EnviroInfo 2002 (Wien)
Environmental Communication in the Information Society - Proceedings of the 16th Conference
 ISBN: 3-9500036-7-3

First Steps in the Development of a Web Service Framework for Heterogeneous Environmental Information Systems

Uwe Radetzki[1], Sascha Alda[1], Thomas Bode[1] and Armin B. Cremers[1]

Abstract

In this paper we present our first steps in developing a web service framework for heterogeneous environmental information systems. This framework allows an easy assembly of data and method services which are distributed over the entire web. In the field of web services, two consortia already conduct endeavors to standardize access to data and method of a web service. Besides, several standardization efforts for building distributed systems arise from the field of distributed objects and component technology. Our approach combines and integrates these technologies within one framework and, thus, provides a higher interoperability among all these different standards. The aggregation as well as the interaction among these services is prescribed by a XML-based workflow process engine which is also part of the framework. Another integral part of this framework is the XMT model which allows the integration of incompatible services by means of portable algorithms. We further demonstrate a conceivable application of our framework in a scenario pertaining to disaster management, in particular to the field of fire fighting.

1. Introduction

In this age where a rapidly increasing volume of spatial and environmental information is collected, the problem is in the majority of cases not to access these data separately but rather to correlate and integrate them in a more meaningful manner. This is because the relevant information for users making serious decisions is stored in many distributed and different systems using heterogeneous data types and formats. Further, this information should be available on several and even mobile devices.

This will be important in the field of disaster management for example in the case of fire fighting. Every year, forest fires bring destruction, injury, and even death. The ability of incident managers in the headquarters to control fires quickly, thereby limiting the damage, is to a great extent dependent on their ability to gather and combine

[1] Institute of Computer Science III, University of Bonn, Roemerstr. 164, D-53117 Bonn, Germany, email: {ur | alda | tb | abc}@cs.uni-bonn.de, http://www.cs.uni-bonn.de/III

various information, to deploy resources, and to perform emergency evacuations efficiently. Further, proper information has to be available for action forces on site, supporting them in decision-making, too.

A group of firemen fighting against a fire source in an unknown woodland has to trust in several geographical information as well as in instructions of their headquarter leaders. This information must provide an adequate view of the landscape they are resided. Such view should contain courses of rivers, fire lines, firewalls, locations of buildings, etc. Furthermore, the safety and well being of all fire fighting resources on the fire line, as well as their ability to effectively battle the fire, depend on up-to-date information about current fire sources, contact with immediate supervisors, knowledge of their actual location on the fire line, and prediction of fire behavior which is the result of combining fire source information with current weather and wind information. Vice versa, firemen may have information about a changed situation on site, e.g. altered fire lines or even new fire sources. Thus, they should be able to supply the headquarters with the new data. It is quite evident that action forces on site will use other hard- and software platforms than the incident managers in the headquarters.

Taking the instructive example above we propose a dynamically adaptable web service infrastructure as an adequate software support for such scenarios. Such a framework should allow a flexible and problem-oriented integration of different services and applications supporting interoperability among them. Further, different user profiles should have an adequate access on these resources.

The rest of this paper is organized as follows: section 2 presents identified requirements for handling the heterogeneity in environmental information systems. In the subsequent sections our approach is outlined and evaluated in a fire fighting scenario. The last section concludes the current results of our work and suggests future directions.

2. Analysis of Requirements

In the previous section we outline that a software solution in disaster management for instance has to address two major targets: heterogeneity and flexibility. In the majority of cases the problem is not to access several distributed data sources and method services separately but rather to combine them in a meaningful and flexible fashion. This problem arises as a result of the different forms of heterogeneity: The technical heterogeneity focuses on the problem that data and method services are working on different hard- and software platforms. For example, a couple of them may be accessible through CORBA objects whereas others are realized as internet-capable services which are accessible through HTTP Get / Post. The same fits on the clients, too, because different user groups have different application profiles using various forms of hard- and software. On a syntactical level the data is stored in in-

compatible formats and may be different in terms of data modeling concepts, data encoding techniques and access functionalities. The data access for instance may be different in terms of various query languages, like SQL, OQL or proprietary access methods. Furthermore, by means of semantic heterogeneity the different data and method formats place high demands on their users and application programmers since they have to know the exact semantics of these data and methods. A more detailed description can be found in (Bergmann et al. 2000).

The rapid pace of change in an emerging environment causes the demand for flexible and adaptable software solutions which are not fulfilled by monolithic and closed systems. In the field of disaster management for instance new emergency plans may be developed which result in changed or even new requirements for interactions of different services and people. Thus, the first aspect of system flexibility is the ability to vary these interactions dynamically and in an easy way. Further, such maintenance should be performed from different locations, e.g. by users working with mobile devices. In such environments it is necessary to extend the system by combining the distributed resources with local ones in a previously unpredictable manner. So another aspect of system flexibility will be the integration and assimilation of new local pieces of software from different operational environments that may also be written in different programming languages with distributed data and method services. Due to the nature of research the whole realm of spatial data and analysis methods will never be known. Thus a third aspect of system flexibility will be the possibility to integrate new a priory unknown data types seamlessly.

2.1 Approaches

In principle, several approaches dealing with the problem of heterogeneity focus on pure data integration and do not consider the application of several methods. Here, standard techniques are data warehouses and database federations which are both require an explicit schema integration.

Another way to overcome the barrier of heterogeneity is to standardize the interfaces for data and method access. In the geo information working area the Open GIS Consortium (OGC) specifies geo-related services in order to allow a range of different networks, applications, and platforms to profit from these methods and information. The OGC defines the semantics as well as the syntax of services using a common geo-related vocabulary. In this way the homogeneity of these services should solve the problem of interoperability in the geo information working area. But during standardization highly sophisticated and specialized services as well as non-geo-related services needed in fire fighting scenarios for instance, cannot be taken into account. The World Wide Web Consortium (W3C), however, specifies a technical basis for web services by defining XML-based languages such as SOAP (W3C 2000/2001) or WSDL (W3C 2000/2001) allowing to access services in a uniform

way. In the fields of distributed computing and software component technology the Object Management Group (OMG) defines another technical standard platform called CORBA (OMG 1999/2001) for building distributed systems. However, the later ones address only the technical heterogeneity but less the syntactic or semantic aspects.

Consequently, we propose a method which combines several standardization efforts with a flexible and extensible web service framework, supporting a higher interoperability between different resources. This framework should be easily maintained and adapted to modified requirements. In order to combine such a distributed framework with local applications we will further suggest a client-side integration platform. The later aspect is not in the main focus of this paper and will not be outlined here. A detailed description on this integration architecture can be found in (Bode et al. 2002, Radetzki et al. 2002).

3. IRIS – A Web Service Framework

In this section we explain the architecture of a web service framework we are currently developing. IRIS, **I**nteroperability and **R**eusability of **I**nternet **S**ervices, is a web service framework which allows users to create own services by combining them out of existing ones. Thereby, the assembly of services occurs through process-controlled workflows.

In general, the IRIS web service framework is conceived to adapt and reuse distributed services for different purposes and application scenarios. On an abstract level the architecture encompass four significant concepts (see fig. 1).

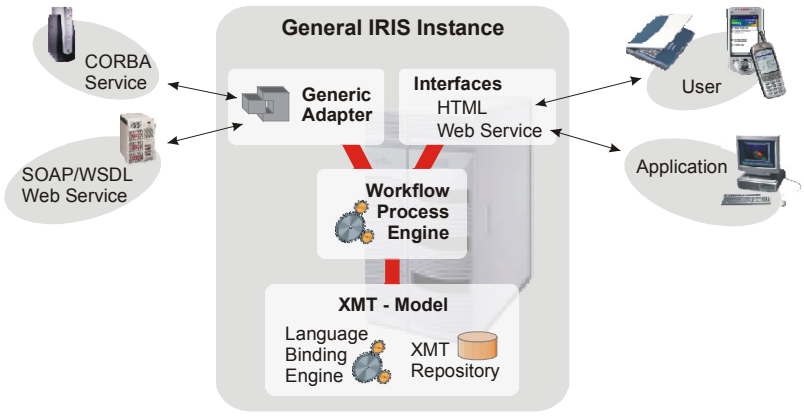

Fig. 1: The IRIS web service framework.

The *Generic Adapter* allows a technical encapsulation of different middleware technologies like CORBA or Web Services. In this way different resources can be easily integrated and handled uniformly. In order to take benefit from standardized geo-related services and technical standards, we are building service interface definitions based on WSDL for OGC-conform web services. Today, no standardized technical definition for the OGC Web Services Suite are available, but we guess this situation will change in the future because several efforts are undertaken to combine W3C web service specifications with the OGC ones. The Generic Adapters which are invisible for the users execute operations on and send events to services they are encapsulating with respect to defined workflows.

The *Service Composition Model* within the IRIS architecture is realized through process-controlled workflows. These workflows are defined for different purposes and executed by the *Workflow Process Engine*. Because the set of workflows is not fix, the architecture is extensible with new services and adaptable for changed requirements. Every workflow consists of a *control flow* and a *data flow*. The control flow describes how the integrated services should interact within a specific scenario. It specifies, for instance, the processing order, parallel executions and constraints. With respect to a defined control flow an adequate data flow has to be specified. The data flow describes the way how the retrieved data of one service has to be managed or transformed to be useful for other services on the control flow. These definitions fall back on defined *Extensible Mediator Tags* (XMT) in order to transform incompatible data formats efficiently. We should remark that the workflow description is purely declarative.

The *XMT-Model* demonstrates a portable mediator principle which provides a way to combine heterogeneous and incompatible interfaces by means of data transformation steps and algorithms not available as pure web services themselves. This module interacts with a *Language Binding Engine* and a *XMT Repository*. The Language Binding Engine executes functions which are represented within the XMT model by *mediator tags*. Every mediator tag specification consists of an XMT declaration in an XML dialect and an XMT definition implemented by a portable code fragment. Such implementation may be written in Java or XSLT but the Language Binding Engine is extensible with further portable languages. Within these tags we distinguish tags with less semantic relevance, like regular expression searches or inverting of strings, and semantic tags, like prediction of fire directions by wind measurements and fire sources. All of them should be indexed in the XMT Repository which acts as taxonomy of a specific field. In this way the semantic and syntactic heterogeneity should be handled.

Every IRIS instance which is build upon the IRIS framework is managed by well-defined *Interfaces*. These interfaces support the customization and publishing of defined service aggregations. Further, they support interactive access, e.g. for Browsers by HTML, as well as SOAP/WSDL access for the use within other applications.

4. A Sample Fire Fighting Scenario

In this section we briefly describe the usage of the IRIS framework by means of the motivational example in section 1. The example acts as demonstration and may be different in reality. The aggregation of services occurs through a declarative XML workflow description. This XML-based language is a variant of the Web Services Flow Language (WSFL) (Leymann 2001) representing workflows in form of activity diagrams (OMG 1999/2001). Substantially the WSFL model is extended by our XMT model.

Let's imagine a weather information service (*WeatherService.com*) supporting clients with wind, temperature and other weather-related information and a geo-related database service (*GeoObjectService.com*) where users can store geo-related objects, like fire source locations. We want to build a fire prediction service (*FirePediction-Service*) for firemen on site providing them with images of the calculated fire direction. Fig. 2 illustrates an extraction of this aggregation:

```
 1 <workflow name="FirePredictionService">
 2   <serviceProvider name="FireLocationProvider">
 3     <locator service="GeoObjectService.com"/>
 4   </serviceProvider>
 5   <activity name="fireLocation">
 6     <performedBy serviceProvider="FireLocationProvider"/>
 7   </activity>
 8   <!-- line 8-13: the same for WeatherService.com -->
33 </workflow>
```

Fig. 2: Example of the service composition model (extract, line 14-32 see below)

At the beginning of this description the aggregated service is labeled using the *name* attribute of the *workflow* tag (line 1). In the following steps, the needed services are included (line 2-4) and their functions are represented as activities (line 5-7, only the geo-related database service is illustrated here). Then the control flow between these activities are defined (line 14-19, fig. 3). The control flow determines the order in which the activities are executed. In this example the operation requests from the workflow source to the both services are executed in parallel and after finishing the results are combined into the workflow sink. The *flowSource* and *flowSink* are predefined activities which define the start and end point of the process-controlled aggregation. Within these activities the new service operation (*predictReq*) is defined. The data flow between the different activities is defined through a list of data link elements (line 20-32, fig. 3). A data link element defines a message exchange between a source activity and a target activity. It should be mentioned that web service operations have a message for the request (*toMessage*) as well as for the response (*fromMessage*) of it; parameters are described by message parts. Line 21-23, for instance, indicates that the result (*windValue*) of the message *windInfoReq* from the *windAc-*

tivity activity is passed to the flow sink. Finally, the *predict* mediator tag is tied up with the flow sink calculating the predicted behavior of the fire direction (line 27-31). Here, the tag *xmt:predict* represents the executable function. The algorithm (not illustrated here) may be defined in Java in order to be portable in code and efficient in calculation. Every service description is validated before it is accepted by the IRIS framework.

```
14   <controlFlow>
15     <controlLink from="flowSource" to="fireLocation"/>
16     <controlLink from="flowSource" to="windActivity"/>
17     <controlLink from="fireLocation" to="flowSink"/>
18     <controlLink from="windActivity" to="flowSink"/>
19   </controlFlow>
20   <dataFlow>
21     <dataLink from="windActivity" to="flowSink">
22       <map fromMessage="windInfoReq" toMessage="predictReq"
           fromPart="windValue" toPart="windValue"/>
23     </dataLink>
24     <dataLink from="fireLocation" to="flowSink">
25       <map fromMessage="fireInfoReq" toMessage="predictReq"
           fromPart="fireValue" toPart="fireValue"/>
26     </dataLink>
27     <xmt activity="flowSink" message="predictReq">
28       <xsl:variable name="result">
29         <xmt:predict wind="$windValue" fire="$fireValue"/>
30       </xsl:variable>
31     </xmt>
32   </dataFlow>
```

Fig 3: Definition of the control flow and data flow

Fig. 4 illustrates the resulting process-controlled service composition. If the new service is triggered by users or other applications the workflow process engine will be activated executing the service description. Thereby, inherent mediator tags will be executed by the XMT language binding engine. The workflow descriptions can be accessed by the IRIS interfaces and through web browsers suitable maintained.

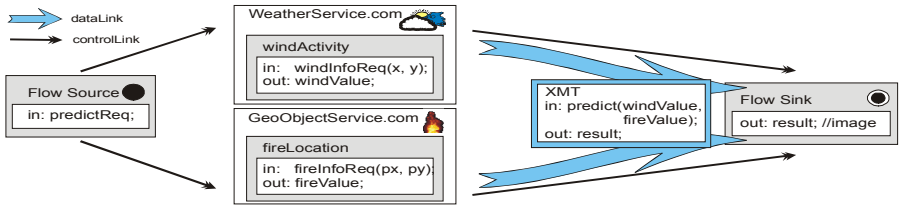

Fig. 4: Workflow process diagram

Further, every defined service composition can be published and used within other compositions by the web service interface. In this way, a hierarchically aggregation model is achieved.

5. Conclusion

Web services are an evolutionary step in designing distributed applications (Curbera et al. 2001). An essential feature is the composition of these services to new one in a flexible and adaptable way. Our solution to this problem is a web service framework, called IRIS, that supports process-controlled aggregated services in disaster management scenarios like fire fighting. We address the different forms of heterogeneity by a generic adapter and a portable mediator principle. Currently several parts of the IRIS architecture are already implemented. We further investigate methods for prediction of possible mediator tag combinations in order to combine incompatible services. Although we have drawn our motivation from disaster management, our framework is also useful in other application domains.

References

Bergmann, A., Breunig, M., Cremers, A. B. and Shumilov, S. (2000): A Component Based, Extensible Software Platform Supporting Interoperability of GIS Applications. In A. B. Cremers, K. Greve (eds.): Environmental Information for Planning, Politics and the Public, Proceedings of the 12th International Symposium "Computer Science for Environmental Protection" of Gesellschaft für Informatik (GI), pp. 90-102, October 4-6, 2000, Bonn, Germany.

Bode, T., Cremers, A. B., Radetzki, U. and Shumilov, S. (2002): COBIDS: A component-based framework for the integration of geo-applications in a distributed spatial data infrastructure. In: Annual Conference of the International Association for Mathematical Geology (IAMG), September 15-20, 2002, Berlin, Germany.

Curbera, F., Nagy, A. and Weerawarana, S. (2001): Web Services: Why and How. Workshop on Object-Oriented Web Services - OOPSLA, Oct. 15, 2001, Tampa, Florida, USA.

Leymann, F. (2001): Web Services Flow Language (WSFL 1.0), IBM Corporation, May, 2001.

OMG – Object Management Group (1999/2001): CORBA 3.0. OMG TC Document ptc/99-10-04, October 29, 1999, Unified Modeling Language (UML) Specification, version 1.4, September, 2001.

Radetzki, U., Witterstein, G. and Cremers, A. B. (2002): An Integration Platform for Heterogeneous Services in Life Science Applications. Workshop on Bioinformatics - 8th International Symposium on Methodologies for Intelligent Systems (ISMIS), June 26, 2002, Lyon, France.

W3C – World Wide Web Consortium (2000/2001): Simple Object Access Protocol (SOAP) 1.1, Web Services Description Language (WSDL) 1.1.