Most other service chaining approaches confine themselves to handling either syntactic or semantic service descriptions. The proprietary formats of these descriptions hamper an effective integration of discovery, composition and execution of multiple services. In essence, service chaining should help a user by

Address for correspondence: Rob Lemmens, Department of Geo-Information Processing, International Institute for Geo-Information Science and Earth Observation (ITC), P.O. Box 6, 7500 AA Enschede, The Netherlands. Email: lemmens@itc.nl

providing an appropriate combination of executable services to solve a specified problem or query. Current XML-based service description languages, such as the Web Ontology Language-Services (OWL-S) and the Web Service Description Language-Semantics (WSDL-S), allow us to build a geoservice-reuse architecture based on common ontologies and shared service descriptions. Our approach uses annotation as a bridge between the syntax and semantics of services. This paper reports on its context and implementation issues. The target groups of this research are geo-information engineers who are confronted with information integration issues and service interoperability issues, and secondly, information engineers who in general are confronted with distributed information and with end users that need to access distributed services as one virtual application.

1 Introduction

As with all information system domains, GIS has recently been influenced to a large extent by Internet developments, resulting in an increasing availability of client/server applications using distributed geo web services. Web services are software systems that provide specific functionality to a group of clients over a computer network. Geo web services – often referred to as geo-services – are a special kind of web services that support the handling of geographic information and typically enable the user to derive new geo-information, based on spatial, temporal and thematic relationships. New challenges lie ahead to integrate these services into meaningful service chains (e.g. by using a shop locator of provider X together with a route planner of provider Y) and make them instantly available to the clients.

The integrated exploitation of those distributed services 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').

The evaluation of the fitness-for-use of a service or a service chain can be performed by executing the service(s) in pre-defined tests, but it is typically done first by interpretation of their descriptions. The discovery of a service involves finding a match between a service request and service advertisements. In case of more than one match, a service user may choose the best, based on human or machine inference of the properties of each match. There are two options:

1 The user finds a composite service that fulfils the task.
2 There is no single composite service that fulfils the task. The service request has to be decomposed, for which there are three options:
   (a) The user 'manually' performs the decomposition and repeatedly uses the discovery application for finding a match for each service part.
   (b) The discovery application performs the decomposition automatically.
   (c) The user performs the decomposition semi-automatically, i.e. by 'suggesting' service parts and using the discovery application to 'fill the gaps'.

We believe that all the abovementioned options require service descriptions to be standardised in an integrated framework of service elements.

This paper presents service descriptions that are based on commonly agreed rules for service characterisation. Such service descriptions are said to be committed to these rules *by contract*. Contracts that contain such rules may result in service interoperability and this can be achieved at syntactic and semantic levels. In this paper a syntactic service description represents the name and input/output parameters of a service, and its operations. A semantic service description represents the meaning of those parameters and the meaning of the functionality of a service. Service descriptions that refer to both levels are referred to here as deep service descriptions. By *deep* in this article we refer to the notion of deep annotation by Handschuh et al. (2003), in which HTML pages are not only described by scraping the presentation (the actual web page) but by exposing the structure and context of the database or information structure that formed the basis for creating the web page. Similarly, we provide an integrated framework in which semantic and syntactic service descriptions are addressed jointly, both in the service discovery and composition. Typically, semantic issues are treated in the discovery and syntactic ones in the composition process, leading to lower levels of service interoperability between those services found and those composed. Furthermore, such service descriptions take into account the specific characteristics of geographic phenomena that are handled by these services.

The remainder of the article is organised as follows. Section 2 provides some background and discusses our approach of service chaining and service description. Section 3 presents our proposal for a semantic interoperability framework as well as methods for ontology-based description of geo-services. Section 4 describes an integrated architecture and implementation approach for enhancing geo-service chaining through a risk mapping use case. Finally, Section 5 summarises the paper and provides suggestions for future work.

2 Service Chaining and Service Description

To deploy software applications that assist in service chaining, services need to be characterised in a formal way. However, current practice is that, apart from syntactic descriptions, if semantic service descriptions exist at all, they are written informally. This section provides the context of service characterisation and a method to formalise service descriptions.

2.1 Service Characterisation

To evaluate a service's fitness-for-use and to create meaningful combinations of services, it is necessary to model the essential properties of services that facilitate their discovery, composition and execution. A service can be characterised by the information it deals with (data input and output) and how it processes this information (the operations that the service makes available to the user). The abstraction of information and processes is an important instrument to simplify the view on the often complex structure of a service.

2.1.1 Information abstraction

The unambiguous integration of information requires the resolution of data heterogeneity between information resources, syntactically and semantically. Heterogeneity issues can

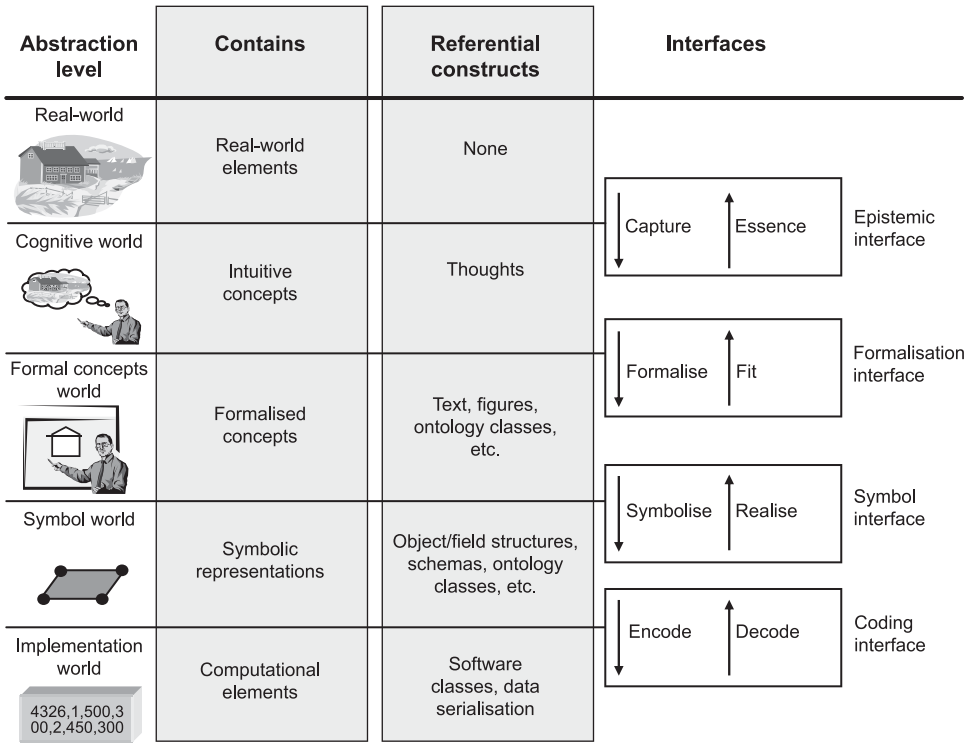

Figure 1 Information abstraction stack (following Kottman 1999 and Fonseca et al. 2002)

be better resolved when the semantics of underlying data models are made explicit. For this, publicly available (standardised) data models are considered to be important assets within a user community.

Information systems deal with different types of artifact to represent real-world phenomena. To disentangle the contextual elements, one often breaks down complexity by a separation of concerns. This can be accomplished by information abstraction. The distinction of layers for the purpose of complexity reduction has proven useful in traditional database design, where a conceptual level, a logical level and a physical level are identified. Similarly, in geo-information science, several models have been proposed, mainly driven by interoperability issues. Figure 1 shows a five layer abstraction model adapted from the five-universe paradigm, proposed in Fonseca et al. (2002). The nine layers of abstraction of the OGC feature model (Kottman 1999) is an even more refined model.

The top layer contains elements of the real-world that we are living in. In this layer, the elements are as they are in the real-world, and as soon as we describe them, these are becoming abstractions of reality located in one of the layers below. In the second layer, the cognitive world, the real-world elements are captured by intuitive concepts in human thought which are communicated by natural language. In the third layer, the formal concepts world, the intuitive concepts are formalised. This is typically accomplished by constructing a consensus-based model of concept definitions and concept

relationships, possibly supported with a domain ontology. The formalisation can be materialised in standards that are documented in text and model representations such as UML diagrams. In the fourth layer, the symbol world, formalised concepts are represented symbolically. Geographic features are represented according to the object/field model. The fifth and lowest level is the implementation world, containing computational elements materialised as software classes and data serialisation. In this stack, the term data set (or data) is distinguished from information. Information represents a real-world phenomenon at all levels below the real-world level. Data resides at the lowest level only and represents all the layers above. The practical implication of the layers can be understood as follows. A spatial data set resides at the lowest level, but the same data set represents the characteristics of real-world phenomena at all levels. A road, for example, can be partly characterised by: (1) its road class definitions such as 'primary road' (e.g. meaning a 12 m wide road) at the formal concept level; (2) the geometry that represents the road in the database (symbolic representation level); and (3) its data elements (implementation level). The examples given obviously do not describe the road in its entirety, i.e. a more complete description also contains the position of the road which is described by its coordinates with respect to a coordinate reference system.

2.1.2 Process modeling

The integration of services and the search for a single service requires the evaluation of each service's data characteristics (input, output and tightly-coupled data), classification of the service (in an agreed taxonomy) and may require the evaluation of its internal process structure (the latter exposes semantics of its functionality). Figure 2 represents relationships between static and behavioural entities. The static entities in the diagram (data, feature symbol and feature concept) reside in respectively the implementation world, the symbol world and the formal concept world as described in Figure 1. A service acts – at the implementation level – on the data as representation of real-world phenomena. However, the service may affect the semantics of the data, represented at higher levels in the geo-information stack.

In addition to the generic approaches of process modelling, a more specific analysis of tasks and operations in the geographic domain is needed. The need for formal modelling of geographic processes has been identified in numerous sources. In the past, several attempts have been made to create a classification of geo-operations, e.g. Albrecht (1995) and Chrisman (2002). In an effort to provide a basis for creating geo-service specifications, OGC and ISO have developed respectively the OGC service architecture (Percivall 2002) and the ISO 19119 standard for Geographic Information Services (ISO 2005b). In these specifications, geographic processes are viewed as service chains. ISO 19119 treats service chains as directed graphs and models them using UML activity graphs. To exchange service chain information with other users performing tasks in a similar situation, ISO 19119 provides a taxonomy of GIS operations and introduces a Service Organiser Folder (SOF) as a bag of unordered services to be used in a particular chain. None of the above efforts have resulted in comprehensive formal machine-accessible geographic process models that support semantic interoperability. This forms a part of the motivation to perform the research presented here.

In the context of the Open Geospatial Consortium (OGC) Web Service Common Specification (OWS) (Whiteside 2005), the recently released Web Processing Service (WPS) specification provides the definition of interfaces to integrate spatial and non-spatial

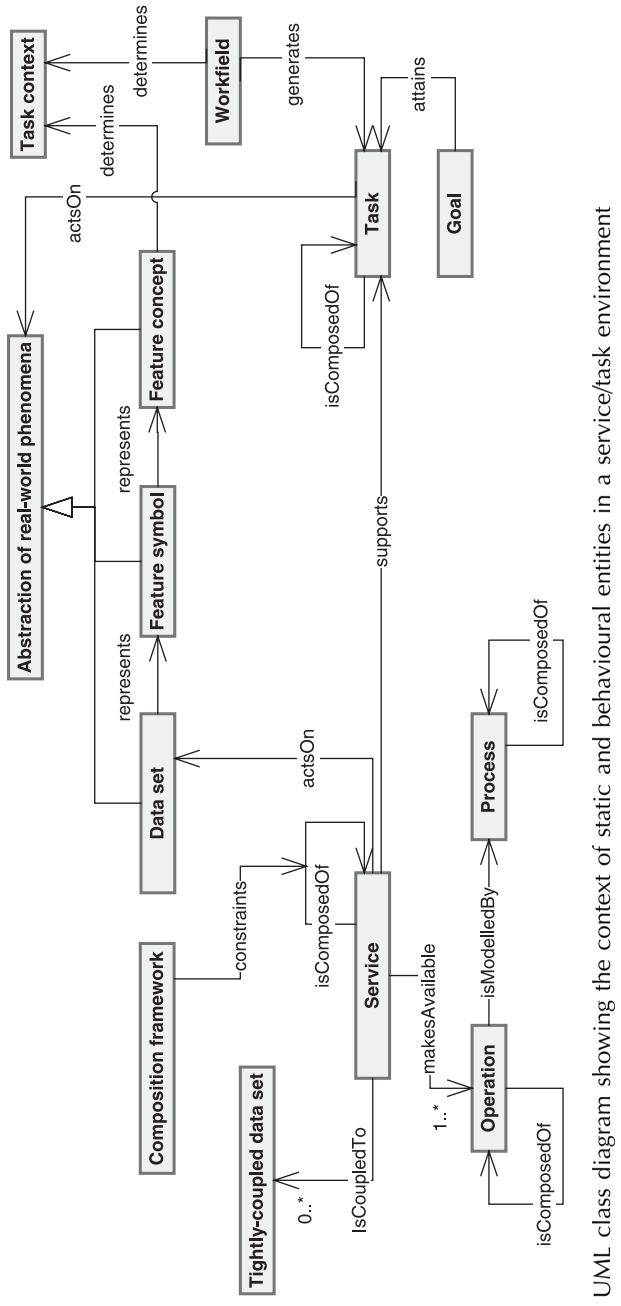

Figure 2 UML class diagram showing the context of static and behavioural entities in a service/task environment

operations and models in Geo Information Infrastructures (GII) containing OGC-compliant services (Schut and Keens 2005). The key aspect of WPS is that all services in compliance with WPS have to implement three common interfaces. The first interface *getCapabilities* is well-known to all OGC services, which describes the processes (operations) offered by a WPS service. The *describeProcess* interface shows details about input and output parameters of a specific process (operation) available in the *getCapabilities* interface together with an indication of the actual functionality of the process. The *execute* interface actually performs the service. This interface contains the implementation for concrete geoprocessing tasks. WPS is an interesting approach within the geoservice chaining context because the output of a WPS service can *a priori* be easily connected to the input of another WPS service to form complex geoservice chains. This can be done, for example, by properly using the three interfaces described earlier. Yet, the WPS specification is a discussion document still evolving and, at this moment, it provides no explicit support for service chaining. The work by Anderson and Moreno-Sanchez (2003) is a first attempt to create geo web services from OGC specifications. More recently, Friis-Christensen et al. (2006) and Kiehle (2006) have provided examples of geoprocesing applications using WPS on top of GII, yet both works present integrated clients accessing WPS services in a pre-established order instead of allowing users to chain *ad hoc* services discovered in catalogs. This is also a challenge we pursue in this paper.

2.2 Service Chaining

Service chaining is typically performed as a sequence of discovery, composition and execution (Figure 3). Sometimes, however, discovery, composition and execution are performed as an iterative procedure, e.g. part of the chain is discovered after another part has been composed or even has been executed. Further, the 'mode' of service chaining is influenced by other types of variations: (1) the degree of control a human user has on the discovery, composition and execution process; and (2) the fact that certain services in the chain have been prescribed. Service chaining has *atomic services* as basic building blocks (from an agreed service taxonomy) and creates composite services as a result (the resulting structure does describe a part of the semantics of the composite service).

2.3 Service Descriptions

The main goal of syntactic service descriptions, such as the Web Service Description Language (WSDL) (Curbera et al. 2002), is to describe interfaces of web services for invocation purposes. Our interest in WSDL is at this stage 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 OASIS Web Service Business Process Execution Language (WSBPEL) (Alexandre et al. 2006) which expresses in an XML-based language how a set of web services is to be invoked. Both specifications treat web services at the syntactical level, which is insufficient for creating meaningful descriptions of web services.

A semantic service description describes the meaning of a service's functionality and input and output. A semantic service description consists of the following characterisation elements of its geo-operation(s):

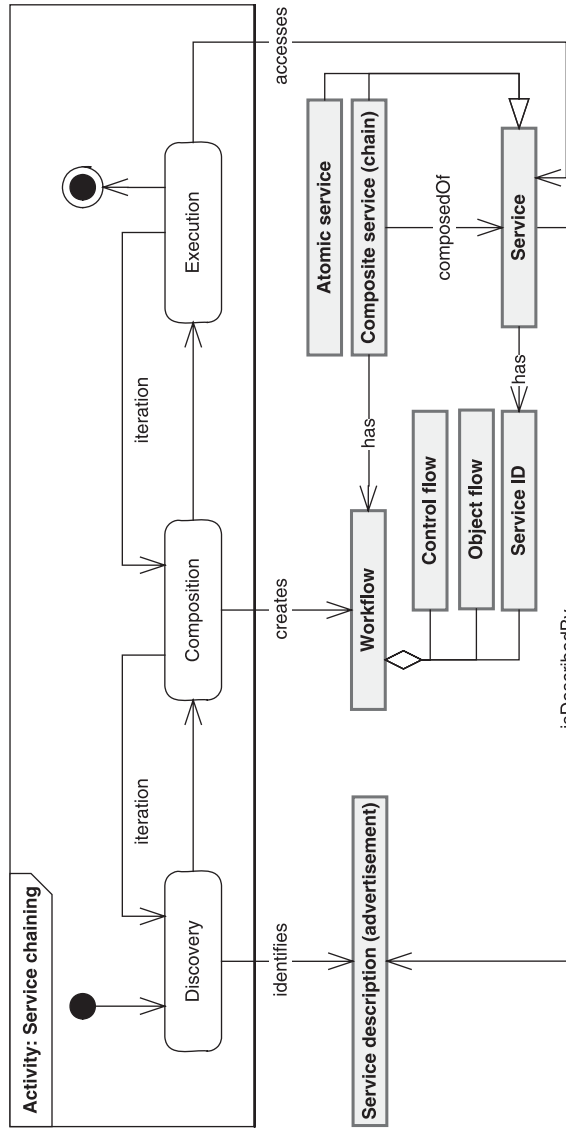

Figure 3 The elements of service chaining, depicted in the combination of a UML activity diagram (top) and a UML class diagram (bottom)

1. Description of each operation's input and output parameter types;
2. Description of geo-information that is tightly-coupled to each operation (such as the road network used in a route optimisation operation);
3. Classification of each geo-operation's functionality;
4. Description of the control flow in (virtual) composite operations (the initial building blocks are from the level of atomic operations); and
5. Pre- and post-conditions of the service.

In our approach, service descriptions are anchored in a so-called semantic interoperability framework. Semantic service descriptions are necessary during discovery and for the meaningful combination of two or more services. In this article, semantic service descriptions are proposed to be based on ontologies as knowledge structures of a specific application domain. These ontologies contain concepts about service input, output and functionality and they make part of the semantic interoperability framework as described in Section 3.

The relationship between information resources and ontologies is made through a process that we call *annotation*. It can be seen as the creation of meta-information, using ontologies as reference frameworks. In our case, the link between the abstract and concrete composition of services is realised by annotation, which connects ontology elements with parameters of executable code. We distinguish between two annotation approaches: OWL-S grounding and WSDL-S.

The first approach is based on the Web Ontology Language (OWL). 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. The first annotation approach, OWL-S grounding, is part of the OWL-S ontology. OWL-S does not explicitly describe the concrete I/O messages, but rather OWL-S grounding specifies 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, based on the combination of OWL-S process and grounding, allows one to control the interaction between web services (Paolucci et al. 2004).

The second mechanism for service annotation, WSDL-S, is based on the WSDL specification that solely represents the syntactical behavior 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. 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. In our approach, we have borrowed the *modelReference* attribute (among others present in the WSDL-S specification) to semantically annotate operations and parameters. Suppose a gazetteer service, for instance the Alexandria Digital Library (ADL) Gazetteer (http://middleware.alexandria.ucsb.edu/client/gaz/adl/index.jsp) which forms part of the 'RiskMap' service chain described in

Section 4.1. This service offers the operation *getCoordinates* that returns a location given a city name. Then, for example, the annotation for the operation *getCoordinates* (WSDL *operation* tag) refers to the concept *LocSpat* in the geo-operation ontology defined in the semantic interoperability framework (see Section 3), which formally defines an operation that returns a spatial attribute type, based on a location description. In the same manner, input and output parameters (WSDL *part* tags) are annotated with the concept *CityName* and *Point* respectively. The following WSDL-S snippet exemplifies the annotated *getCoordinates* operation of the ADL Gazetteer service.

```
<wsdl:message name="getMsgResponse">
    <wsdl:part name="coordinates" element="xsd1:ResponseType"
        wssem:modelReference="Ontology0#Point"/>
  </wsdl:message>
<wsdl:message name="getMsgRequest">
    <wsdl:part name="name" element="xsd1:RequestType"
        wssem:modelReference="Ontology0#CityName"/>
  </wsdl:message>
<wsdl:portType name="Gazetteer">
    <wsdl:operation name="getCoordinates"
        wssem:modelReference="Ontology0#LocSpat"/>
  </wsdl:portType>
```

Currently, no existing OGC specification deals with deep service descriptions (meaning those that include both syntax and semantics) in support of service (and data) discovery and chaining. The OGC Geo Semantic Web Interoperability Experiment (GSW IE) (Lieberman et al. 2005) has embarked on issuing semantic queries in relation to WFS, but not so much on discovery services in general. Einspanier et al. (2003) have identified the need for the integrated use of syntax and semantics in service chaining. A promising research for creating semantic web services in the geospatial domain is the ongoing European project SWING (Roman and Klien 2007), which aims to provide suitable tools for annotation, discovery, composition and invocation of geo-services by using the Web Service Modelling Ontology (WSMO) (Fensel et al. 2006). Other related research (Klien et al. 2006) addresses geographic ontology design, client interfacing, and reasoning in disaster management although mainly on the semantic aspects of service chaining.

3 Semantic Interoperability Framework

In this article we model the semantics of services with the help of a *semantic interoperability framework* (Lemmens 2006). A semantic interoperability framework is defined as the combination of ontologies, their relationships, and methods for ontology-based description of information sources (services, data sets, etc.). The framework serves the semantic interoperability between information sources.

At the crossroads of Semantic Web and web service applications, we find approaches that make use of explicitly stated semantics of web service characteristics. In such approaches, the so-called Semantic Web Services (SWSs) are considered to be semantically enriched and supportive of semi-automatic discovery, composition and execution. Currently, some prominent SWS approaches are OWL-Services (OWL-S) and

the Web Service Modelling Ontology (WSMO) as part of the Semantic Web Services Framework (WSMF) (Fensel et al. 2006).

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.

Generally, OWL-S covers a wide range of applications and WSMF's ontology WSMO is more focussed on e-commerce applications. For solving heterogeneity problems, WSMO uses mediators, which are special services defined for that purpose. In this research, OWL-S has been selected as the starting point. Although the actual processes, described with the OWL-S model can become rather complex, the model itself has a clearly defined structure. It is embedded in OWL and it is thus well rooted in the well-established theoretical foundation of Description Logics. Further, OWL-S is generally considered to be more adaptive than the other approaches and implementation independent because it does not prescribe use of specific services. Finally, at the time of implementing the prototype, OWL-S was the most mature of the SWS approaches and offered the necessary and sufficient constructs for modelling the characteristics of geo-services as targeted in this article.

At the basis of our proposed framework are three types of formal ontology, combined in a single ontology named *OnToGeo*:

- 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. The large box labelled NEN3610 in Figure 4 shows an example feature concept ontology, i.e. the NEN3610 data model (a Dutch geo-information model based on ISO 19100 standards).
- A **feature symbol ontology** formally defines the abstract elements that make up a feature in an object/field model, based on the ISO 19109 standard (ISO 2005a). This model distinguishes three abstraction levels, i.e. meta-level, implementation level and data level. In Figure 4, the two concepts above the NEN3610 box, are part of the feature symbol ontology. The feature symbol ontology also includes concepts on geometry, location and coordinate systems.
- A **geo-operation ontology** called *OPERA-R* 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, see Figure 5 (inspired by the ISO 19119 service taxonomy; ISO 2005b) and its typical input and output parameters. The parameter types are represented by symbol ontology elements and by specific operation classes, which are described in detail in Lemmens (2006).

Based on the above framework, *semantic queries* can be formulated in the form of OWL statements, such as:

$$R \equiv \textit{opera:LocSpat},$$

in which *R* is a so-called *requesting* concept, defined by the *LocSpat* operation. The name 'LocSpat' represents an operation type that reads a location attribute type (e.g. instantiated as an address type) and produces a spatial attribute type (e.g. instantiated

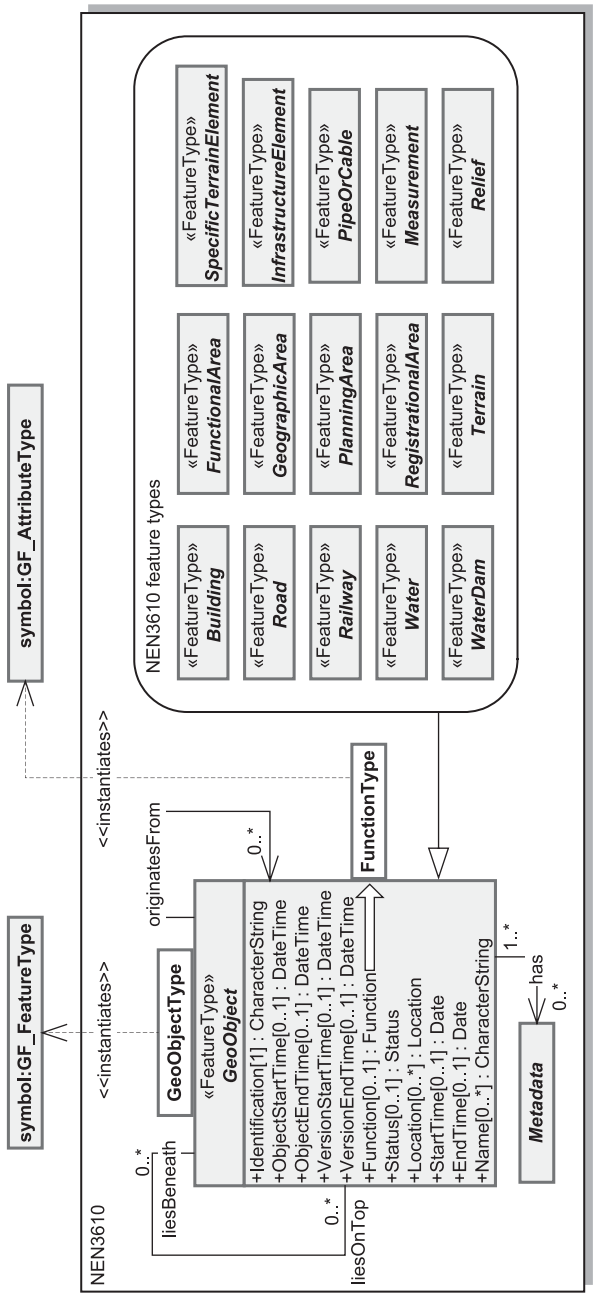

Figure 4 Overview of the NEN3610 data model. All classes in the dashed box are subclasses of *GeoObject*

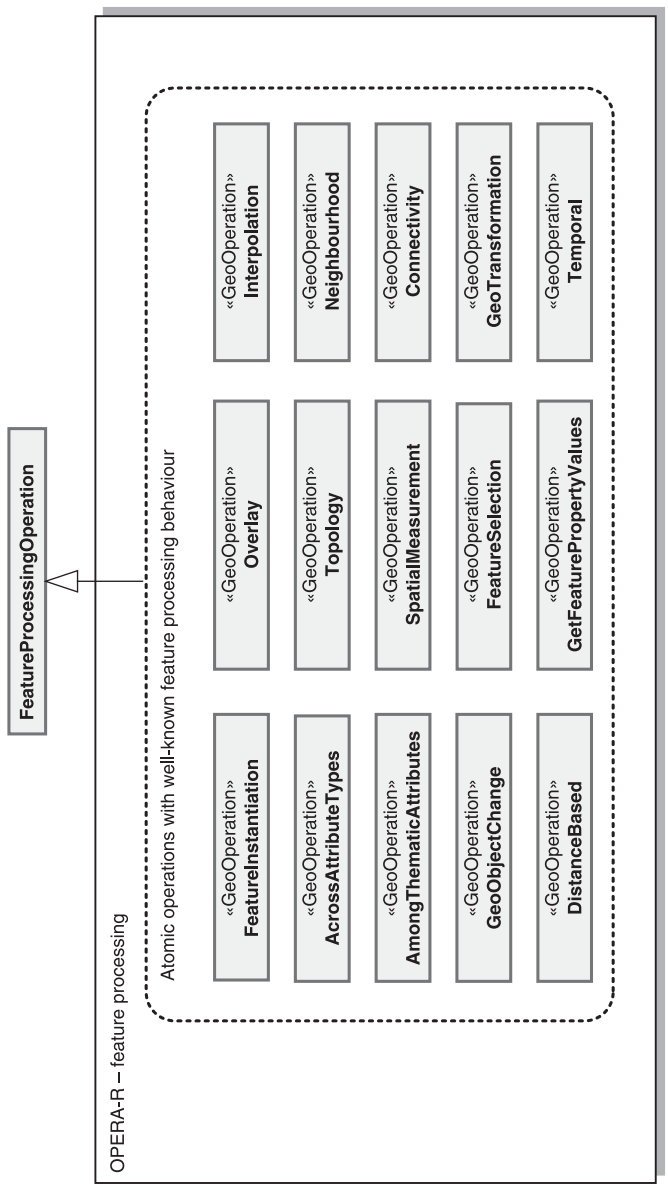

Figure 5 Feature processing operation classes in the OPERA-R geo-operation ontology

as a geometric object type), which is typically found in a gazetteer service. The *LocSpat* operation is a subclass of the *AcrossAttributeTypes* operation depicted in Figure 5.

The result of a semantic query is obtained by matching the requesting concept with other concepts in the ontology and by providing their semantic relationships (e.g. equals, sub-class, is disjoint with, intersects), with the help of an ontology reasoner.

4 Integrated Architecture and Implementation

This section presents our integrated approach for service chaining using syntactic and semantic descriptions (as illustrated in Figure 6). 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.

Figure 6 shows the integrated architecture of the combined prototypes as mentioned in the introduction. It implements the service chaining process as depicted in Figure 3. OnToGeo serves the interoperability between different processes as it is used for: (1) WSDL annotation; (2) discovery and abstract composition; and (3) concrete composition. Workflow documents form links between GeoMatchMaker, Integrated Component Designer and the workflow engine. New composite services that are created in the Integrated Component Designer are fed into the registry, ready for use in newly defined tasks. The thick arrows in the diagram indicate the workflow performed by an application user. In this case it shows the result of the workflow of a risk mapping use case that is used throughout this section. It is described in Section 4.1.

4.1 Use Case: 'RiskMap' Chain

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 called *RiskMap* 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 Service, implemented with MapServer WMS software (http://mapserver.gis.umn.edu/). 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.

4.2 Discovery and Abstract Composition

The GeoMatchMaker prototype has been developed to perform ontology editing and service chaining in one, integrated software application. It has been developed in the Eclipse Java developing environment. The development of GeoMatchMaker entailed the modification of the OWL-S editor of SRI International (Elenius et al. 2005) by providing it with a connection to the RacerPro reasoner and a simple user interface to interact with the reasoner. RacerPro is a knowledge representation system that can be used for reasoning with ontologies (Haarslev and Moller 2003). The prototype application is built

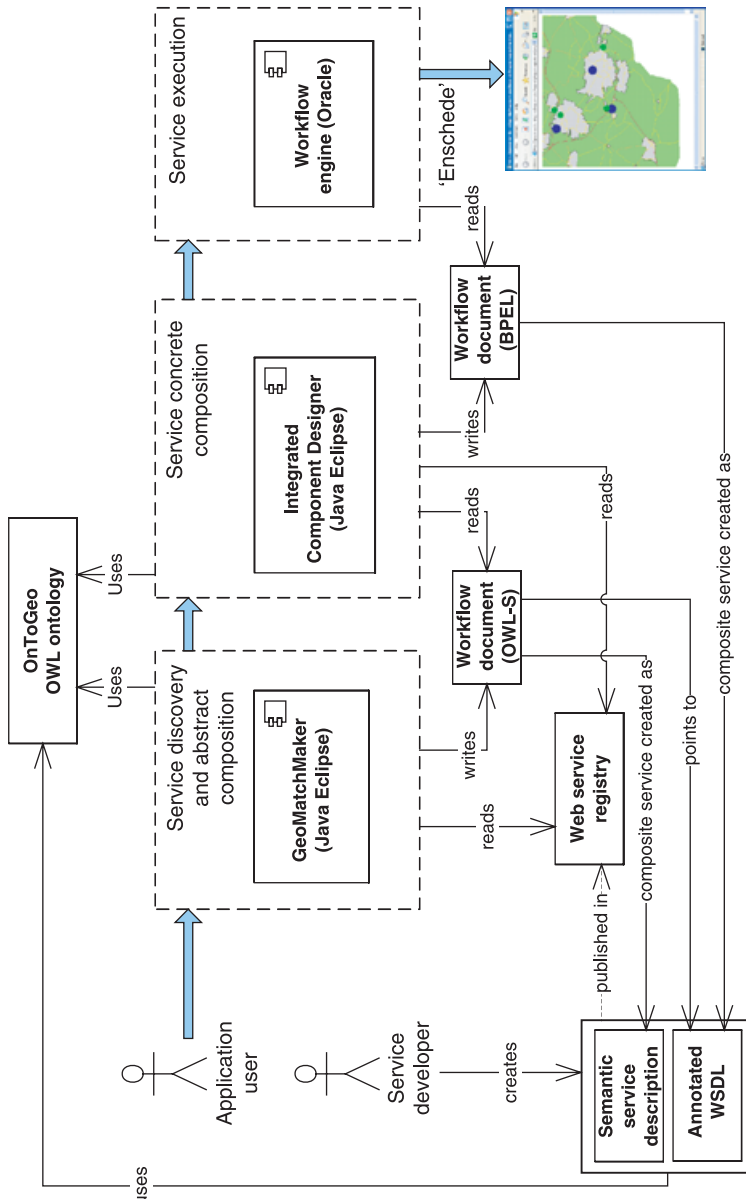

Figure 6 Integrated architecture for service chaining

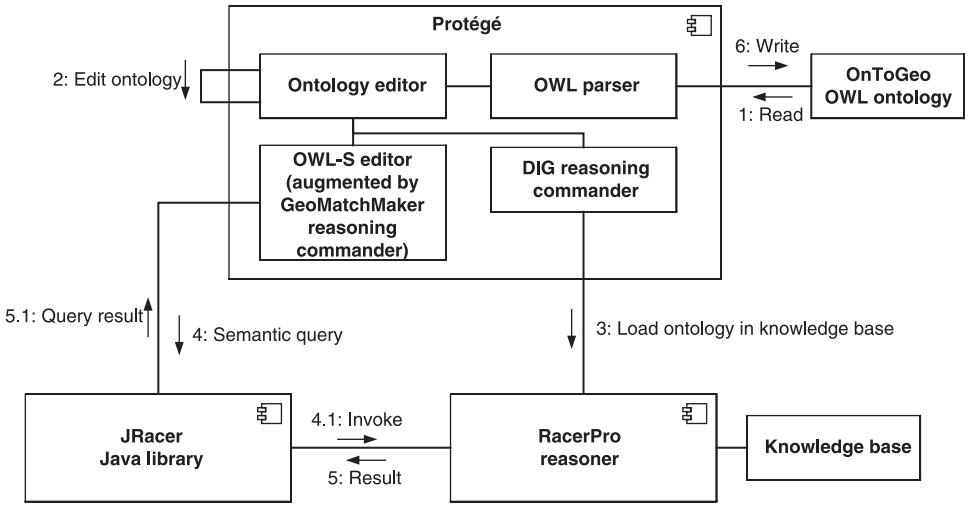

Figure 7 UML communication diagram showing the workflow of a semantic query in the prototype

in Eclipse from modified source code of the OWL-S editor and Java jar files of the Protégé ontology editor software (Knublauch et al. 2004). In this way, GeoMatchMaker behaves like Protégé, but with the additional functionality of reasoning with service chains.

The workflow of a matchmaking effort is presented in Figure 7. After reading the OWL document ontogeo.owl (step 1, loading the initial ontology database), a semantic query is formulated as a requesting concept R in step 2. In step 3, the ontology is loaded in the RacerPro knowledge base by invoking a procedure of the Protégé DIG reasoning commander. In step 4, the query is run by selecting the query instance of step 2 (the concept named *_ProbePrecedingWMSGetMap* indicated with an outline on the left-hand side in Figure 8) and subsequently selecting the newly created query option in the OWL-S editor's process window (see the boxed button with question mark in Figure 8). The JRacer command call invokes the corresponding RacerPro function (step 4.1) and RacerPro's response is captured (step 5) as a string (containing one or more ontology elements, in this case individuals). In step 5.1, the query result is displayed in a separate window (the central window in Figure 8). The window shows the following items:

- The selected request. In fact, this can be any request specified as a concept in the ontology, but, considered the purpose of the prototype, it will involve in practice a request for geo-information or for a geo-operation.
- The ontology elements (in this case individuals) found as instances of the requesting concept.
- Suggestions for a relaxed request. This is a rudimentary implementation by providing the superclasses of the originally selected request, which can be used to specify a new request in step 4.

In step 6, the ontology (with additional probe class) may be stored again as an OWL document. In addition, Figure 8 shows the adapted OWL-S editor GUI. When selected, a composite service is shown by the editor as a graph (right-side window). The menu

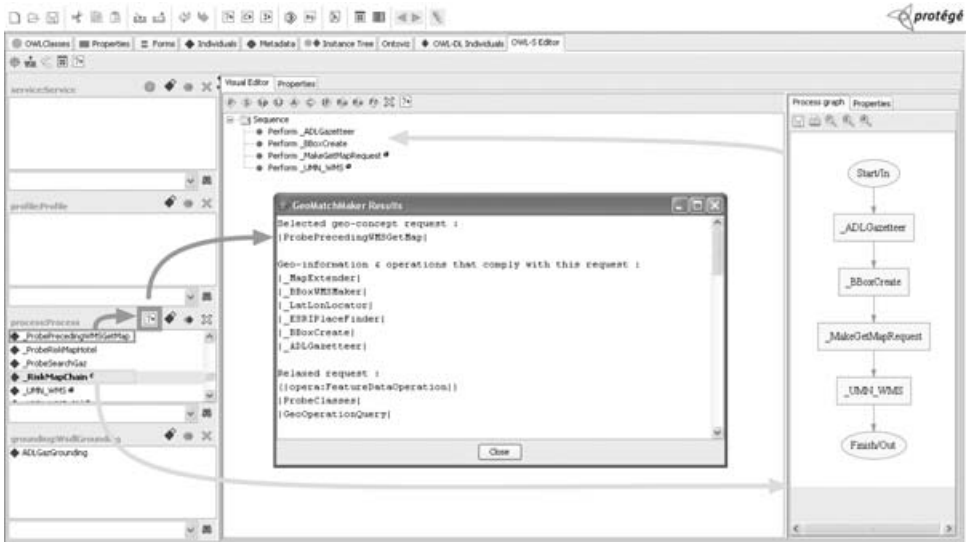

Figure 8 Screenshot of the OWL-S editor GUI, augmented by the GeoMatchMaker

bar of the visual editor's centre window contains diamond-shaped buttons that are used for the creation of the OWL-S control flow elements, such as sequence (S) and Repeat-until (Ru). The thick dark grey arrows represent the activation of the request for services that can be chained with the GetMap request of the MapServer Web Map Service, as presented in the use case. The thick light grey arrows point to the representation of the Riskmap service chain in the native GUI of the OWL-S editor.

4.3 Concrete Composition and Execution

In general terms, the concrete service composition approach presented in this paper stems directly from the definition of the component-based 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. Furthermore, the standard ISO 19119 (ISO 2005b) defines three design patterns for geographic service composition according to the degree of complexity of the resulting web service chain to the user: transparent or user-defined chaining; translucent or workflow-managed chaining; and opaque or aggregate. Alameh (2003) states that the translucent chaining pattern offers better benefits compared to other two patterns, as this pattern is midway between transparent and opaque chaining patterns, taking the best from these two patterns. That is the ease of use as 'one unit' (opaque), but also the added semantics because of 'inside view' (transparent).

Our composition approach is therefore influenced by these two ideas in the following way. From the component-based perspective, we provide the integration component concept (as the component model) and the service composition methodology (the composition method). Integrated components are the fundamental building blocks for service composition by reusing and combining simpler components to form complex

ones. An integrated component encapsulates descriptive, functional, structural and binding (how IO parameters are linked) aspects representing different but complementary views. For example, descriptive and functional aspects are useful for discovery because they express the functionality offered by a reusable service, while structural and binding aspects are critical for composition and execution. As the latter aspects are not necessary during the discovery process, they should be kept unknown to the user in this phase for simplicity and clarity reasons. To offer a suitable level of encapsulation, access to an integrated component is controlled by a public and private interface. The public interface openly expresses an integrated component's functionality, whereas the private interface is an internal view encapsulating how an integrated component performs the functionality expressed by the public interface. In terms of encapsulation the notion of integrated component is also similar to the translucent chaining pattern discussed earlier because it reduces the design complexity of geographic service chains to the user.

Workflow patterns proposed by van der Aalst et al. (2003) are important to fully define an integrated component (the private interface) because they show how an integrated component is internally organized as a combination of simpler web services or other integrated components. Workflow patterns play the role of composition operators facilitating the creation and composition of integrated components. More specifically, we have derived a set of abstract patterns from the original workflow patterns (van der Aalst et al. 2003) grouping them into two categories: selection patterns are in charge of the integrated component creation whereas composition patterns are involved during the composition of integrated components. Since integrated components by definition address reusability, the abstract patterns complement them providing flexibility (selection pattern) and structure (composition patterns). Users usually find service operations that have operation-level mismatches but perform the same functionality. For example, two service operations have the same functionality but may differ in the operation name or type of parameters. Abstract patterns provide then a solution to avoid mismatches of service operations by hiding them under the same integrated component operation due to use specific selection patterns. This lets us take advantage of the flexibility feature due to availability of multiple service operations for the same reusable service functionality. A detailed explanation of the pattern analysis and examples of selection and composition patterns can be found in Granell et al. (2005).

The workflow of the composition approach is presented in Figure 9. Service discovery (section 5.2) produces an OWL-S document that contains an abstract chain, that is, a suitable web services list for composition (step 1). As the service composition is carried out in terms of integrated components, the first step consists of creating integrated components from such a list by means of either the *IC Creation* process or *IC Composition* process (Figure 9). We offer two different possibilities for creating IC using the *IC Creation* process (step 1 top). The first one automatically creates the corresponding integrated component from an annotated WSDL or WSDL-S file. The second possibility allows users to manually generate a new integrated component by annotating it with the concepts taken from shared geo-ontologies (OnToGeo OWL ontology in Figure 6). In both cases, the resulting integrated component is created from single web services by specifying a selection pattern. As commented early, a newly integrated component has associated two descriptions. A private XML-based description contains how a service or services included in the integrated component are combined (selection patterns), and a public WSDL-S description is generated which will be used latter for the semantic discovery. Both integrated component's descriptions are registered in an *IC Repository*

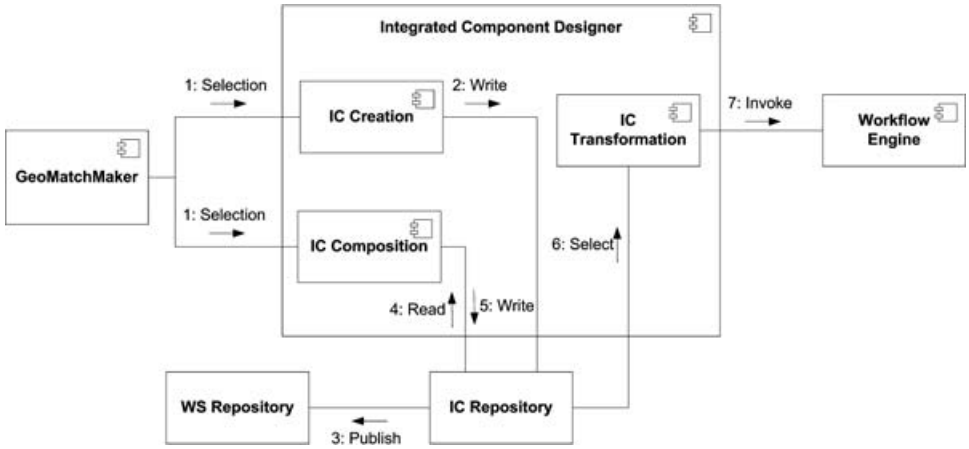

Figure 9 UML communication diagram showing the workflow of the integrated Component Designer in the prototype

(step 2). In addition, its public WSDL-S description is published in a public web service repository to be discovered by the service discovery process (step 3).

However, 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 by the *IC Composition* process (step 1 bottom). In this case, the creation process is not necessary because the integrated component already exists because indeed it is discovered. Therefore, the *IC Composition* process (see Figure 9) is responsible for constructing complex integrated components – choosing an appropriate composition pattern (sequence, parallel, loop, etc.) that depends on the business logic – by incrementally reusing existing ones taken from the repository and registering it as in the creation case (step 5 and 3).

Figure 10 depicts a screenshot of the Integrated Component Designer applied to the use case. In particular, it shows a graphical editor for defining the combination of integrated components to create the target 'RiskMap' composition (identified by the method *getRiskMap*). Indeed, such composition combines (reuses) two other integrated components already available – *LocationAttrToBox* and *UMN_WebMapService* – by using a sequential composition pattern. Each of them is a composition itself. The former contains the first two services in the abstract chain, a gazetteer service and a geometry-based operation for creating a polygon given its central point, forming an intermediate composition that takes a city location as input and produces a bounding box. The latter encapsulates a full *getMap* request reusing a couple of single services for making a valid *getMap* request.

The user might execute the desired composition through the *IC Transformation process* (see Figure 9). After selecting an integrated component form the *IC Repository* to be executed (step 6 in Figure 9), the IC Transformation process serialises such an integrated component description, representing in this case our 'RiskMap' composition, into a WSBPEL process document (Alexandre et al. 2006). The needed algorithms to transform integrated components descriptions into a WSBPEL process document are detailed in Granell et al. (2005). Finally, a workflow engine is fed with the generated

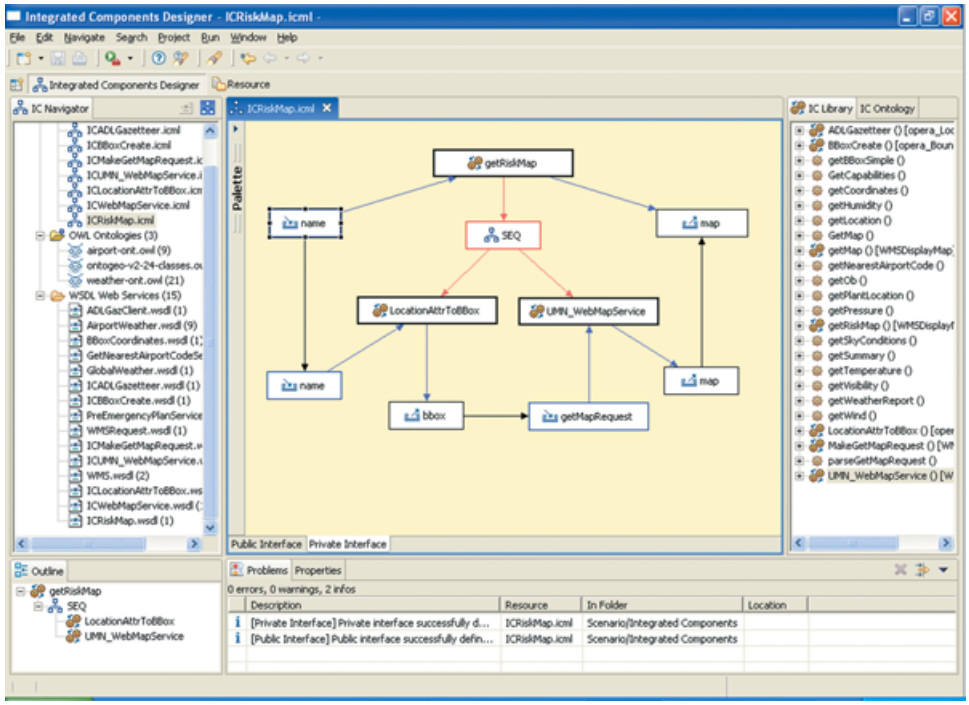

Figure 10 Screenshot of the Integrated Component Designer, showing a graphical editor for composing integrated components

WSBPEL process that produces the desired result to the user (step 7 in Figure 9). We have tested the resulting WSBPEL process document in a workflow engine like the Oracle BPEL Process Manager (http://www.oracle.com/technology/products/ias/bpel/index.html) as a workflow engine for service execution.

5 Conclusions

This article has presented an integrated framework for all facets of service chaining from service discovery to execution. The enhancement of the service chaining lies in the fact that the framework is based on so-called deep service descriptions, including both the service syntax and semantics. Most other approaches handle the two separately, resulting in an ineffective combination of discovery, composition and execution of multiple services. In our approach, ontology-based descriptions are used as representations of service requests and advertisements in a matchmaking process. The matchmaking is performed by an ontology reasoner, which can infer implicit relationships that exist in a knowledge base containing service descriptions as sets of concepts. The offered solution is flexible and extensible. The link between the abstract (conceptual) and concrete composition of services is realised by annotation, which connects ontology elements with parameters of executable code. From the service composition perspective, the integration and composition of geo-services is currently becoming critical for the rapid

evolution and use of geoinformation infrastructures. Therefore, one goal of the research carried out in this work has been to pursue the creation of repositories of reusable integrated components for developing complex and customised web applications based on the principles of reusability and integration of (deep) service descriptions.

From our implementation experiences, the WSDL-S approach has been implemented with less 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 WSBPEL. From a practical point of view, a hybrid solution is therefore still preferred. The deployment of the approach requires key organisations such as OGC to develop and maintain domain independent parts of a semantic interoperability framework and organisations with a GII mandate to manage its domain dependent parts.

A serious limitation of the current implementation of our approach is found to be the creation of the service meta-information (which resembles the problem of creating metadata in general). The creation of advertisements (by a data set/service provider) and requests (by a consumer) needs the development of more user-friendly graphical user interfaces (GUIs) and metadata management services (Missier et al. 2007). Although the Protégé ontology editor supports entering individuals with the help of rather user-friendly forms (similar to database entry forms), the implementation of ontology class-based descriptions is rather difficult. Moreover, a normal user cannot be expected to familiarise her/himself with an ontology editor. An interface is suggested with which a user can enter a query with help of menus, keywords and keyword suggestions that are drawn from the ontology.

Other issues for future work involve: (1) methods for automatically aligning and maintaining ontologies that different information communities use (ontology mapping); and (2) developing more sophisticated algorithms for service meta-information propagation and developing ranking mechanisms for semantic query results. In addition, context-aware service composition and discovery should be included in the integrated prototype, letting users discover and compose services according to their location or preferences.

References

van der Aalst W, ter Hofstede A H M, Kiepuszewski B, and Barros A P 2003 Workflow patterns. *Distributed and Parallel Databases* 14: 5–51

Akkiraju R, Farrel J, Miller J, Nagarajan M, Schimidt M-T, Sheth A, and Verma K 2005 Web Services Semantics (WSDL-S): Technical Note W3C. WWW Document, http://www.w3.org/2005/04/FSWS/Submissions/17/WSDL-S.htm

Alameh N 2003 Chaining geographic information web services. *IEEE Internet Computing* 7(5): 22–9

Albrecht J 1995 Universelle GIS-Operationen. Unpublished PhD Dissertation, Fachbereich Sozial-und Kulturwissenschaften der Hochschule Vechta

Alexandre A, Arkin A, Askary S, Barreto C, Bloch B, Curbera F, Ford M, Goland Y, Guízar A, Kartha N, Liu C K, Khalaf R, König D, Marin M, Mehta V, Thatte S, van der Rijn D, Yendluri P, Yiu A (eds) 2006 OASIS Web Services Business Process Execution Language Version 2.0. WWW Document, http://www.oasis-open.org/committees/documents.php? wg_abbrev=wsbpel

Anderson G and Moreno-Sanchez R 2003 Building web-based spatial information solutions around open specifications and open source software. *Transaction in GIS* 7: 447–66

Baader F, Calvanese D, McGuinness D, Nardi D, and Patel-Schneider P (eds) 2003 *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge, Cambridge University Press

Chrisman N 2002 *Exploring Geographical Information Systems* (Second edition). New York, John Wiley and Sons

Curbera F, Duftler M, Khalaf R, Nagy W, Mukhi N, and Weerawarana S 2002 Unravelling the Web services web: An introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing* 6(2): 86–93

Einspanier U, Lutz M, Senkler K, Simonis I, and Sliwinski A 2003 Toward a Process Model for GI Service Composition. In *Proceedings of Münster GI-Days*, Münster, Germany: 31–46

Elenius D, Denker G, Martin D, Gilham F, Khouri J, Sadaati S, and Senanayake R 2005 The OWL-S Editor A Development Tool for Semantic Web Services. In *Proceedings of the Second Annual European Semantic Web Conference (ESWC)*, Heraklion, Crete

Fensel D, Lausen H, Polleres A, de Bruijin J, Stollberg M, Roman D, and Dominque J 2006 *Enabling Semantic Web Services: The Web Service Modelling Ontology*. Berlin, Springer

Fonseca F, Egenhofer M, Davis C, and Camara G 2002 Semantic granularity in ontology-driven geographic information systems. *Annals of Mathematics and Artificial Intelligence* 36: 121–51

Friis-Christensen A, Bernard L, Kanellopoulos I, Nogueras-Iso J, Peedell S, Schade S, and Thorne C 2006 Building service oriented applications on top of a spatial data infrastructure: A forest fire assessment example. In *Proceedings of Ninth AGILE Conference on Geographic Information Science*, Visegrad, Hungary: 119–27

Granell C, Gould M, Grønmo R, and Skogan D 2005 Improving reuse of Web service compositions. In *Proceedings of the Sixth EC-Web Conference*, Copenhagen, Denmark: 358–67

Haarslev V and Moller R 2003 Racer: An OWL reasoning agent for the Semantic Web. In *Proceedings of the International Workshop on Applications, Products and Services of Web-based Support Systems, held in conjunction with the 2003 IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada: 91–5

Handschuh S, Staab S, and Volz R 2003 On deep annotation. In *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungary

ISO 2005a *ISO 19109: Geographic Information – Rules for Application Schema*. Geneva, International Organization for Standardization (ISO)

ISO 2005b *ISO 19119: Geographic Information – Services*. Geneva, International Organization for Standardization (ISO)

Kiehle C 2006 Business logic for geoprocessing of distributed geodata. *Computers and Geosciences* 32: 1746–57

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* 30: 102–23

Knublauch H, Fergerson R W, Noy N F, and Musen M A 2004 The Protégé OWL plugin: An open development environment for Semantic Web applications. In *Proceedings of the Third International Semantic Web Conference*, Hiroshima, Japan

Kottman C (ed) 1999 The OpenGIS Abstract Specification, Topic 5: Features, Version 4. Wayland, MA, OpenGIS Consortium Project Document No. 99-105r2

Lemmens R L G 2006 Semantic Interoperability of Distributed Geo-Services. Delft, Nederlandse Commissie voor Geodesie (NCG), Publications on Geodesy, New Series No 63

Lieberman J, Pehle T, and Dean M 2005 Semantic evolution of geospatial web services. In *Proceedings of W3C Workshop on Frameworks for Semantics in Web Services*, Innsbruck, Austria (available at http://www.w3.org/2005/04/FSWS/Submissions/48/GSWS_Position_Paper.html)

Martin D, Burstein M, Hobbs J, Lassila O, McDermott D, McIlraith S, Narayanan S, Paolucci M, Parsia B, Payne T, Sirin E, Srinivasan N, and Sycara K 2004 OWL-S: Semantic markup for web services. WWW Document, http://www.w3.org/Submission/OWL-S/

Missier P, Alper P, Corcho O, Dunlop I, and Goble C 2007 Requirements and services for metadata management. *IEEE Internet Computing* 11(5): 17–25

Paolucci M, Soudry J, Srinivasan N, and Sycara K 2004 A broker for OWL-S web services. In Cavedon L, Maamar Z, Martin D, and Benatallah B (eds) *Extending Web Services Technologies: The Use of Multi-Agent Approaches*. Berlin, Springer Lecture Notes in Computer Science No. 4504: 92–9

Percivall G 2002 *The OpenGIS Abstract Specification, Topic 12: OpenGIS Service Architecture, Version 4.3*. Wayland, MA, OpenGIS Consortium Technical Report

Roman D and Klien E 2007 SWING: A semantic framework for geospatial services. In Scharl A and Tochtermann K (eds) *The Geospatial Web: How Geo-Browsers, Social Software and the Web 2.9 Are Shaping the Network Society*. Berlin, Springer: 229–34

Schut P and Keens S (eds) 2005 *Web Processing Service (WPS) Interoperability Experiment: Final Report*. Wayland, MA, OpenGIS Consortium Interoperability Experiment No. 05-051

Szyperski C 1998 *Component Software: Beyond Object-Oriented Programming*. New York, Addison-Wesley

Whiteside A (ed) 2005 *OGC Web Services Common Specification Version 1.0*. Wayland, MA, OpenGIS Consortium