Framework for the Development of Environmental Risk Management Services According to the ORCHESTRA Architecture

Martin Schmieder\textsuperscript{1}, Thomas Usländer\textsuperscript{1}, Julian Fischer\textsuperscript{2}

Abstract
This paper presents the ORCHESTRA Service Container Framework (OSCF) which provides an infrastructure for the development, deployment and execution of services according to the architecture of the ORCHESTRA project (http://www.eu-orchestra.org). Due to the architectural approach of ORCHESTRA the framework is based on a core set of generic ORCHESTRA services which are listed and briefly described. Providing implementations of these services is the foundation for the development of applications and services on the thematic level including risk management services. The OSCF is implemented in Java and is based on Apache Axis2 and the XMLBeans data binding. It offers services and components to support the development of new ORCHESTRA-compliant services and ORCHESTRA service networks. The paper gives examples how the OSCF supports service developers. The access control mechanism is shown which illustrates the interaction between elements of the OSCF. Within ORCHESTRA the OSCF has been used by various pilot implementations based on risk management scenarios. As an example one of the resulting service networks is described in more detail.

1. Introduction
ORCHESTRA (Open Architecture and Spatial Data Infrastructure for Risk Management) is an integrated EU project which was initiated in September 2004 due to the increasing awareness of safety issues relating to environmental risk management. Its overall goal is the design and implementation of an open, service oriented software architecture (SOA) as a contribution to overcome the interoperability problems in the risk management domain (Denzer et al., 2005). This paper focuses on the programming aspects of ORCHESTRA by presenting the ORCHESTRA Service Container Framework (OSCF) as a framework for the development of risk management services. As shown in Figure 1 the OSCF and the web services on which it is based follow the ORCHESTRA Architecture and provide implementation support for it.

Figure 1: Relationship between the OSCF and the ORCHESTRA Architecture

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure1.png}
\caption{Relationship between the OSCF and the ORCHESTRA Architecture}
\end{figure}

\textsuperscript{1}Fraunhofer IITB, Fraunhoferstr. 1, 76131 Karlsruhe, Germany, email: schniederuslaender@iitb.fraunhofer.de
\textsuperscript{2}Environmental Informatics Group, Goebenstr. 40, 66117 Saarbrücken, Germany, email: julian.fischer@enviromatics.net
Due to this relationship section 2 of this paper first presents the architectural approach of ORCHESTRA and the layering of the ORCHESTRA Architecture. A core set of generic services is listed in section 3. Section 4 introduces the OSCF and its components. Section 5 gives examples of its usage in providing functional support for service developers. By means of a typical risk assessment scenario section 6 illustrates how ORCHESTRA services can be combined to form service networks realised within various ORCHESTRA pilots.

2. The ORCHESTRA Architecture

The architectural approach of ORCHESTRA is described in the Reference Model for the ORCHESTRA Architecture (RM-OA) that has been published as discussion paper of the Open Geospatial Consortium (OGC) (Usländer, 2007).

The RM-OA basically follows a two-step approach. The first step focuses on abstract specification of interfaces, services and exchanged information in UML on a platform-neutral level. The resulting ORCHESTRA Architecture is still independent of a particular service infrastructure having in mind that service infrastructures usually change in the commercial IT world in periods of around 10 years (OSF-DCE, CORBA, Web Services, …).

The second step is to map the abstract specification to one or more chosen service infrastructures resulting in ORCHESTRA Implementation Specifications. The primary service infrastructure for which implementation specifications are conceived is based on W3C Web Services.

Besides the distinction between abstract and implementation specifications ORCHESTRA provides a clear definition of and distinction between architecture services and thematic services (including risk management services). Figure 2 shows the layering of the ORCHESTRA Architecture.

![Figure 2: ORCHESTRA Architecture](image-url)
Based on existing standards and products for service infrastructures, ORCHESTRA Architecture Services (OA Services) enable the setting up of a generic information infrastructure (so called info-structure). These services provide all those functionalities that can be designed in a generic way, i.e. for any type of risk and environmental application domain and for any type of organisation. OA Services are further classified into two sub-categories:

- OA Info-Structure Services which play an indispensable role in the operation of an ORCHESTRA service network depending on its required characteristics.
- OA Support Services which support the provision of OA Info-Structure Service functionality (as an implementation option) or facilitate the operation of the service network.

On top of this layer ORCHESTRA Thematic Services (OT Services) and thematic applications may be built to enhance the info-structure according to the needs of the user in a thematic environment, in particular for risk management and for crisis management. OT Services provide application domain-specific functionality. However, both within and between different application domains, high-level functions that have a generic nature may be identified. These services are called OT Support Services and provide a generic service that facilitates the development or interactive composition of thematic functionality.

In order to validate the ORCHESTRA Architecture various ORCHESTRA pilots have been defined and implemented based on risk management application scenarios in a cross-border and/or cross-domain context:

- Pan-European assessment of natural hazards
- Floods and forest fire risk prevention assessment in the Tordera basin (Catalonia, Spain)
- Risk assessment for the road network in the French-Italian border region
- Environmental risks caused by ship traffic in the German Bight. As an example for ORCHESTRA based risk assessment this pilot is described in more detail in section 6.

### 3. Overview about the ORCHESTRA Architecture Services

As the OA Services are the foundation for thematic services and applications, a subset of them are described briefly in the following table.

**Table 1:** A subset of ORCHESTRA Architecture Services

<table>
<thead>
<tr>
<th>Service</th>
<th>Overview Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OA Info-Structure Services</strong></td>
<td></td>
</tr>
<tr>
<td>Authentication Service</td>
<td>Proves the genuineness of principals (i.e. the identity of a subject which may be a user or a software component) using a set of given credentials.</td>
</tr>
<tr>
<td>Authorisation Service</td>
<td>Provides an authorisation decision for a given authorisation context.</td>
</tr>
<tr>
<td>Catalogue Service</td>
<td>Provides the ability to publish, query and retrieve descriptive information (metadata-information) for resources (i.e. data and services) of any type. May be used as a data catalogue, service registry or both.</td>
</tr>
<tr>
<td>Feature Access Service</td>
<td>Provides the ability to select, create, update and delete feature instances and feature types available in a service network. Features provided are instances of feature types defined in an ORCHESTRA Application Schema (OAS).</td>
</tr>
<tr>
<td>Service</td>
<td>Overview Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Map and Diagram Service</td>
<td>Visualises, symbolises and enables geographic clients to interactively visualise geographic and statistical data. Creates an image document which may be a map, a diagram or a thematic map.</td>
</tr>
<tr>
<td>Service Monitoring Service</td>
<td>Provides an overview about service instances currently registered within a service network, e.g. actual status (e.g. running, stopped, offline), statistical information (e.g. average availability, response times).</td>
</tr>
<tr>
<td>User Management Service</td>
<td>Creates and maintains subjects (users or software components) including groups (of principals) as a special kind of subjects.</td>
</tr>
</tbody>
</table>

**OA Support Services**

<table>
<thead>
<tr>
<th>Service</th>
<th>Overview Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate Operation Service</td>
<td>Changes coordinates on features from one coordinate reference system to another.</td>
</tr>
<tr>
<td>Service Chain Access Service</td>
<td>Supports the creation of an executable service instance based on an explicit description of a service chain (e.g. in BPEL) and its registration in a service catalogue. The chain can then be executed as a single service (outside the scope of this service).</td>
</tr>
</tbody>
</table>

4. **The ORCHESTRA Service Container Framework**

In the previous sections it was shown that thematic applications and ORCHESTRA Thematic Services are based on a core set of generic OA Services. Providing implementations of these OA Services is the foundation for the development of applications and services on the thematic level. This naturally leads to the idea of a service framework. The framework combines default implementations of OA Services which can be adapted and configured to establish a service network including thematic services and applications. Thus, the framework allows establishing a “container” of service instances.

Therefore, during the implementation phase of the ORCHESTRA project, an ORCHESTRA Service Container Framework (OSCF) was developed. It is based on the chosen service platform, i.e. W3C Web Services (W3C 2004) with message exchange based on the protocol stack TCP/IP, HTTP and SOAP. It includes a set of ORCHESTRA-compliant services some of which are implemented as adapters for external non-ORCHESTRA services, e.g. OGC services.

However, the OSCF is more than just a set of independent service implementations. It provides also components which can be used as building blocks to facilitate the development of new services, both on the generic and on the thematic level. Such components can also be used on the client side in order to provide a convenient programming interface to access a service. As most ORCHESTRA services were implemented in Java it was decided to make the OSCF a Java-based framework.

The OSCF was not implemented from scratch but is based on the well-known Apache Axis tool which provides common functionality for web services in general. There exists a re-design of Axis called Axis2 (Axis2 2007). Axis2 version 1.0 has been released during the ORCHESTRA implementation phase in summer 2006. It allows easy integration of the XMLBeans data binding for the mapping between XML messages and Java classes. By using this data binding some problems could be solved that were encountered for XML schemas based on the Geography Markup Language GML. For that reason and because of its additional enhancements and flexibility Axis2 was chosen as basis for the OSCF.

Figure 3 outlines the principle parts of the OSCF. There are three types of components:

- Components which are illustrated in dark colour directly belong to the OSCF.
- Components in light colour are provided or generated by Axis2 and are used as basis for the OSCF.
Unfilled boxes symbolise components which have to be provided by developers in the process of implementing ORCHESTRA services and clients.

Figure 3: ORCHESTRA Service Container Framework

The OSCF comes with an initial set of services. As an example, the figure shows UAA services (dealing with access control) and a Catalogue Service which acts as service adapter for an external service like e.g. an OGC catalogue. These services are deployed in the Axis2 web application (symbolised by the surrounding box in the figure) which acts as service container.

Incoming SOAP messages are dispatched by means of Axis2 handlers to the target service. Such a service is represented by a corresponding Axis2 archive (aar-file) whose internal structure is illustrated in the figure by the service on the left. The service’s message receiver transforms the XML contents of the message to Java parameters and calls the requested Java operation. The available operations are symbolised by small boxes \( op_1 \ldots op_N \) within the skeleton. Both message receiver and skeleton are Axis2 components which are automatically generated from a WSDL description of the service interface by means of the WSDL2Java tool of Axis2. Now, operation specific code is executed which is provided by the service implementer and which is based on common functionality offered by the ORCHESTRA Service Library (OSL). The OSL belongs to the OSCF and is added to each service as a set of Java jar-files.

In addition, the figure shows in the upper part two clients accessing the services. The client on the right hand side uses a service-specific stub to call a service operation. Like a skeleton a stub is generated by WSDL2Java. A stub maps operation calls to SOAP messages and is part of the OSL. The client on the left does not use the OSL as it may e.g. not be a client written in Java. Instead it may use another languagespecific utility to construct SOAP messages fitting to the WSDL of the target service.

On the way between client and service each request and response message is passed through a stack of modules which may pre- or post-process the message. The figure shows the modules above the services.
A module is represented by an Axis2 module archive (mar-file) and performs tasks which are common to many services and can be handled in a uniform way. The OSCF comes with an initial set of modules, the most basic one being a logging module which logs incoming and outgoing messages. The OSCF is extensible with respect to implementing and adding new modules as well as new services.

5. Functional Support

The main elements of the OSCF are services, modules and the OSL. As an example for modules, the logging module has already been mentioned. This section gives examples on how the OSL provides functional support for service developers. In addition, as an example of the functionality of services, the access control mechanism is shown. This also illustrates the interaction between elements of the OSCF.

5.1 Default Implementations of Service Interfaces

The OSL is designed as a typical Java framework to support development of ORCHESTRA-compliant services. As ORCHESTRA defines some interfaces which are common to all or at least multiple services, the OSL provides default implementations for these interfaces. Here are two examples of such interfaces.

The Service Capabilities interface defines an operation `getCapabilities` to deliver service meta-information which can be stored by means of a Catalogue Service to be used for various purposes like e.g. service discovery or invocation. In order to implement the operation, a service developer can delegate the call to a `GetCapabilities` Handler being part of the OSL as shown in Figure 4. For each supported meta-information schema a `Capabilities Provider` can be registered to deliver capabilities according to that schema. A default implementation of such a provider reads the capabilities from an XML resource file.

Another common interface is the Monitorable interface used in the context of service monitoring. The `getStatus` operation of the interface allows retrieval of service status information which is collected periodically by the Service Monitoring Service in order to support visualisation of statistics and status of the whole service network. A service developer can make use of a `GetStatus` Handler as shown in the same figure. The handler is part of the OSL and retrieves status information in response to a `getStatus` request.
5.2 Access Control

Access to resources (services or data) may need control by authentication and authorisation mechanisms. For this purpose, ORCHESTRA defines three services: the User Management Service, the Authentication Service and the Authorisation Service, together referred to as UAA services. The respective abstract specifications are by intention specified at a high level of abstraction in order to be able to cope with various established UAA mechanisms. A platform specification then defines the concrete UAA concept which is realised. Within the OSCF authentication by username/password and role-based authorisation is implemented. Once corresponding users have been registered by means of the User Management Service, controlled invocation of a service operation is done by the following steps shown in Figure 5.

![Figure 5: Controlled access to a service operation](image)

A client uses the Authentication Handler of the OSL to send user name and password to the Authentication Service (step 1). If successful a new session is initiated represented by session information which is returned to the client. The client then calls the service operation (step 2) including as proof of its identity the previously obtained session information. On the service side the Authorisation Handler provided by the OSL checks whether the session is valid (step 3) and whether the authenticated caller is allowed to perform the operation (step 4). Only if both checks are successful the operation itself is performed (step 5). Note that transmission of session information needs protection which may include digital signing and/or encryption of message contents.

6. An Example Use Case – The Orchestra Pilot “German Bight”

The OSCF provides components and services to build an ORCHESTRA service network. Various such service networks were realised within the context of ORCHESTRA pilots. In the following the ORCHESTRA pilot “German Bight” (Bügel, Usländer, 2006) is described as an example.

The pilot enables end users and providers of environmental information systems access to and creation of multi-risk maps for risks caused by ship traffic in the German Bight. The generation of the maps is based on integrated use of information about risk factors (e.g. chemical pollution), shipping routes, environmental databases (e.g. about distribution of species), numeric models (e.g. currents, dispersion) and cartographic data. This information is exploited in simulation models. In “what if” scenarios (e.g. introduction of new shipping routes) the generated risk maps can visualize potential risks for certain areas.

The pilot realises an ORCHESTRA service network which comprises the following services:

- an ORCHESTRA Catalogue Service (CS)
- one or more ORCHESTRA Simulation Management Services (SiMS)
• one or more Simulation Data Services (SDS) providing access to simulation data by means of ORCHESTRA-conformant interfaces, like the Feature Access Service (FAS) interface
• an ORCHESTRA Map and Diagram Service (MDS)
• ORCHESTRA UAA Services for access control to data sources

Besides, there is a web based main application allowing the authorised user to retrieve meta-information about all available services from the Catalogue. Thus, the user discovers the services and decides about which SiMS to select for a simulation run, and which SDSs should deliver the required data. After the simulation phase, the main application converts the results computed by the SiMS into a format needed for further processing steps, e.g. visualisation of maps in GML format, and stores the data at an SDS in order to be accessible through an SDS/FAS interface. Finally, the main application uses the Map and Diagram Service to generate the risk map from the simulation results.

7. **Benefits and Outlook**

The OSCF provides an efficient implementation of services and components supporting both development and deployment of ORCHESTRA-compliant services in order to establish a network of services and applications in the risk management domain. The OSCF allows re-use of common service functionality by means of a flexible, Java based framework. Among its core functionality support for access control and service monitoring is provided. The OSCF can easily be extended by implementing and adding additional services and modules. It has been used by various ORCHESTRA pilots based on risk management scenarios. The gained experiences will further improve the OSCF. It is planned to offer the software as open source framework in the near future.

**Acknowledgment**

This work has been funded by the European Commission through the Integrated Project FP6-IST 511678 ORCHESTRA (see [http://www.eu-orchestra.org](http://www.eu-orchestra.org)). Special thanks to the members of the sub-project 3 “Open Architecture” for their good co-operation.

**Bibliography**


