An Incremental Approach to Web Service Composition *

Carlos Granell † José Poveda ‡ Michael Gould §

Abstract

Geographic web services will soon become subsumed in the e-commerce world, a world where the composition of simple or atomic services to build compound services is a key characteristic. Since currently no detailed model of composite geographic web services exists, and within the scope of work on a European Union-funded project, we define such basic composition as part of an effort to test for geographic web service interoperability. Rather than adopting current static methods, we build on the concept of incremental composition and provide a model for defining, composing and invoking compositions in a flexible manner. We demonstrate a prototype application for this purpose, and illustrate its utility through a simple arithmetic function composer scenario in which complex arithmetic expressions could be easily evaluated in base of the composition of atomic services. Benefits of this incremental composition model over process-oriented alternatives are mentioned, and necessary semantic and other extensions are outlined.

Keywords: *Web service composition, incremental composition, semantic interoperability, interoperability testing.*

Resumen

Los servicios web geográficos se incorporán en un futuro próximo al mundo del comercio electrónico, donde la composición de servicios compuestos a partir de servicios simples o atómicos se convertirá en un instrumento esencial. Actualmente, no exite un modelo concreto para la composición de servicios web geográficos, y dentro del marco de trabajo de un proyecto europeo, definimos las bases de esta composición como parte de las pruebas de interoperabilidad de servicios web geográficos. En lugar de adoptar los métodos estáticos disponibles, hemos definido el concepto de composición incremental y un modelo para definir, componer e invocar composiciones de una manera flexible. Para dar forma a este modelo, hemos desarrollado un prototipo que demuestra su capacidad con un sencillo escenario para la composición de funciones aritméticas, en el cual expresiones aritméticas complejas pueden ser evaluadas con facilidad mediante la composición de servicios atómicos. Además, mencionamos las ventajas de la composición incremental frente otras aproximaciones orientas al proceso así como la necesidad de incorporar al modelo otras extensiones adicionales como puede ser la semántica.

Palabras clave: *Composición de servicios web, composición incremental, interoperabilidad semántica, pruebas de interoperabilidad.*

*The work described here has been partly supported by the projects TIC-2000-1568-C03 (Spanish Ministry of Science and Technology), IST-2001-37724 (European Union) and CTIDIB/2002/336 (Valencia government: Generalitat Valenciana).

†Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I, Spain. canut@uji.es

‡Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I, Spain. albalade@uji.es

§Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I, Spain. gould@uji.es

1 Introduction

E-commerce has become so important over the past few years that it promises to become the global business paradigm of excellence in the near future. E-commerce services will naturally subsume the niche market of geographic information services. For this reason, the development of the concept of web services, its discovery and composition - composite services based on the chaining of other services both atomic and composite - have become as one of the principle characteristics for a mature and consolidated e-commerce [16].

The study of workflow has over the past few years provided techniques for modelling the flow of activities, normally known *a priori*, that are executed procedurally within a controlled, homogeneous intra-organizational environment, and where a single participant controls the flow. Within the web context, however, it is not always possible to encounter services which are homogeneous with respect to their composition. In the heterogeneous and dynamic context that is the web, diverse services may appear and disappear without warning. For electronic commerce to reach its expectations, it would seem unwise to attempt migration of traditional process modelling techniques to these new open environments [14]. It would seem more logical to adapt the process models to inherent positive characteristics of the web. For this reason, and within the realm of the European Union-funded project ACE-GIS (Adaptable and Composable E-Commerce and Geographic Information Services) [10] we propose a novel approximation defining a declarative composition model which is consistent with the open and dynamic characteristics of web architecture and behaviour. ACE-GIS proposes to build a developers platform for model-driven design and invocation of compound geographic information services (in addition to basic e-commerce services such as authentication and eventually payment). While judging the conformance of a novel web service to the implementation specifications within Open GIS Consortium is quite straightforward[1], the leap from one-to-one conformance to true interoperability among diverse web services has yet to be realised. In fact, measurable interoperability has yet to be defined satisfactorily, which is why the authors undertook the present study.

The remainder of the paper presents the incremental approach as a declarative model for web service composition. Section 3 describes in some detail the components which form the conceptual architecture, and the specific process of composition and invocation, fundamental building blocks in the conceptual architecture which exploits the incremental composition concept. Then we provide a simple example of a prototype implementation of composition and invocation in section 4. Finally, we provide conclusions and ideas for future research.

2 Incremental Approach to Web Service Composition

From our viewpoint, a generic web service can be considered either an atomic or composite service. An atomic service is an Internet-based software component that does not rely on other web services to fulfill user requests. An example of an atomic service is a weather service that provides weather-related information such as wind direction and magnitude. On the other hand, a composite service, a so-called opaque service chain in ISO/OGC terminology [13], is defined as a set of such general services (atomic or composite) working together to offer a value-added service. As part of a pilot application in the ACE-GIS project, a toxic gas dispersion model has been selected as an example of a composite service which is composed of several atomic services such as *GetNearestAirport*, *GetWind*, *GetGasDispersionPlume*, and *GetGasDispersionMap* providing key information for a realistic emergency plan in case gas toxic release from chemical plant. So, an important issue will be to offer not only atomic

[1] See www.opengis.org and follow the link to *Implementations Specifications*

services to be used ad hoc but also composite services based on the chaining of other services (atomic or also composite) [3]. In both case the resulting composition should appear to the user as a single entity with which to interact. In this sense, the incremental composition model described in [11] outlines a declarative model for web service composition, which is centered around descriptive aspects, on interoperability and on scalability. Basically, a web service composition is defined as a set of generic web services which interact according to certain logical rules. These logical rules specify the composition pattern (serial, parallel, etc.) which describe the execution order of the services involved in the composition, and the connection flow established for the data dependences between services. This affords several benefits including encapsulation of the underlying service complexity and straightforward reuse of a composition in future compositions.

Incremental composition differs from the architecture of traditional workflow management systems because the latter are essentially process oriented in the construction and execution of the composition graph. In existing approximations the developer normally must work with the entire graph, and so complexity increases noticeably as the composition grows. However, the approach described here holds the advantage that in the majority of process models, as complex as they may be, the logical composition graph may be decomposed into basic patterns, serial or parallel, of just two services at a time. The final composition encapsulates the complexity of the model in a manner similar to that of class hierarchy in object oriented languages; the main difference being that here we refer to abstract interfaces and not instantiations of classes.

The reader may be wondering why we have not exploited one of the several existing languages for the composition of web services, such as BPEL4WS, WSCI, BPML, etc. [2]. Within our perceived use case, chaining 2 or 3 well-known geographic web services - say a Web Map Server to a Web Feature Server - to known e-commerce services such as authentication, is not necessary to overload the system with notions of conditional iteration of web services or feedback among them. Therefore, in the context of the goals of the ACE-GIS project, we consider web service compositon without a procedural language or a subset of well-known compositional languages, that is, avoiding the need for a compositional language, only based on appropriate pattern definitions and the inherent structure of the service description. In addition, there currently exists no international consensus on standard languages for composition, most are merely commercially-driven proposals [1] and, there also exists no clear convergence among industry initiatives (BPML, BPEL4WS, etc.) and the research community efforts in semantic description of web services (DAML-S, OWL-S [8]). Similarly, within traditional flow context, we recognise efforts toward defining standards for workflow interoperability (Wf-XML or SWAP) between workflow management systems [12], but this also has not shown a great deal of consensus.

Composing web services requires the description of each service so that other services can interact with it. The language for describing operational features of web services, in most all composition languages, is WSDL [7]. In treating a composition as an atomic web service, however, this composition should also carry a WSDL describing the entire functionality of the composition (see Figure 1). In these composition descriptions only the abstract description is present, without details on concrete implementation: things such as access protocols or location points. Service compositions at the abstract level fit comfortably within the incremental composition concept, unlike concepts of tightly coupled components within distributed computing models. In addition to the abstract WSDL description a composition encapsulates the specification of composition pattern and connection flow established for the services within a composition. These are encoded declaratively in an XML WSDL extension. Additional aspects such as the quality of service (QoS) desired in a composition, semantic

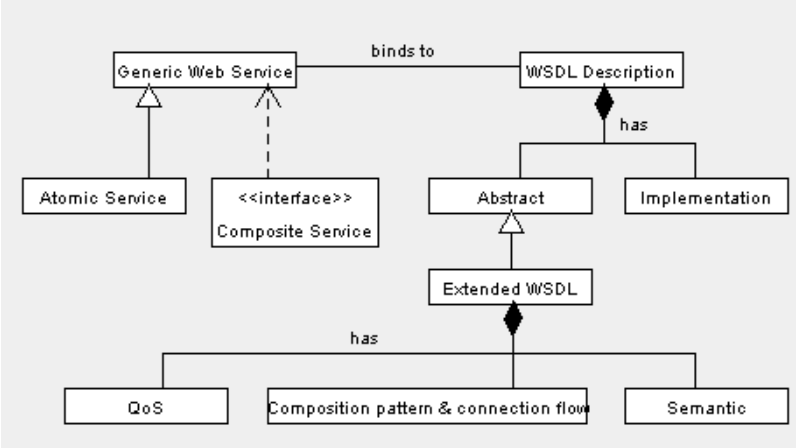

Figure 1: Relations in incremental composition (after [4]).

attributes or security considerations, may also be embedded declaratively in the description. This together, abstract WSDL plus XML extension, forms an abstract interoperability interface of the composite service, as a single conceptual entity, which is interpreted by the *Interpreter* and *Invocation Handlers* (see section 3) in the model presented. This extended composite WSDL description is fundamental for the incremental composition process, as the approximation follows an abstract component-based model, which facilitates interoperability and connectivity between services.

3 Conceptual Architecture

To be able to define the function views that are established in the proposed architecture for composition and invocation of web services, we describe first a global vision of this conceptual architecture (see figure 2).

Basically, the architecture supporting the incremental composition concept is composed of 5 layers providing a sort of middleware between the users and the external components such as other web services, registries and catalogs:

- **User layer**, formed by applications and interfaces directly accessible by the end user;
- **Composition layer**, responsible for generating the declarative description of the composition;
- **Additional components layer**, which contains other components of the composition, such as security, quality of service, semantics or system monitoring;
- **Invocation layer**, which interprets the composition and executes it;
- **Local storage layer**.

Following, we describe briefly the funcionality of the basic components of the architecture, excluding components from both local storage and external layers.

Coordinator This component (not included in Fig. 2), as its name indicates, coordinates the other components, guaranteeing the desired information flow among user requests, middleware and external components.

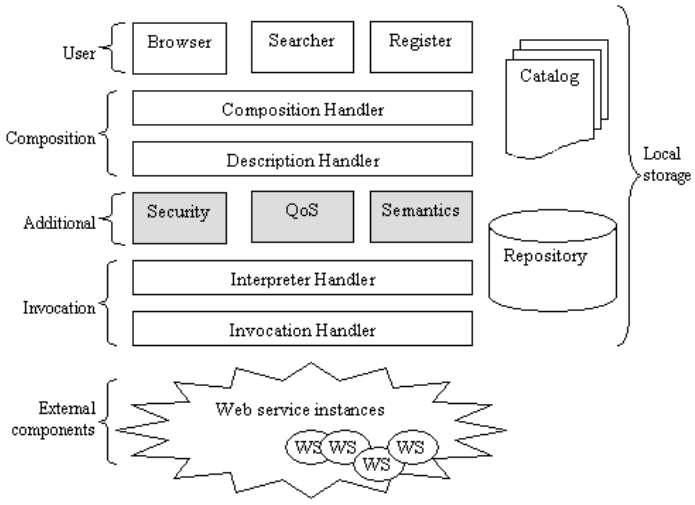

Figure 2: Proposed conceptual architecture for incremental composition (shaded components denote under development).

Browser This component interprets both simple (standard) and composite web services descriptions.

Searcher Provides the means to discover services both in external catalog registries (UDDI) and distributed local and external registries (WSIL).

Register Once a new composition is created, it can be published in a local registry, thus exposing the new composition for future use.

Composition Handler This component encodes the composition pattern and connection flow in XML fomat to combine a pair of services. This description is interpreted by *Description Handler* to create a valid extended WSDL description.

Description Handler This module is responsible for generating a complete description of the external behaviour of the composition, according to WSDL specification, from both WSDL description services involved in the composition and declarative XML format returned by *Composition Handler*. The resulting description is interpreted by *Invocation Handler* to execute the desired composition.

Security, QoS, Semantics These components are not strictly necessary to carry out a simple web services composition, however they may facilitate certain composition of value-added services. For example, if critical compositions are needed for public safety reasons, it would be necessary to incorporate aspects of authorization and user's authentication. Within the context of e-commerce, clients and providers define agreements from both ends, specifying quality of service characteristics that improve notably the success of the composition [6]. On the other hand, to allow dynamic services discovery and composition, it is necessary to semantically enrich the service descriptions.

Interpreter Handler This component is responsible for interpreting the composite service description for execution. It analyzes the composition description in search of the services which compose it.

Invocation Handler This component traverses the tree in backtracking-mode, by which each visited leaf node represents an atomic web service invocation. *Invocation Handler* uses an API in order to dynamically select the concrete mechanism to access the service interface, to encode messages and encapsulate them in the transport protocol according to WSDL description and, finally, to invoke the concrete web service instance.

From this conceptual architecture of middleware layers, we establish essentially two functional views of the system, to aid in composition and invocation of services:

- Composition view;
- Invocation view

The composition of services involves primarily components of the user and composition layers. At the user layer are found the components *Searcher*, *Browser* and *Registry*, while the components *Composition Handler* and *Description Handler* together form the composition layer of the conceptual architecture. Similar to the composition view, the invocation view contains components pertaining to the user layer, which are assigned to locate, select and navigate among the desired web services. Logically, this view also incorporates the invocation layer, which is necessary for the execution of web services compositions; this layer is comprised of the *Interpreter Handler* and *Invocation Handler* components.

The remainder of this section describes in detail the process of composition and invocation of web services, under the guidance of the incremental composition conceptual architecture.

3.1 Composition view

In figure 3 we illustrate a UML sequence diagram outlining the mode in which the search and composition of Web services is carried out.

The first 4 steps define the search for available web services, atomic or already composite, supposing the presence of a service catalog accessible locally or remotely (ultimately via Web). Once the services are selected, the composition process is initiated (steps 5-14). This composition is supervised by the user with assistance from the *Coordinator* component. During the composition process, first we establish a composition pattern and connection flow for the messages involved in linking a pair[2] of web services (step 5). As mentioned earlier, component-oriented composition consists of decomposing complex compositions in basic patterns of pairs (avoiding the need for a flow language). In this manner the desired composition is constructed in multiple iterations of composing two at a time: for example, simple to simple forming complex, then complex plus another simple, etc., in an *incremental*-mode. Even through current composition languages allow construction of several services simultaneously [2], however within geographic web service context it is not realistic that a user would combine, for example, 10 Web Map Services 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 each time. In the prototype application presented the user manually establishes the connections between the two services. One of the immediate goals is to eliminate this need for human intervention to reach increasingly pure levels of automated composition, perhaps initially monitored by the user. This can only be accomplished through the addition of semantic translators, ontology parsers and addition of semantic tags [15], in line with the grand proposal of the Semantic Web [5]. These extensions are already contemplated within the presented conceptual architecture. For example,

[2]There is no problem in defining patterns of higher cardinality.

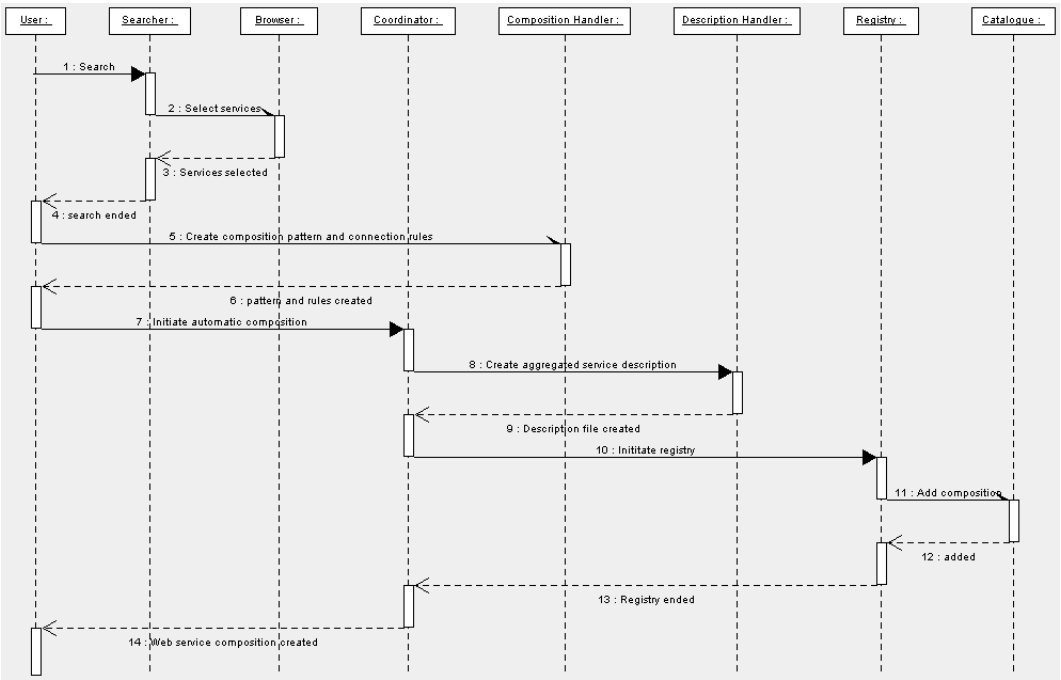

Figure 3: Sequence diagram for search and composition of services.

a semantic component describing web services and their data types will be in charge of enriching these descriptions from domain ontology (such as for example, emergency experts' vocabulary) in order to allow an inference engine to be able to perform "matchmaking", to dynamically detect valid pairs of web services.

The composition process starting at step 7 is automated and controlled by the *Coordinator* component. During step 8 the *Description Handler* component encodes the composition pattern and connection flow in a declarative format based on XML. Later the components from the invocation layer interpret this encoding at the moment of invocation of the compound service. This format describes two fundamental aspects of the composition: how to connect and in which order to combine the two services, and what should be the external behaviour of the completed composition. The first aspect defines the type of composition, normally serial, and the connection flow for representing the data mappings between the two services. The second aspect describes the abstract interface of the two services viewed as a single, independent web service. On this occasion the abstract part defined in the WSDL specification is used. Finally the resulting extended composite WSDL description of the composition is added to the service catalog, thus exposing the new composition for future use (steps 10-13).

The extended composite WSDL description is compatible with the WSDL standard. Any WSDL viewer should be capable of interpreting correctly the extended WSDL descriptions produced by our prototype, ignoring the new tags added by our model.

3.2 Invocation view

Figure 4 includes a UML sequence diagram which illustrates the mode of search and invocation of both atomic and composite web services.

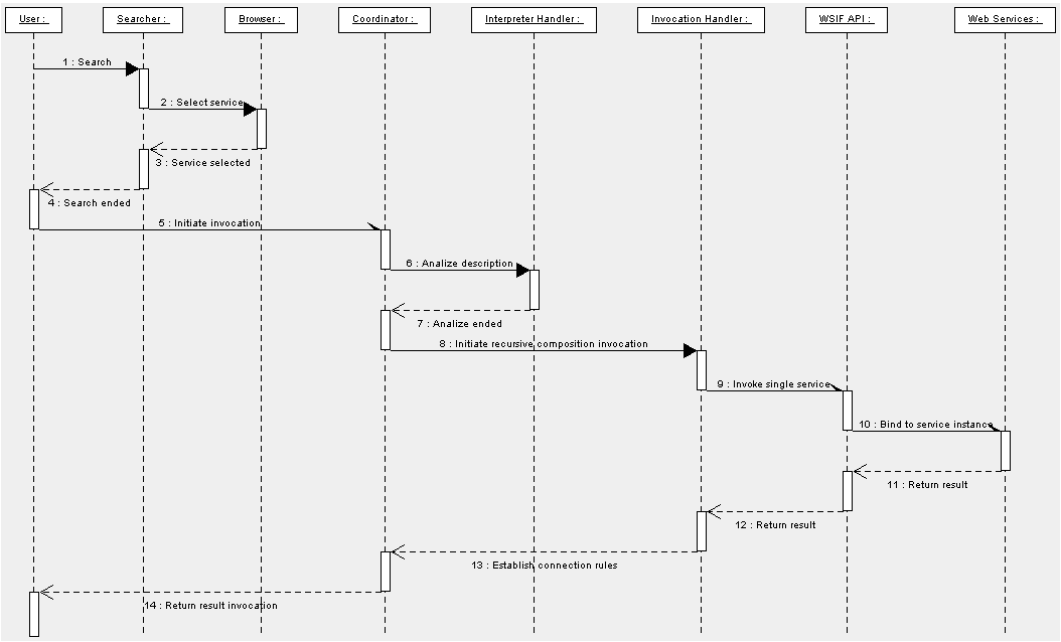

Figure 4: Sequence diagram for search and invocation of services.

In general terms the invocation of a composition requires an analysis of the associated description to encode the logical composition graph as a tree structure. Then, once this structure is created it is traversed in backtracking-mode, invoking the component web services and chaining the results obtained according to the connection flow defined in the extended composite WSDL description.

As in the composition view, first off the user selects the service or composition he or she wishes to invoke. The invocation process is totally automated and controlled by the *Coordinator* component. The user need only facilitate the initial parameters of the composition and await the composition's response (steps 5 and 14). The *Interpreter Handler* (step 6) realises two basic functions. The first, recursively interpret the extended WSDL of the composition. As all compositions are comprised of services, the *Interpreter Handle*r analyzes the composition description in search of the services which compose it. If it encounters atomic services then the search halts and no further analysis is needed. If, however, it encounters composite services, the *Interpreter Handler* continues analyzing the new composition until it reaches a pair of atomic services.

This analysis algorithm extracts the logical composition graph from the very structure as defined by the composition description. The second function of the *Interpreter Handler* is the creation of the tree structure as an implementation of the logical composition graph. In this interpretation process we can appreciate how the composition graph is implicitly defined by the composition structure. As a consequence, at no moment does the user need to construct the complete graph at design time in order to create the desired composition.

Once the tree is created, the *Invocation Handler* component takes control to initiate the invocation process (steps 8-13). Invocation is a backtracking process where the leaf nodes are directly invoked as atomic web services (step 10) via the Web Services Invocation Framework (WSIF) [17], and open-source API provided by the Apache Software Foundation (www.apache.org) which permits the invocation of web services in manner which is indepen-

dent of protocol or access method (SOAP, HTTP GET/POST, etc.). Internal nodes on the tree are not directly executable because they describe virtual compositions. Their mission is to interpret the composition pattern and connection flow of the children nodes, that is, decide whether to execute them in series, parallel etc. and how to establish the data flow between these children nodes.

The current implementation of our model executes the service composition via the invocation of each instance and connects the information between services as a function of the data flow established by the user. This vision of static composition has availability problems if one of the services constituting the flow is not operational or available at the moment of execution. Future investigations are aimed at dynamic services composition at execution time, whereby the composition itself is able to detect functional anomalies and can substitute defective services with alternates in a transparent manner.

Aside from the mentioned limitations, backtracking-mode 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 of the composition.

4 An Example: Arithmetic Functions Composition

In this section we present a web based prototype application which has been developed for the rapid integration, composition and invocation of web services, in an incremental fashion. Utilization of this prototype permits us to test and demonstrate a first approximation of the component based composition described here, its pros and cons, as well as the backtracking process in the composition invocation.

This simple example is illustrative in order to demonstrate the capabilities of the prototype application. Future work, already under way, is to apply the prototype to geographic web services, beginning with well-known services such as those defined by Open GIS Consortium, such as a composition of a Web Map Service and a Web Feature Service. While these tests will be far more realistic and useful to an end user, they pose a degree of complexity which does not help us to explain the basic concepts in a didactic manner here.

Our simple example is based on the four basic arithmetic operations (addition subtraction, multiplication, and division) each of which is exposed as a separate web service. Each web service implements an operation through a single public method of the type:

$$\begin{aligned} \textit{Add(op1, op2)} &: \textit{AddReturn} \\ \textit{Subtract(op1, op2)} &: \textit{SubtractReturn} \\ \textit{Multiply(op1, op2)} &: \textit{MultiplyReturn} \\ \textit{Divide(op1, op2)} &: \textit{DivideReturn} \end{aligned}$$

Imagine that we wish to solve compositions of arithmetic functions. For example if f y g are combinations of basic arithmetic functions, then $f \otimes g$ would be a valid composition of functions. The function f might be defined as the combination of a sum and a division, whereas the function g might be another combination of the operations subtraction and multiplication:

$$\begin{aligned} f(x, y, z) &= \frac{x+y}{z} \\ g(u, v, w) &= (u - v) * w \end{aligned}$$

In this manner the composition $f \otimes g$ would take the following form:

$$f \otimes g = f \otimes g(x, y, z, v, w) = g(f(x, y, z), v, w) = ((\frac{x+y}{z}) - v) * w$$

If we analyze the resulting composition, it would be formed by a combination of simple arithmetic services. Evidently, it would be totally inefficient to attempt to create a separate and permanent web service for each possible arithmetic composition. In this case no service exists to resolve $f \otimes g$, however exploiting the combination of existing atomic services it becomes possible to meet the special needs of the desired combination.

The remainder of this section describes in some detail how we realize the composition and invocation of arithmetic functions using the prototype which implements the incremental composition model.

4.1 Composition of arithmetic functions

The construction of compositions in the proposed model follows an incremental, top-down design, that is to say it is initiated with the union of atomic services to form intermediate compositions, until arriving at the compound compositions desired by the user. In order to create the composition $f \otimes g$, we previously would need to have created the intermediate compositions f y g. Consequentially, there are three new compositions awaiting in a registry, to be reutilized at another moment or by other users: f, g and $f \otimes g$.

The component *Composition Handler* is charged with specifying the composition pattern and connection flow. Specifically, the output parameter *AddReturn* of the *Add* operation with the input parameter *op1* of the *Divide* operation. The other input parameter of *Divide*, *op2*, is defined as a parameter external to the composition, that is, introduced by the user at the moment of invocation. Additionally, in this prototype it is possible to define input parameters as constants, whose value remains embedded in the extended WSDL description of the composition. Once the parameters defining the composition are configured, the *Description Handler* component creates the associated description and adds it to the registry of available services.

In the same way, the user creates a composition g and then the final composition $f \otimes g$, from the intermediate compositions f and g. This iterative method of gradual composition encapsulates the complexity of the composition at design time. It is far easier for the user to compose $f \otimes g$ from f and g than from the basic arithmetic operations themselves. Also, the incremental composition model alongside the possibility to independently invoke each intermediate composition created, facilitates debugging a priori. Utilizing this prototype it is possible to create each intermediate composition, execute it, and validate it correct operation. This characteristic helps save effort in later debugging steps in the final composition.

4.2 Invocation of arithmetic functions

The invocation process is centred on tree structure for the logical composition graph. The general invocation process of any composition is comprised of the following steps. The user introduces the initial composition parameters. Then the components of the invocation layer are charged with creating the tree based on the description of the composition, with invoking the composition while establishing the data flow between connected services. Finally the component *Coordinator* returns the resulting data to the user.

Returning to the example of invocation of the composition $f \otimes g$, the user introduces the initial parameters of the composition. Next, the *Interpreter Handler* is responsible for creating the tree structure. During the process of analysis the *Interpreter Handler* creates a node for each composition or atomic web service encountered, establishing double links between a father node and its children nodes. The initial invoked composition is identified

by the root node. Each of the atomic services or intermediate compositions forming a given composition is converted into a child node of the node identifying the main composition. Specifically, the composition $f \otimes g$ is converted in root node and the compositions f y g as children nodes on the left and right respectively. Continuing the analysis recursively, the node associated with the composition f contains as children the atomic web services *Add* and *Divide*, while the node associated with the composition g contains the services *Subtract* and *Multiply*.

Once the tree is constructed, we need only to invoke it to obtain the composition's result. It is not possible in all cases to establish a priori the concrete value of the data flow between the services comprising the composition. Normally dependencies at execution time will impede the specification of concrete values for all parameters of web services. It is necessary to utilize a backtracking mode algorithm to traverse the tree, in order to realize simultaneously the downward search and adjustment of inter-service data flow while returning. For example in invoking composition f, the services *Add* and *Divide* are defined in sequence. In this case it is not possible to begin execution of the service *Divide* without first obtaining the results of *Add*. In fact, the result of invoking the composition $f \otimes g$ shows the hierarchical structure of the tree which represents the logical composition graph.

5 Conclusions

In this paper has been presented a model for incremental composition of web services within the context of the ACE-GIS project [10], on-going research on geographic web service interoperability. This approach has allowed to test and establish the basis for the interoperability problem among services.

The model proposes a definition of web service compositions which are loosely coupled based only on abstract syntactic descriptions in WSDL. As a major difference over other known compositional languages, this approach does not define a small set of well-known languages. We propose a different perspective for composition 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. This incremental composition is specified declaratively in XML WSDL extension, as an aggregation of external descriptions of the services as well as the composition pattern and connection flow that define the logical rules among the contained services. So, from the inherent structure of this incremental composition the *Interpreter Handler* generates a tree structure which the *Invocation Handler* then traverses in order to invoke the composite service. Future work will include the dissemination of the incremental model components under open source license and the presentation of the WSDL extensions 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, such as the mathematical function composition scenario. Now, we are working to extend our incremental composition model to include any XML-Schema's user-defined data type interchanged between services, such as emergency response scenario. That would generalize our model to compose any available web service, consequently permitting the composition of web services based on principles of interoperability and scalability.

Additionally we are working with ACE-GIS partners (University of Münster) to incorporate semantic aspects (extensions of DAML-S, OWL-S) of the emergency management situation, with the goal of improving semantic interoperability within the composition. Finally, to help assure that the resulting service composition is somewhat fault tolerant with

regard to service availability and quality of service, we are also investigating dynamic composition at execution time [9], so that the composition will be able to detect service faults and replace faulty services with substitutes meeting user requirements.

References

[1] W.M.P. Van Der Aalst. Don't Go with the Flow: Web Services Compositions Standards Exposed. *IEEE Intelligent Systems*. 18(1): 72–76, 2003.

[2] S. Aissi, P. Malu, and K. Srinivasan. E-Business Process Modeling: The Next Big Step. *IEEE Computer*. 35(5): 55–62, 2002.

[3] N. Alameh. Chaining Geographic Information Web Services. *IEEE Internet Computing*. 7(5): 22–29, 2003.

[4] B. Benatallah *et al.*. Towards Patterns of Web Services Composition. In S. Gorlatch and F. Rabhi (Eds): *Patterns and skeletons for parallel and distributed computing*, UK Springer Verlag, 2002.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*. 284(5): 34–43, 2001.

[6] J. Cardoso. Quality of Service and Semantic Workflow Composition. PhD Thesis, University of Georgia, Athens, GA, 2002. http://lsdi.cs.uga.edu/j̃cardoso/Silva-Cardoso_Antonio_J_200208_PhD.pdf.

[7] R. Chinnici *et al.*. Web Services Description Language (WSDL) Version 1.2: Core Language. *W3C Working Draft*. 11 June 2003. http://www.w3.org/TR/wsdl12/.

[8] The DAML Services Coalition. DAML-S: Semantic Markup for Web Services, 2003, http://www.daml.org/services/daml-s/0.9/daml-s.pdf.

[9] S. Ghandeharizadeh et al.. Proteus: A System for Dynamically Composing and Intelligently Executing Web Services. In *Proceedings of the First International Conference on Web Services (ICWS 2003)*, Las Vegas, Nevada, June 2003.

[10] M. Gould *et al.*. The ACE-GIS Architecture for Service Chaining: Contribution to INSPIRE. In *Proceedings of 9th Workshop EC-GIS*. A Corua, Spain, 2003. http://www.acegis.net/.

[11] C. Granell, J. Poveda, and M. Gould. Incremental weak composition and invocation of geographic web services. In S. Levachkine, J. Serra, and M. Egenhofer (Eds.): *Proceedings of the 2nd. International Workshop on Semantic Processing of Spatial Data (GEOPRO 2003)*. Mexico City, Mexico, 2003, 179-187.

[12] J.G. Hayes *et al.*. Workflow Interoperability Standards for the Internet. *IEEE Internet Computing*. 4(3): 37–45, 2000.

[13] ISO/TC211, Geographic information - Services (ISO/DIS 19119) v4.3. International Organization for Standardization. Also Open GIS Consortium Abstract Specification 13. 2002.

[14] M.P. Singh. Physics of Service Composition. *IEEE Internet Computing*. 5(3): 6–7, 2001.

[15] E. Sirin, J. Hendler, and B. Parsia. Semi-automatic Composition of Web Services using Semantic Descriptions. In *Proceedings of Web Services: Modeling, Architecture and Infrastructure Workshop at ICEIS 2003*. Angers, France, 2003.

[16] S. Tsur *et al.*. Are Web services the next revolution in e-commerce? (Panel). In *Proceedings of the VLDB Conference*. Rome, 614-617, 2001.

[17] Web Services Invocation Framework, Apache Foundation, 2003. http://ws.apache.org/wsif/.