EnviroInfo 2004 (Geneva)
Sh@ring – EnviroInfo 2004
 ISBN: 28 29 30 275-3

COBIDS: A Component-Based Framework for Sharing Standardized and Non-Standardized Geo-Services

T. Bode, A. B. Cremers, U. Radetzki, S. Shumilov[1]

Abstract

Flexible and user-oriented integrated access to data and method services in very heterogeneous and dynamic information environments is an important precondition in order to give adequate support to innovative applications in many domains. In order to simplify the integration of services, a new approach based on component technologies has been developed. The developed infrastructure allows new, a priori unknown components to register themselves in a plug-in fashion. We have implemented a prototype system and demonstrated its use in an experimental geoscientific environment, called OPALIS. In this paper we present the evolution of the OPALIS software to the client platform COBIDS. The latter supports integrated access to distributed standardized and non-standardized services as well as connecting domain-specific local tools needed for processing the shared data.

1. Introduction

Monitoring spatial and environmental data, for instance acquired by satellite imaging or remote sensing, is essential for managing natural resources; from urban planning to predicting earthquakes, epidemic plagues and diseases (Andrianasolo 1999). Due to the fact that these data is acquired and managed by different groups, often with project-specific structures and formats and specialized software for analysis, we need a way of efficiently sharing both the data and the developed methods for data processing in order to integrate all the relevant information and tools in a meaningful manner. With the advent of service-oriented architectures (SOA), especially the spatial-data-infrastructures (SDI), a technical basis for building such *'Networks-of-Sharing'* has become available. To realize the outlined vision, we need tools for an easy service-based publication of data and methods as well as an appropriate platform for the integrated access of already deployed services.

[1] Dept. of Computer Science III, University of Bonn, Roemerstr. 164, D-53117 Bonn,
email: {tb, abc, ur, shumilov}@iai.uni-bonn.de

In order to simplify the publication of services, various domain-specific standardization efforts have been made, e.g. by the OpenGIS Consortium (OGC) or the International Organization for Standardization (ISO). But these efforts focus mainly on syntactic interoperability and lack solutions for semantic interoperability as well as connector facilities for project-specific data and specialized analysis software (Bernard 2003). Moreover, such standardization processes are very lengthy and time-consuming. Often new application domains, data types or analysis algorithms can not be included short-term, which is especially a problem for research groups and the sharing of non-standard data and methods. Basically, we are convinced that for a significant amount of needed services no standardization on the semantic level will be available in time. In contrast to the domain-specific standardization of services several tools are available supporting users in deploying domain-independent, W3C-conform services, called Web Services (Curbera 2002).

In typical application scenarios the accessibility of a single service is not sufficient. In disaster management for instance, the combination of services providing weather data, satellite and geographic information is essential. Since the semantic-aware integration of required heterogeneous services cannot be achieved directly, a software platform is needed where this integration can be realized in an application-specific context which is restricted according to the precise needs of the user.

In this paper we present the evolution of the OPALIS software to the client platform COBIDS (***Co**mponent-**B**ased **I**ntegration of **D**ata and **S**ervices*). The latter supports integrated access to distributed standardized and non-standardized services as well as connecting domain-specific local tools needed for processing the shared data. The focus is on the modifications of the client platform necessary for integrating standardized and non-standardized geo-services and tools.

2. From OPALIS to COBIDS

In (Bergmann 2000) we have presented an extensible software platform dealing with uniform access to and integrated processing of heterogeneous and distributed sources. The proposed platform is based on the OPALIS (***O**pen **Pa**leoecological **I**nformation **S**ystem*) client software, JavaBeans and CORBA component technologies. Further, it uses an XML-based protocol in order to transfer and interact with data. The framework has a three-tier architecture encompassing a component-based client platform, a built-in server for accessing and caching distributed data and meta-information, and various project-specific data sources. CORBA middleware is responsible for accessing the server and distributed data sources.

For transforming data depending on user requirements a mediator-based approach is used (Wiederhold 1999). Mediator components are responsible for transforming data depending on user requirements. An internal mediator framework of OPALIS incorporates a rudimentary registry, a generic format for data representation,

mediator components and a retrieval unit for these components. The data within the generic format is defined as needed either by instantiating an object or using an XML-based representation. We call these representations together as gXML. In order to handle the data efficiently, a central storage component maintaining the gXML data is implemented.

Since a proprietary naming and catalog server is built directly into the system, the OPALIS platform lacks flexibility to cope with the emerging numerous standards in SDI. In order to cope with standardized services we are redesigned the functionality of the OPALIS client platform in COBIDS (***Co**mponent-**B**ased **I**ntegration of **D**ata and **S**ervices*). The COBIDS framework provides enhanced client-side tools for semantic interoperation. Figure 1 provides an overview of the most important building blocks of the framework. The key component – as mentioned before – is a component-based and adaptable client platform which acts as a client-side broker enabling the interoperation between heterogeneous distributed services and local tools.

Within the client platform every remote service is represented by a specific Interaction Component (IC). Mediator components inside the client platform allow transferring information between particular ICs. The communication protocol used between the IC and a remote service depends only on the service itself, e.g. CORBA/IIOP or SOAP/HTTP. Integration of OGC-conform services and local applications is discussed in the following sections 3.2 and 3.3, respectively. The connectivity of local applications is also realized by means of ICs and can be done using one of four different strategies depending on the kind of the application.

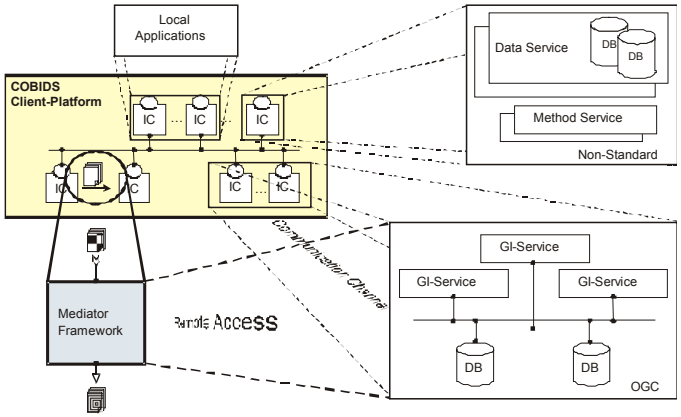

Figure 5:Architecture of the COBIDS platform

In order to cope with different data structures and semantics on demand, the OPALIS mediator framework has been replaced by a generic dynamically extensible mediator framework of the COBIDS (see section 3.1). OPALIS server modules implementing the framework which were built directly into the OPALIS system have

been isolated from the client platform and realized as autonomous services. Some of the framework functionalities now are best realized by a combination of existing external services (Radetzki 2002).

3. COBIDS Highlights

3.1 Mediation facilities

As mentioned before, the OPALIS mediator framework consists of mediator components, the gXML representation and a rudimentary registry and retrieval unit. While the mediator components translate data between data-producing ICs and data-consuming ICs, they process only relevant parts of the data being sent. Several mediator components can be combined dynamically by the retrieval unit in order to realize a desired transformation between different ICs. The syntactic discrepancies of different data types are homogenized by gXML. Every data format can easily be mapped to this representation and therefore can be accessed in a uniform way by the ICs within the client platform. The advantage of being data-type-independent has the trade-off of being slightly less efficient, because the object representation needs to be interpreted. Especially the discovery facility of the retrieval unit is enhanced within the COBIDS platform.

The discovery and the composition of mediators are realized by a string comparison of object interface names (represented as URI, thus they are supposed to be unique). For instance, some IC supports data of the type **Coordinate** whereas another IC requires the data in **MapType**. Then the system takes a mediator which accepts **Coordinate** type and produces **MapType** (possibly with the usage of intermediate mediators). If the mediator accepts a different object interface than the **Coordinate** type in spite of probably similar attributes and attribute types, it cannot be identified, because of the different interface names.

We have overcome this problem by extending the standard Ontology Web Language (OWL) and defining a new *mediator profile language* which allows more refined semantic specifications of mediator's capabilities. The syntactic annotation is done by relating the data information to a type system (e.g. XML Schema), whereas the semantic annotation is made by relating data types to concepts of a shared (domain) ontology. Further, every mediator profile embeds textual information as well as a taxonomical classification of its application domain.

The advanced retrieval unit of the COBIDS platform supports different matchmaking facilities and can take advantage of these mediator profiles. Besides simple keyword matching the matchmaking facilities encompass also advanced signatures and concept matching, i.e. they support subsumption relations as well as synonym relations and can create compositions of mediator components (Radetzki 2004). Now, with these possibilities we are able to identify mediator components, even if

their interface names are unequal. However, if the interface names are unequal, but the retrieval unit identifies similarity between mediator components in the concepts or attribute types, then the user has manually to verify these mediator components whether they are appropriate.

3.2 Integration of OGC services

While OPALIS concentrated on highly specialized services, COBIDS steps out into the standardized world. Therefore, one of the most important tasks in the development process was designing ICs integrating OGC web services.

While suitable ICs have to be specified and implemented for each service when dealing with non-standardized services, an IC encapsulating a standardized GI-service has to be built only once. For the interaction with, for instance, an OGC-conform Web Feature Service (WFS) an adequate IC can support any other WFS within the client platform.

Since OGC web services have the same architecture as other non-standardized web services (cf. the W3C Web Services), the realization of appropriate IC can in principle be done in a very straight-forward way. Nevertheless, some new requirements arose for the mediator framework and data handling within the client-platform. Basically, we had to focus on techniques (a) for handling standardized data like GML and (b) for complex mediation mechanisms involving access to remote services.

To represent objects within the client platform, OPALIS uses its special storage components based on a generic format for data representation (gXML). Each IC has to convert the data provided by a service into this generic format. This conversion process is motivated by the fact that each proprietary service usually defines its own proprietary data structure, so that by relying on an independent platform-inherent exchange format we can drastically reduce the number of possible conversions (= required mediators) from $O(n^2)$ to $O(n)$ (converting n formats to one another vs. converting each format to one generic format and back).

Of course, in the context of standardized services – and consequently standardized data exchange formats like GML – this strategy is completely inefficient. As long as data is shared only between OGC-compliant services it makes no sense to give up the GML representation and explicitly convert it back again. Figure 2 illustrates the flow of data between two such services.

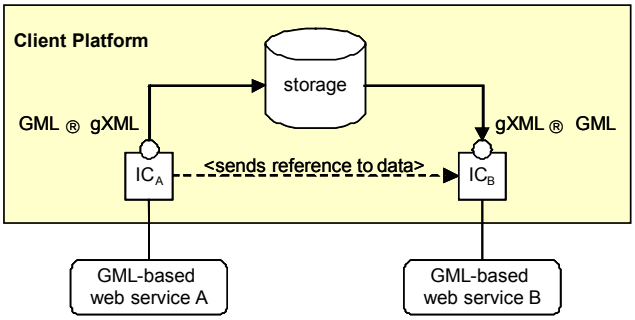

Figure 6: Flow of information between two OGC-compliant web services with the required data formats for the different components

However, we still do need the gXML format inside the client platform, because this is the only way in which objects can be stored. So the gXML format was extended in a way that the (binary) representation of arbitrarily structured data could be directly embedded into a gXML-object, thus avoiding the actual conversion into a format which contains nothing of the original structure any more. In addition, we can now assign unique object identifiers to the objects retrieved from a specific service and integrate additional meta information (e.g. data source, format, etc.). Figure 3 shows an example of GML embedded into a gXML description.

The actual conversion of the received data into one ore more gXML-objects is delegated to the IC, keeping the conversion service-dependent. Note that data is always exchanged between ICs as objects, which can of course be created dynamically (think of a substructure being sent out as a single new object).

With the integration of an IC for OGC Web Map Services, the delivering of complex semantic information in form of a picture was taken into account for the first time. We were faced with the problem that information encoded in a picture could not be accessed by the respective mediators once resampling/conversion became necessary.

For instance, using a map delivered by a Web Map Service as the background image inside a visualization component for vector data is only possible if the map's size and scale meet the requirements of the visualization component. Adjustment of these properties in the mediation process is not possible without renewed access to the original Web Map Service. Again, the mechanisms already implemented in the COBIDS platform enabled us to deal with this new problem efficiently:

1. Image data can be embedded into gXML in a similar way as discussed above in the context of GML. The map's size and scale, a reference to the
2. appropriate IC, the URL of the service and all parameters used in the request can be added to the same gXML-object as additional meta data.

GML:

```
<HYDROGRAPHY fid="HYDROGRAPHY.450">
  <GEOTEMP>
    <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
      <gml:coordinates>10,10</gml:coordinates>
    </gml:Point>
  </GEOTEMP>
  <DEPTH>565</DEPTH>
</HYDROGRAPHY>
```

embedded in gXML:

```
<O id= "WFS.<WFS-URL>.HYDROGRAPHY.450" jc="Hydrography">
  <T>HYDROGRAPHY</T>
  <A n ="GMLFeature" t="binary">
    <HYDROGRAPHY fid="HYDROGRAPHY.450">
      <GEOTEMP>
        <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
          <gml:coordinates>10,10</gml:coordinates>
        </gml:Point>
      </GEOTEMP>
      <DEPTH>565</DEPTH>
    </HYDROGRAPHY>
  </A>
</O>
```

Figure 7: GML embedded into gXML

2. Using this meta data, the mediator framework can easily control the mediation process as to whether a new request needs to be sent to the Web Map Service and the respective mediator can initiate this request via the associated IC. To accomplish this, a new interface was added to the IC dealing with OGC Web Map Services.

As we can see from these results, the integration of the ICs for OGC Web Services has shown the flexibility of the COBIDS architecture and demonstrated how the COBIDS platform can be used as a bridge between proprietary services and the standardized world.

3.3 Integrating local applications

In this section, we show how existing local applications (i.e. tools for simulation, geometric modeling or visualization) can be integrated into COBIDS. Our approach for integration assumes that every integrated application has a corresponding IC providing a mean of bidirectional communication between the controlled application and the COBIDS client. The problem is that the functionality and interfaces of the applications can be completely heterogeneous. Therefore, it is not possible to make a single generic IC, which could be used for interaction with several applications. The ICs are application-specific and use different communication techniques.

The realization of the connection between an application and its corresponding IC depends on the type of the application to be integrated. In general, we have identified four possible strategies for implementing this connection. Figure 4 illustrates them schematically. Here the ICs are represented as thick bounded rectangles, containing a user interface (UI) and implementing the interface of the internal communication component (IComm) of the client platform.

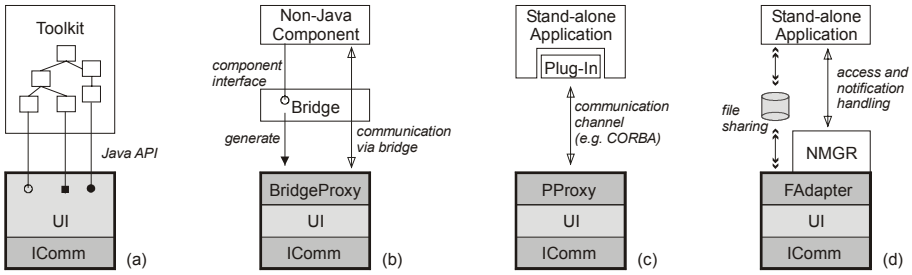

Figure 8: Connecting interaction components with applications

If an application has a Java API or is a JavaBean, then it can be directly accessed by a corresponding IC (Fig. 4.a). In the other cases, the implementation of special bridges or adapters is necessary. The adapters delegate requests between the application's API and its ICs bidirectionally. These adapters can be generated automatically or semi-automatically in those cases where modification of an existing interface is required. For example in the cases b) and c), a straightforward one-to-one mapping of fine-grained operations could be inefficient and give use extremely long message processing times.

Figure 4.b illustrates the case where we either have to deal with a non-Java component, e.g. COM component, or where an existing application has an API that can not be directly accessed from Java. In this case, connecting the component can be realized with help of specialized bridging technology. The bridge is responsible for interaction between different technologies, delegating methods and events by means of automatically generated proxies (BridgeProxy) between the application and its IC. Several commercial bridges are available today. For example, the Java Native Interface (JNI) can be used for the integration of a C++ library. The JNI allows building ICs to operate with applications and libraries written in other languages, such as C, C++, and Assembly. Conversely, we can employ the JNI framework to allow the application to utilize IC's Java objects in the same way as Java code uses these objects. It should be noted that ICs using local applications, which are not written in portable languages like Java, are no longer platform independent and can be used only in distinct environments.

If the application's functionality can be extended, for example the incorporation of plug-ins, like those employed by Netscape or GOCAD, the communication can be organized by developing a plug-in and the corresponding plug-in proxy (PProxy) which communicate by means of a self-defined protocol (Fig. 4.c). Development of an efficient communication protocol is not easy, but can be very important in the case of remote interaction (Shumilov 2003). While the IC currently access objects via remote method invocations, the straightforward use of existing fine-grained operations is inefficient. In general, sending remote messages is much more expensive than using local operation.

In cases where a stand-alone application provides a well-documented file protocol, the integration can be fulfilled by corresponding file transfers (Fig 4.d). Those are controlled by a corresponding file adapter (FAdapter) which interacts with a local file base and a notification manager (NMGR) which guarantees the serializability of file accesses. This component is necessary in the cases where both interaction component and local application may simultaneously access, i.e. read or write the same file. The notification manager is therefore responsible for the management of locks and for notifications concerning modified data. The manager could be also realized by a small script running on the local platform and acting as a channel. Developers of such ICs have to use the native file format defined by the application. Finally, in all other cases, integration is only possible with much overhead.

4. Conclusion

In this paper we have presented the evolution of the OPALIS framework into the COBIDS platform which provides integrated access to heterogeneous standardized and non-standardized distributed services as well as local applications and tools. Integrated use of services and applications is achieved by means of a loose coupling of respective interaction components within a component-based client platform. In this way the platform prevents the global schema integration and in the same time is powerful enough to integrate many different kinds of services and applications regarding of their nature and standards.

By enhancing the mediation facilities, i.e. describing mediator capabilities syntactically as well as semantically, we are able to identify suitable mediators more accurate. That means, mediators can be selected which are not directly support the exact data structures. Further, by extending the generic representation in the way that GML and other binary data can directly integrated into gXML with additional meta information, standardized services can be interleaved more efficient. The meta information can also be used by mediators in order to request distributed services directly. Finally we have identified four possible strategies for integrating local applications efficiently.

The developed technology is universal and can be applied in various domains. The use of this interoperable COBIDS platform with its facilitated exchange of data and models between applications and services leads to improved scientific results.

Acknowledgement

The authors would like to thank Thomas Erdenberger, Sebastian Mancke and Jörg Wagner for valuable discussion and comments.

Bibliography

Andrianasolo, H. (et al) (1999): A Methodology in Detailed Environment Mapping for Viral Disease Survey, ACRS

Bergmann, A., Breunig, M., Cremers, A.B., Shumilov, S. (2000): A Component Based, Extensible Software Platform Supporting Interoperability of GIS Applications, Proceedings of the 12th Int. Symposium "Computer Science for Environmental Protection" of Gesellschaft für Informatik (GI), Bonn, pp. (1)90-102

Bernard, L. (et al) (2003): Ontologies for Intelligent Search and Semantic Translation in Spatial Data Infrastructures, Photogrammetrie - Fernerkundung - Geoinformation (PFG), No 6, pp 451-462

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

Curbera, F. (et al) (2002): Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing, 6(2), pp. 86-93

Radetzki, U. (et al) (2002): First Steps in the Development of a Web Service Framework for Heterogeneous Environmental Information Systems, Proceedings of the 16th Int. Conf. "Informatics for Environmental Protection" (EnviroInfo 2002), pp. (1)384-391

Radetzki, U., Bode, T., Cremers, A. B. (2004): Mediatorbasierte ad hoc Integration autonomer Web Services. Informatik 2004, 34. Jahrestagung der Gesellschaft für Informatik (GI), Workshop: Dynamische Informationsfusion, Ulm, Germany

Shumilov, S. (2003): Integrating existing Object-Oriented Databases with distributed Object Management Platforms. Developed and evaluated on the example of ODBMS ObjectStore and CORBA, PhD Thesis, Department of Computer Science III, University of Bonn, Germany

Wiederhold, G. (1999): Mediation to Deal with Heterogeneous Data Sources, In: A. Vckovski, K.E. Brassel, H.-J. Schek (eds.): Interoperating Geographic Information Systems, Proc. of the 2nd Int. Conf., Zurich, Switzerland, Lecture Notes in Computer Science, no. 1580, Springer, pp. 1-16