An object-oriented approach to GI web service composition

C. Granell, J.F. Ramos
Department of Information Systems
Universitat Jaume I
E-12071 Castellón, Spain
{canut, jromero}@uji.es

Abstract

In this paper we describe a novel approach to the incremental, semi-automated method for composition of web services in a geographical domain. First, we present the incremental component concept as a component that comprises both atomic and composite services under the same conceptual entity. Based on this concept, we describe an incremental methodology for composition by means of incremental component reuse. In addition, we present a prototype system in which this method is being tested in an emergency management scenario as part of the ACE-GIS project.

1. Introduction

The web services paradigm was designed to offer interoperability among diverse and heterogeneous applications [6]. The basic web specifications such as UDDI, WSDL and SOAP allow that services interfaces are described in a standard and uniform manner. A simple or atomic service is an Internet-based software component that does not rely on other web services to fulfill user requests [12]. A wind information service, a Web Map Service (WMS) or Web Feature Service (WFS) can be considered atomic web services. In contrast to atomic services are complex services, which are built by combining existing atomic or complex services working together to offer a value-added service. For example, as part of the pilot application in the ACE-GIS European project (www.acegis.net), an emergency complex service can be developed by composing several atomic services (see section 4). In this paper, we denote an *atomic service* as a single service and a *composite service* as a service that employs other services. The services included in a composite service are its *contained services*.

Due to their platform and language-neutral nature, web services provide the ability to chain other web services in order to offer value-added functionalities that are then invoked using web technologies [5]. However, to enable creation of a value-added composite service, it is necessary to provide a certain level of interoperability among contained services. In the geographic domain, Open GIS Consortium (OGC) has launched recently the Open Web Services initiative phase 2 (OWS-2) to address interoperability requirements and to enhance standards that enable discovery, access and use of geographic data and geoprocessing services [13]. In this paper we discuss a novel approach for ensuring the functionality of each contained service, in order to go beyond one-to-one conformance to a standard, toward true interoperability among diverse web services. We show an approach for composing web services in the geographical domain, based on *incremental component* concept and object-oriented aspects, and present the first steps toward composition and invocation of diverse OGC services in an emergency scenario.

2. Web service as *incremental component*

Currently, compositional languages for web services are process-oriented in the sense that they attempt to define an entire workflow graph consisting of several services and their connectivity. This approximation lacks certain benefits of the object-oriented perspective, such as encapsulation, reuse, or scalability. For example, suppose a composite service that comprises several atomic services. In the case that we desire to reuse only a part of the composition, we need to create a new composition (and the associated workflow script) according to these new requirements. In BPEL [3], there is no possibility of reusing contained processes because those do not own associated WSDL descriptions.

To enable an object-oriented perspective for service composition, we need to redefine the concept of web service taking into account the object-oriented

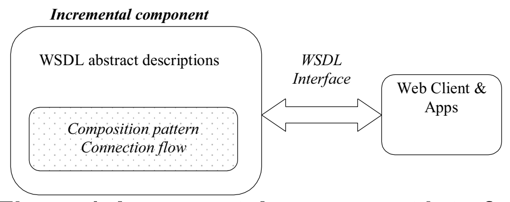

Figure 1. Incremental component interfaces.

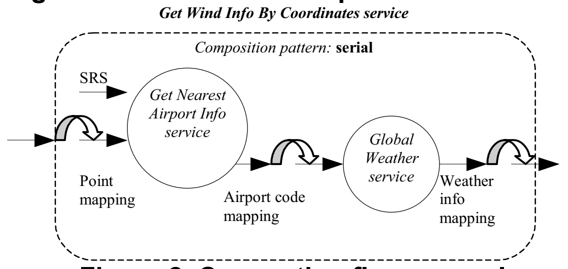
Get Wind Info By Coordinates service
Composition pattern: serial
SRS
Get Nearest Airport Info service
Point mapping
Airport code mapping
Global Weather service
Weather info mapping

Figure 2. Connection flow example.

characteristics aforementioned. Therefore, we describe the *incremental component* concept, similar to the traditional component software concept [14], which exposes service interface descriptions together with associated business logic as a single conceptual entity with which to interact. This component can be combined, searched, invoked or used as any normal web service. Analyzing this definition, the *component* concept provides interesting benefits [14]. First, it is an independent unit since encapsulates its constituent features and behavior from others components. Second, independence implies that these component are reusable, that is, can be composed in other compositions providing high level scalability. Finally, an incremental component has no persistent state because it is just a declarative description. The qualifier *incremental* taken here refers to the incremental methodology for service composition (see section 3).

In order to provide this duality, the incremental component provides two functional interfaces (see Figure 1). On one hand, the incremental component contains an abstract service interface that exposes publicly the service behavior. This *public interface* is described in WSDL, thus it (incremental component) can be treated as a standard web service by other applications or web clients. On the other hand, the incremental component also presents a *private interface* (dotted box) that specifies the underlying business logic of the composition, in which are described a list of contained services, their orchestration (execution order of contained services), and the connection flow established for the data dependences between these contained services. In other current proposals, such as the BPEL composition language, the service interface descriptions and associated orchestration are held separately. However, in the incremental component, both service interface description and associated orchestration are encapsulated together forming a unique entity. According to our model, all *composite services* are *incremental components*, such that these terms are interchangeable. In addition, *atomic services* can be considered *incremental components* without private interface.

The *public interface* provides abstract WSDL definitions of the composite service itself, without details of concrete implementation: things such as access protocols or location points. In particular, messages, operations, portTypes and, if necessary, types and schemas are described in this interface. Note that the WSDL interfaces of the contained services are not specified in the composite service, providing a high degree of encapsulation. The optional section of types and schemas defines the XML schema used by the composite service. We provide reuse and extension of existing schema defined in the contained services. In this sense, input and output XML schemas of the composite service are created from any contained service schema, if necessary. For example, suppose that a complex type declaration called Point exists in a Get Nearest Airport Info service (see Figure 2). If the Point declaration is included necessarily in the Get Wind Info By Coordinates service, its XML schema declaration is kept in the composite service named by its fully qualified name, avoiding collisions between type declarations with the same name yet disparate contents. At this moment, semantic aspects related to the schemas reuse are not considered.

In the *private interface*, encoded declaratively in an XML WSDL extension, are specified the business logic in order to define the interactions between the contained services comprising the *composition pattern* and the *connection flow* descriptions. On one hand, the composition pattern description indicates the contained services orchestration, whether they are invoked in a serial or parallel manner. In addition, it also includes a list of contained services. On the other hand, the connection flow description is charged with establishing data flow between the contained services themselves as well as data dependences between the composite service and the contained services. Figure 2 illustrates the four kinds of mapping mechanisms defined in the connection flow section. The *input mappings* connect the input parameters of the composite services with the input parameters of its contained services. In contrast to input mappings, the *output mappings* connect the output parameters of the contained services to output parameters of the composite service. The data mappings between contained services are piped by *internal mappings*. Finally, a special case of internal mappings are *constant mappings* in which a simple constant value is assigned at design time to an input parameter belonging to both the composite service and the contained services. For instance, the SRS parameter used in the Point type is specified as a constant value and it is not visible from input message of the Get Wind Info By Coordinates service. In the three first mappings, messages among services are mapped by means of XPath [7] queries.

To increase the interoperability between services, we define the concept of *incremental component*. Its *public interface* provides a WSDL interface, while the *private interface* gives enough clues to construct a composite service in terms of the contained services. This high-level view of a service allows the encapsulation of the underlying service complexity and reduces the entire graph complexity since dependences between services are also reduced. In addition, there is no need for a workflow script as in other proposals, since the incremental component together with the incremental composition process (see next section) provides a straightforward means for reusing compositions in future compositions.

3. Composition of incremental components

The composition model presented here provides an *incremental methodology* for web services composition, which is centered around descriptive aspects, on interoperability and on scalability. This model, using the incremental component concept defined previously, provides a high-level approach for describing interactions between web services, taking into account an object-oriented perspective (in contrast to process-oriented approach) which provides encapsulation, reuse and scalability characteristics. According to the conceptual architecture that supports the incremental composition model [10], we establish two functional views of the system – composition and invocation view – in order to achieve the composition and invocation of services.

The composition process is responsible for creating the desired composition in terms of incremental components. The outcome of this process is the composite service seen as a logical composition graph given the user requirements. The main principle consists of decomposing complex compositions into basic patterns [1] of pairs of incremental components. In this manner the desired composition is constructed in multiple iterations of composing two at a time. While most current composition languages allow construction of several services simultaneously [2], in our geographic service context – indeed, in many use cases – it is not realistic that a user would combine many WMS at the same time. Therefore, in principle, we prefer a simple and consistent manner to create compositions instead of increasing the cardinality of services composition during a single graph definition. In this sense, the incremental approach taken here is more cautious because the developer moves ahead pair by pair, in an *incremental* mode, assuring herself along the way that the composition is functional at each step before moving ahead. For example, figure 2 depicts a couple of atomic web services: Get Nearest Airport Info and Global Weather. Our goal is to obtain a composite service called Get Wind Info By Coordinates from the serial composition of these atomic web services. A user specifies the private interface provided by the composite service while the public interface is automatically generated. In this case, serial is assigned to composition pattern and diverse data mappings are established for the messages involved in linking this pair of services aligning with user requirements. The process continues forming new composite services from other atomic or composite services. Finally, the resulting composition is actually a composition graph, as logical relations between services, which is defined implicitly by the very structure of the composition. Therefore, the private interface provides a means for reusing and composing services in a controllable and consistent manner.

Once the composition is created, the invocation process is charged with analyzing the private interface of the composite service to encode the logical composition graph as a tree structure ready for execution. This graph may be thought of as a tree structure, whose leaf nodes are linked to atomic services, and whose internal nodes represent virtual intermediate compositions, and where the root node represents and encapsulates the entire composition. Compositions are virtual in the sense that each one functions independently, without the need to know that it is being used by other services. In addition, each internal node (or composition) does not exist physically as an atomic service since these compositions only contain abstract descriptions (public interface). For this reason, a composite service can be considered as a logical or virtual service.

The algorithm for tree traversal has been developed on the basis of the depth-first search, in which extremes (leaf nodes) are searched first, while paying attention to the *composition pattern* description. The *composition pattern* (serial, parallel) dictates how to

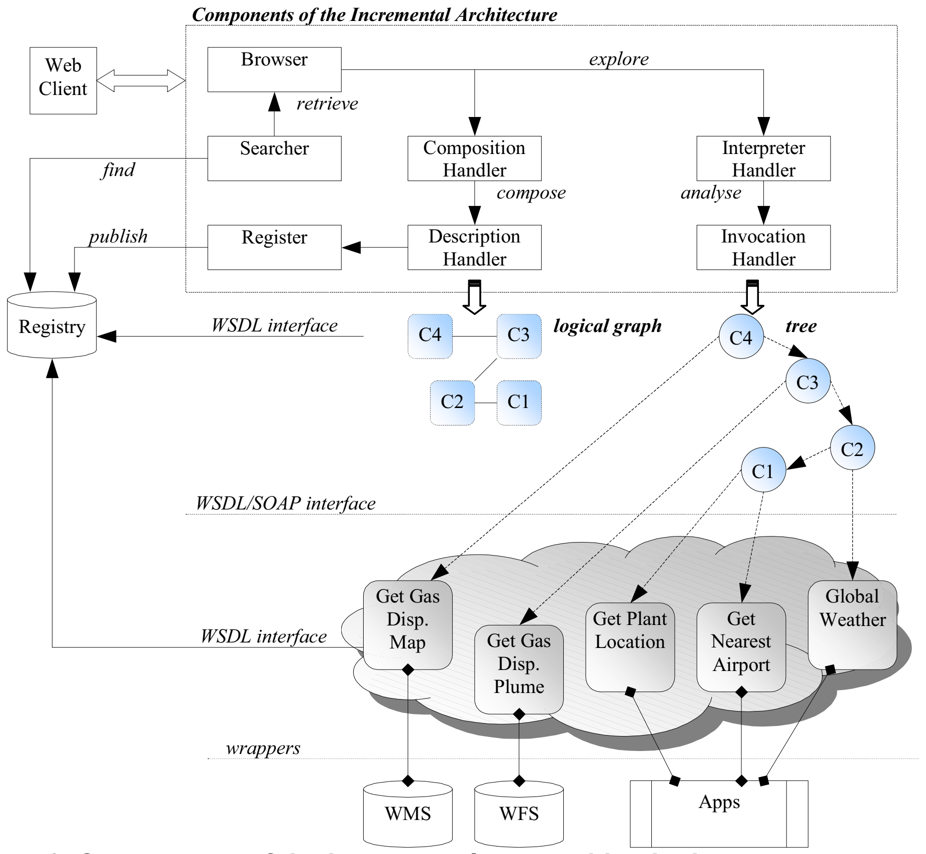

Figure 3. Components of the incremental composition in the emergency scenario.

traverse the tree invoking directly the atomic web services found and, chaining the results obtained according to the *connection flow* description defined in the private interface of each composite service visited. Common with other proposals for compositional languages is that invocation through the incremental composition model achieves a dynamic binding with concrete web service instances at execution time. The access mechanism to a service is not known until the moment of invocation. In this way, the composition using the abstract interfaces provides maximum flexibility compared to binding at design time.

4. Emergency use case

In this section we present an emergency pilot as use case to investigate the interoperability among diverse and more complex web services, and to test both the *incremental component* concept and the *incremental model* presented here. This use case is one of the pilot applications proposed in the ACE-GIS project. The emergency pilot [11] deals with accidents involving toxic gas releases from a chemical plant located at an arbitrary location.

The proposed architecture is shown in Figure 3. The *Registry* contains both atomic service and composite service descriptions according to WSDL. The *Composition Handler* and the *Description Handler* form the core of the composition process, and they build the logical composition graph as a logical sequence of services that represent the final composition. For instance, C1 service represents the serial sequence of Get Plant Location and Get Nearest Airport; in turn C2 is forming by C1 and Global Weather service. The process continues until we obtain the emergency service (C4). Note that C1 to C4 are incremental components, so C2 can only access to the public interface of C1 service, encapsulating the C1 service complexity. In fact, the dependences between the contained services are drastically reduced due to the public and private interfaces defined in the incremental component.

In the invocation process composite services are selected and invoked. The *Interpreter Handler* analyses the selected composite service. This module is charged with generating the tree structure from the logical composition graph. This virtual graph is encoded by the inherent structure of the composition, that is, the tree structure is generated by interpreting the private interface of each composite service involved in the composition. Finally, the *Invocation Handler* traverses it in order to execute the entire composition.

5. Conclusions and future work

We have presented a novel approach for composition and invocation of web services within the context of the ACE-GIS project, on-going research on geographic web services interoperability. This approach is aimed at becoming a first step in resolving some of the key interoperability problems among GI services.

First, in order to introduce the object-oriented perspective in service compositions, we presented the concept of *incremental component*, which comprises the idea of atomic and composite service under the same entity. In addition, service interfaces and their associated business logic are also encapsulated, forming a unique conceptual entity. As a distinct benefit over known composition languages, this approach does not define a language or a small set of well-known compositional languages. Just as the web's success can be attributed to its simplicity [8], we propose a process in which services are composed in an incremental and controllable manner, avoiding the need for a compositional or flow language to describe complex graphs of multiple service interactions, based only on appropriate composition patterns definition. This incremental composition is specified declaratively in XML WSDL extensions, as an aggregation of public interfaces, that is, the external behavior of the contained services as well as the private interfaces containing the composition pattern and connection flow. Future work will include the dissemination of the incremental model components under a open software license and the presentation of the WSDL extensions (private interface) of the component service to relevant standards bodies, beginning with OGC and OASIS, in order to improve its general acceptance and use.

Thus far we have developed a prototype capable of guiding the user in the creation of compositions formed by an arbitrary number of existing web services. In addition, this prototype is also capable of dealing with any XML-Schema's user-defined data type interchanged between services, by means of data mappings, such as in the use case of the emergency scenario.

Future work concentrates on assuring that the resulting service composition is somewhat fault tolerant with regard to service availability and quality of service. In this line, we are investigating new composition patterns [1] in order to support dynamic composition and service changes in the composition at execution time [4, 9], so that the composition will be able to detect service faults and replace faulty services with substitutes meeting user requirements.

6. Acknowledgements

The work described here has been supported in part by European Union project IST-2001-37724 (ACE-GIS) and by Spain Ministry of Science and Technology grant TIC-2000-1568-C3. Feedback from ACE-GIS project partners has been especially helpful.

References

[1] W.M.P van der Aaslt *et al.*. "Workflow Patterns". *Distributed and Parallel Databases*, 14(1):5-51, 2003.
[2] A. Aissi, P. Malu, and K. Srinivasan, "E-business process modeling; The Next Big Step", *IEEE Computer*, 35(5):55-62, 2002.
[3] T. Andrews *et al.* (eds.). Business Process Execution Language for Web Services Version 1.1. May 2003. http://www.ibm.com/developerworks/library/ws-bpel/.
[4] M.S. Akram, B. Medjahed, and A. Bouguettaya. "Supporting Dynamic Changes in Web Service Environments", In *Proc. of First ICSOC.* Springer-Verlag, pages 319-334, Trento, Italy, 2003.
[5] N. Alameh, "Chaining geographic information web services", *IEEE Internet Computing*, 7(5):22-29, 2003.
[6] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*, Berlin, Springer-Verlag, 2003.
[7] J. Clark and S. DeRose, XML Path Language (XPath) Version 1.0. http://www.w3.org/TR/xpath/, 1999.
[8] N.J. Davies, D. Fensel, and M. Richardson. The Future of Web Services. *EBT Technology Journal*, 22(1):118-130, 2004.
[9] S. Ghandeharizadeh et al., "Proteus: A System for Dynamically Composing and Intelligently Executing Web Services". In *Proc. of the First Intl. Conference on Web Services*, Las Vegas, Nevada, June 2003.
[10] C. Granell, J. Poveda, and M. Gould. "Incremental Weak Composition and Invocation of Geographic Web Services". In *Proc. of the 2nd Intl. Workshop on Semantic Processing of Spatial Data*, pages 179-187, Mexico, 2003.
[11] C. Granell, J. Poveda, and M. Gould. "Incremental Composition of Geographic Web Services: An Emergency Management Context". In *Proc. of the 7th Conference on Geographic Information Science,* pages 343-348, 2004.
[12] B. Medjahed *et al*. "Business-to-business interactions: issues and enabling technologies". *The VLDB Journal*, 12(1):59-85, 2003.
[13] OGC, OGC Web Service Phase 2 (OWS-2). http://www.opengis.org/initiatives/?iid=7, 2004.
[14] C. Szyperski. *Component Software. Beyond Object-Oriented Programming*. New York, Addison-Wesley, 1999.