ARCHITECTING DISTRIBUTED GEO-INFORMATION SERVICES: BEYOND DATA INFRASTRUCTURES

Javier Morales, Mostafa Radwan

International Institute for Geo-information Science and Earth Observation (ITC), Enschede, The Netherlands
jmorales@itc.nl, radwan@itc.nl

Commission IV, WG IV/4

KEY WORDS: System Design, Specification, Abstraction, Requirements, Open Systems, Infrastructure, Distributed, Value-added.

ABSTRACT

During the last decade technological developments have facilitated the access to geo-information and have made it easier to manipulate, reducing the effort and skills required to exploit it effectively. As a direct consequence of this trend, a more sophisticated spatial awareness has developed among the general public resulting in geo-information being required to support many daily activities. This is creating a growing dependence of people and organisations on geographic information, which has converted geo-information into a precious resource. To support this new working environment, the role of the traditional Geographic Data Infrastructure (GDI) has to change, from being a simple data discovery and retrieval facility to become an integrated system suitable for the provision of customised information and services.
A service is a collection of functions or operations organised in a way that they exhibit a behaviour of value to a user. The functions used within a service are provided by independent entities, and these functions are available at different system nodes. Such services have to be formally specified before they can be properly implemented and used in a composition. The internal structure of the service (i.e., the service realisation) describes how different components interact to provide a desired information service.
In this paper we present our proposed architecture for a geo-service infrastructure and a design methodology that facilitates the specification and access of distributed geo-services over the Internet (Web Services, Internet GIS, etc) in the context of the GSI concept. The methodology proposes a repository service for creating, updating, validating, accessing and sharing service models and service instances, using an XML-based interchange format.

1 INTRODUCTION

During the last decade technological developments have facilitated the access to geo-information and have made it easier to manipulate, reducing the effort and skills required to exploit it effectively. As a consequence, the use of geo-information is expanding beyond the traditional users (government organisations), to include new users communities (telecommunication industry, emergency services, and tourism, a.o.).

As a direct consequence of this trend, a more sophisticated spatial awareness has developed among the general public resulting in geo-information being required to support many daily activities. This is creating a growing dependence of people and organisations on geographic information, which has converted geo-information into a precious resource. However, more users that require geo-information for different applications means geo-information providers have to deal with a large variety of specific information requirements to satisfy. Dealing with these requirements is a challenging task, specially taking into account that the way geo-information information is perceived, expected and used depends very much on the current forms and shapes of markets and technology, which make these requirements very dynamic.

Geo-information providers have realised that satisfying today's geo-information market, that is a variety of users with appropriate geo-information services, in large volumes and in near real-time mode, goes beyond the capacity of 'single' organisations. Therefore, these organisations are seeking for mechanisms that enable them to work together in a more collaborative way. To support this new working environment, the role of the traditional Geographic Data Infrastructure (GDI) has to change, from being a simple data discovery and retrieval facility to become an integrated system suitable for the provision of customised information and services.

Our research is therefore focusing on the development of mechanisms to describe, combine and manage independent collections of services. This is because, what is required by today's geo-information industry is a system that enables geo-information providers to cooperate and work together in an integrated way. Such system should make it possible for providers to publish and share not only data but business goals, processes, operations, resources, value-added products, etc., unbundling in this way the functionalities of current stand-alone geo-information systems, and making them available as independently developed, yet interoperable autonomous services. We called such a system a *Geo-information Service Infrastructure* (GSI).

A GSI is a system from which specialised geo-information products and services can be obtained by exploiting the

artefacts of an infrastructure of interconnected nodes that include, among others, data repositories, data brokers, service providers, service brokers and clients. Within a GSI, large geo-processing tasks are achieved by combining or chaining artefacts located along the distributed nodes. The GSI system enables Geo-Service Providers to make use of each others functionality to supply a wide range of services and possibly to reach larger groups of users.

For this system to work service providers must create and make available descriptions or models of their individual services, which can be used as the basis for the specification of complex services. A service repository is therefore required as a central component of the system. The service repository supports the exchange of service models between different service providers. If a model properly describes an individual service, that is, with the relevant information at the correct level of detail to enable one to determine what it does and how to access the function it provides, then this service can be easily reused. By reuse of services, we mean the inclusion of a previously designed service in multiple combinations of more specialised service definitions.

2 GEO-INFORMATION SERVICES

The role of the GDI is currently changing, from it being a simple data discovery and retrieval facility to become an integrated system suitable for the provision of customised information and services. We consider a service as the contribution of a system or part thereof to its surrounding environment. This contribution can be defined in terms of data, operations, processes, resources, value-added products or any combinations of them. This For the sake of simplicity we use the term services to denote geo-information services.

Normally developers address the issue of designing complex services by stringing together groups of functions in an ad hoc manner. This approach may satisfy a particular need but doing this separately for different services hampers reusability. Moreover, lack of descriptions of the solutions obtained makes it hard to aggregate solutions to execute complex tasks.

Research is therefore focusing on the development of mechanisms to describe, combine and manage independent collections of services. Here, we introduce a concept that aims at facilitating the generation of sophisticated value-added services. We call it the Geo-information Service Infrastructure or GSI for short. The idea of the GSI is that elementary services can be described, accessed, combined and managed to deliver complex content. Within the GSI, a common method is used to describe elementary services and their interfaces, and then these services are made available for users to create service chains that perform complex geo-processing tasks.

2.1 GSI

A Geo-information Service Infrastructure (see Figure 1) is a system from which specialised geo-information products and services can be obtained by exploiting the artefacts (data, operations, processes, resources, value-added products or any combinations of them) of an infrastructure of interconnected nodes that include data repositories, data brokers, service providers, service brokers and clients. We call the above mentioned artefacts of the infrastructure architectural elements (see section 3). This service framework builds upon the layer of interoperability of information as defined by the OpenGIS implementation specifications (OGC, 1999), therefore separating the actual implementation of services from their definitions and the perception of these services by the users.

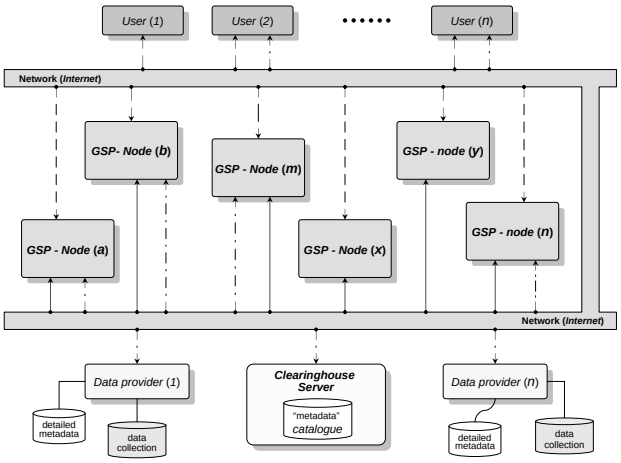

Figure 1: The GSI system concept

Large geo-processing tasks can be constructed by combining or chaining sets of architectural elements located along the distributed nodes. Such combinations of architectural elements provide diverse functionality that satisfies particular sets of requirements. Every architectural element represents an artefact that has an economic value; these architectural elements are assembled to perform operations within the infrastructure, resulting in a specialised architectural element (artefact) that has a value equal or larger than the combined value of the architectural elements used. This architectural approach can be regarded as a "value-added system." By chaining architectural elements one can provide a service. A service is defined as a behaviour of value to the user, which is accessible or instantiated through interaction points (Quartel et al., 2002). This behaviour is exhibited through an appropriate combination of elementary architectural elements.

In order to bind multiple architectural elements into a chain that accomplishes a large geo-processing task, a proper description of the participating architectural elements is required. These descriptions focus on exposing the artefact's internal behaviour, its intended effect and its interaction points or points of composition. These descriptions, which are presented as instances of well-defined models, make it possible to interchange and reuse architectural elements. We call these descriptions system metadata; they are stored and made accessible through a service repository.

The GSI system enables Geo-Service Providers (GSPs) to make use of functionality offered by others to supply a

wide range of services and possibly to reach larger groups of users. Figure 1 illustrates the interactions that take place as GSP-nodes provide services to their users.

Users interact with the different GSP-nodes to request their specific services. Figure 1 shows these interactions as dashed-lines. GSP-nodes may make use of architectural elements available in other GSP-nodes in order to realise a particular service. These interactions between GSP-nodes are shown in Figure 1 as solid lines running from node to node. All connections mentioned here are established through a network.

At the bottom of Figure 1 we can see that additional data collections located at non-GSP-nodes may still be accessed, if needed, either by users or service providers. This is achieved by making use of the conventional data discovery functionality, of the clearinghouse server. These interactions appear in Figure 1 as dashed-dot-lines.

2.2 GSP-node

Figure 2 shows the internal configuration of a GSP-node. The *service repository* contains the descriptions of available architectural elements (service descriptions), either data definitions, process definitions, or previously assembled service chains.

The *geo-processing units* are responsible for the execution of the various functions of the node. These units use *geo-data* and *applications* during operation as specified in the definitions stored in the service repository. The *service design unit* is in charge of defining how the different services are realised. The underlying principles of this architecture are based on OGC technical baseline specifications for OpenGIS Services (OGC, 2002).

The process of generating service chains within the GSI can be broken down into three major activities: defining and registering elementary services, assembling a service chain and delivering the results. Three different roles can be identified from these activities: service providers, service consumers and end users.

Service providers are responsible for describing and making their elementary services available for others to use. We denote the entities that provide these elementary services as components. Service providers make use of a framework (design methodology) that enables the modelling of these components. These models act as descriptors that specify the function and the interaction point(s) of the components.

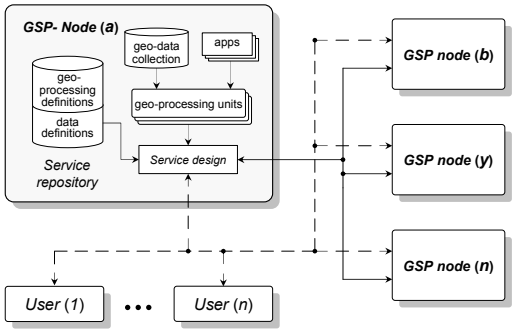

Figure 2: GSI-node internal structure

Service consumers use available components to design more complex services. Service consumers make use of the same framework used by the service providers to assemble individual components into chains. They define these chains by adding control components that govern the relations between the elementary components used in the chain.

These control components help ensuring that the constraints and conditions defined at the interaction points of the elementary components are satisfied. The resulting chain is described as a *service realisation*. The service can be realised by instantiating the behaviour portrayed in the specification. End users trigger the definition and execution of service specifications by posting requests to the system.

3 ARCHITECTURAL ELEMENTS

To deal with the issues concerning the internal structure of a service, we have to identify and characterise the architectural elements within a distributed geo-information system. First of all we ignore details of implementation and communication of these elements in order to focus on the functions they provide and the constraints that apply to their interactions with one another.

Architectural elements basically encompass the different components of the system, however, here we refer to a component in a slightly different manner than the way this term is used by the software engineering community. The term component in software engineering disciplines refers to a self-contained unit of independent deployment, with well-defined interfaces that has no persistent state (Szyperski, 2002). Usually, a component provides a particular function or group of related functions. A component is a reusable unit of composition that can be used to form applications with other components in the same or other computers in a distributed network.

We refer to a component not only to represent units of software, but in general to any architectural element of the system that provides a function required in a larger processing chain. Such processing chain is formally described in a service realisation. Components in this context can thus be used, for example, to refer to some abstract representations of data. or to an action or set of actions performed by a human, and that may yield a necessary result or provide a required function. In order to create abstract representations of components of geo-information systems, GSDM makes use of architectural elements. GSDM ignores the details of component implementation to focus on the roles of components, the constraints upon their interaction with other components, and their use of data.

We distinguish three different types of architectural elements within a GSI system (see Figure 3):

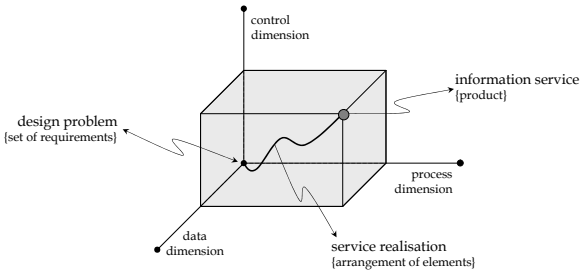

Figure 3: Modelling dimensions

- data elements;
- processing elements; and
- connecting elements.

The *data elements* represent the information that is used, manipulated and/or generated by the system. The *process elements* represent the geo-processing capabilities of the GSI system, which can perform transformations on data elements. The *connecting elements* are like mediators, they represent the relationships between other elements. The connecting elements represent the conditions and constraints that define how the different elements may interact and how they are organised with respect to each other in a service specification. A similar approach to system elements can be found in (Perry and Wolf, 1992, Fielding, 2000).

To illustrate the differences between these elements we can use metaphorical example. Consider the games of soccer and handball. The two games are similar in structure, they use a ball as data element and players as processing elements. The difference between them lies in how these elements are allowed to interact with each other, which are the context and rules of the games (the connecting elements). These connecting elements are defined by game designers based on what they want to achieve with the interactions.

Separation of concerns is the principle behind this modelling dimensions. For example, by separating the concerns on the nature and state of data from the data processing concerns, we simplify the processing elements allowing them to change and evolve independently. In the same way portability of the data is improved by avoiding that the data remain encapsulated and hidden within processing elements. However, the drawback is that we lose the advantages of information hiding and therefore a mechanism is required for processing elements to identify and understand the data types.

The origin of the reference system determined by the modelling dimensions in Figure 3 represents the starting point of a development process. The process is activated by a design problem. Design problems represent the various inputs (requests) to the GSI system that have to be converted into services. A service has to be specified before it can be realised. The path taken through the cube shown in Figure 3 corresponds to the functional specification of a service that satisfies a specific design problem using an appropriate arrangement of components. The execution of a predefined service specification generates the desired service.

4 GSI REPOSITORY

Our approach to geo-information services design focuses on the use of conceptual models as an intermediate step in the development process, which sit in between requirements and the actual implementation. The purpose of this step is to obtain a conceptual description also called abstract specification of system functions (internal or external). This is done solely to enable and facilitate reuse and to enhance flexibility.

The main benefit of these models is to serve as the basis for the specification of complex services. If a model properly describes an architectural element, that is, with the relevant information at the correct level of detail to enable one to determine what it does and how to access the function it provides, then this architectural element can be easily reused. By reuse of architectural elements, we mean the inclusion of previously designed element in multiple service definitions. We use constraint-oriented composition to combine multiple architectural elements (Morales Guarin, 2004). This way it is possible to assemble large processing chains by correctly combining models of (generic) architectural elements to form more specialised specifications (service models) with the intention of provide more specific or particular services (see Figure 4).

Since the models of architectural elements prescribe the behaviour exhibited by individual elements, a service model can be used to choreograph the realisation of the service specified in the model. Additionally, once a service model is available it can itself be reused, as a generic element, in another definition as a part of a yet more specialised service. Reuse and flexibility is ultimately achieved by developing, sharing and integrating well-defined models of all composing elements of the system.

For this approach to work, models need not only to be interchanged between participants, but they also have to be understood by all parties involved. This can only be

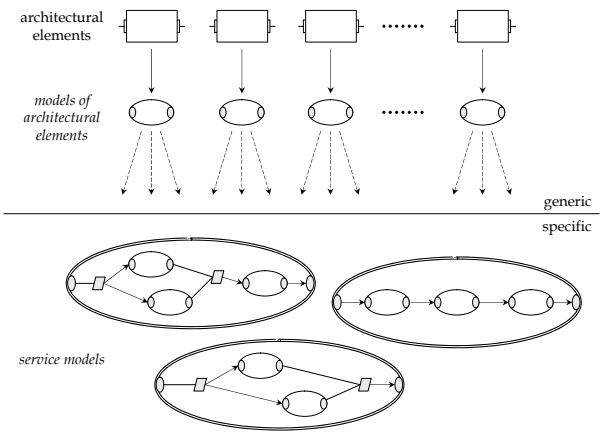

Figure 4: Architectural elements, element models and service models

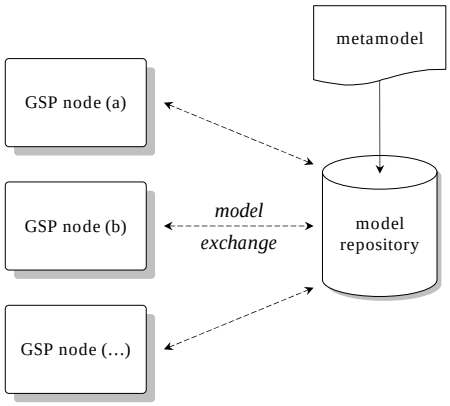

Figure 5: Role of the metamodel in the GSI architecture

achieved if the models are based on the same metamodel. Such metamodel should therefore provide a rigourous abstract syntax for defining models. Figure 5 shows the role of the metamodel in the GSI architecture. The metamodel enables the implementation of a repository where compliant models of GSI services can be stored. Hence, the repository becomes the central component in a GSI system. The repository supports the exchange of models between different service providers, thereby facilitating the use of these models in combinations to form more complex service models that address specialised sets of requirements.

A metamodel defines the set of design concepts and their relationships, which one can use to produce models of some system according to a specific objective. A design concept, also called modelling concept, is a building block that can be used in the construction of a model. A design concept represents one or more related properties of a system or system part that are considered relevant in the design of a system. The complete collection of design concepts should allow one to model all relevant system properties. Relationships between design concepts define the possible ways in which a model can be constructed from of these concepts.

Which metamodel should be used depends on the modelling needs of every particular project. For the purpose of designing GSI systems, we require design concepts suitable for specifying the system's functionality and its environment, the system's internal structure in terms of its composing parts or subsystems and their relationships, and the contribution of each part to the system's overall functionality. This translates into the following modelling needs:

- to represent the system, its logical or physical parts and any external thing that interacts with the system, which could be a person, another system, etc., as single entities capable of exhibiting behaviour;
- to represent the locations where interactions between different entities occur;
- to represent behaviour according to different related abstraction levels;
- to be able to discriminate between the behaviour and the entity that carries the behaviour, such that an entity could potentially exhibit multiple behaviours.
- to be able to structure behaviour into units behaviour and their relationships;
- to represent anything that is used or produced in a behaviour.

Additionally, the selection of an appropriate set of design concepts should adhere to the following quality principles (van Sinderen, 1995):

- *consistency*, which requires that concepts should be consistent in their representation of the aspects in the real world;
- *orthogonality*, which requires that distinct concepts should be used to represent different aspects;
- *propriety*, which requires that concepts should be proper to the modelling needs;
- *generality*, which requires that concepts should be of general purpose in a given domain and the complete set of concepts be sufficient to cover the needs of the domain.

5 METAMODEL FOR GSI

The types of service provided by a GSI system spread along the whole geo-information value chain. This value chain starts with identification of information sources that are used in different geo-processing tasks to create value added geo-information products. These products are subsequently used in various types of analysis with the purpose of deriving new information that is not directly obtainable from the sources. In most of the cases diverse combinations of tasks and information sources are required to solve specific problems and help user communities to make sense of the geographical world that surrounds them.

The structural organisation of this work is consequently formed by arrangements of independent functions and data sources organised in such way that large geo-processing tasks can be accomplished. Any of these arrangements defines a specific behaviour. This behaviour is accomplished through the creation, manipulation or transformation of some items or artefacts, and must be carried out by entities within the system.

According to these criteria and to the modelling needs mentioned in the previous section, we need a metamodel to be able to build repositories where our models can be properly, structurally and consistently stored and retrieved. We introduce a metamodel to be used in the creation of repositories to store our models (see Figure 6). This metamodel is a specialisation of the ISDL metamodel (Quartel, 2003). The metamodel is organised in a number of classes, where each class addresses a group of related design concepts.

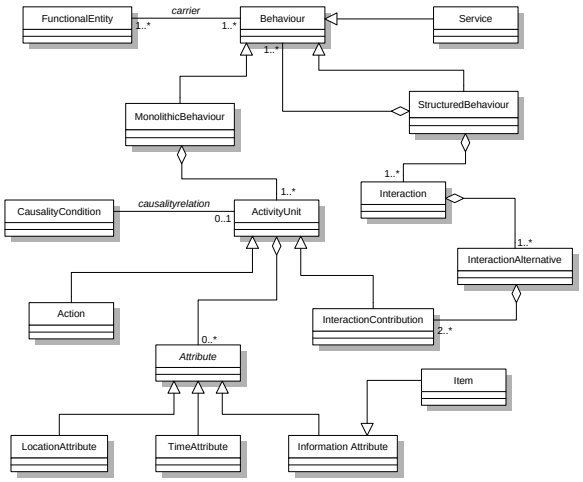

Figure 6: A metamodel for GSI services

The abstract class *behaviour* defines the behaviour concept, which models some type of system behaviour. A behaviour is a (partial) description of the system that describes a distinct part of that system functionality.

An instantiation of a behaviour results in a *service* of value to a user. Multiple behaviours may provide the same service. Behaviours are associated with functional entities. *Functional entities* are capable of exhibiting the characteristics defined in a behaviour. A functional entity represents a logical or physical part of the system capable of executing behaviour in the real world. A functional entity executes behaviour by itself or in cooperation with other functional entities. A behaviour can be carried by more than one functional entity. A functional entity can exhibit more than one behaviour.

Two sorts of behaviour types are distinguished: *structured behaviour*, which are compositions of one or more related smaller behaviour; and *monolithic behaviour*, which are not further decomposed into smaller behaviours.

A monolithic behaviour consists of a group of related *activity units* that can take the form of actions or interaction contributions. An activity unit represents an atomic piece of work at a given abstraction level.

An *action* represents an activity that is defined entirely within a single behaviour. An *interaction* represents an activity in which two or more declared behaviours participate (cooperate). An *interaction contribution* represents the participation of a behaviour in some interaction.

At higher levels in the behaviour hierarchy, the participation of one or more monolithic behaviours in some interaction may be represented by the participation of the structured behaviour in which these monolithic behaviours are defined as sub-behaviours. This may be applied recursively to structured behaviours that are defined as sub-behaviours, such that the participation of a structured behaviour in some interaction may represent the participation of monolithic and structured sub-behaviours.

Alternative groups of sub-behaviours may participate in an interaction. Therefore, an interaction is defined as a set of one or more *interaction alternatives*, where each alternative represents an optional group of participating sub-behaviours.

A *causality condition* is associated with each activity unit. This association is called a causality relation. A causality condition defines the type of relation between two or more activity units. This relation is used to specify how the occurrence or execution of activity units depends on the occurrence or non-occurrence of other activity units.

The completion of an activity produces some result that can be manipulated by other activities. An activity unit can have attributes. These attributes represent the result that is established by an activity unit.

Three attributes are defined:

- the *information attribute*, which represents the product (typically some information) that has been produced by the activity unit;
- the *time attribute*, which represents the time moment at which the product is available;
- the *location attribute*, which represents the location where the product is available.

Items are a special class of the information attribute. Items represent the information that is directly manipulated by an activity unit. The type of manipulation that can be performed on an item by an activity unit are create, use, change or destroy the item.

The metamodel described here only allows for the definition of an abstract syntax for the design concepts used in the creation of GSI models. The semantics of the concepts is provided by the ISDL modelling language (citeISDL). For further detail on the ISDL metamodel, we refer to (Quartel, 2003).

REFERENCES

Ferreira Pires, L., 1994. Architectural Notes: a Framework for Distributed Systems Development. PhD thesis, University of Twente, Enschede, The Netherlands. Ph.D. Thesis, no. 94-01.

Fielding, R. T., 2000. Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, California.

Hadjiefthymiades, S. and Merakos, L., 2002. PoLoS: A universal platform for the development of LBS and other data services. In: Proceedings of the IST Cluster Meeting for Engineering of Service Functionality, Bucharest, Hungary.

Hull, R., Benedikt, M., Christophides, V. and Su, J., 2003. E-services: A look behind the curtain. In: Proceedings of the twenty second ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, ACM Press, San Diego, California, pp. 1–14.

Morales Guarin, J. M., 2004. Model-driven Design of Geo-information Services. PhD thesis, University of Twente, Enschede, The Netherlands. CTIT PhD-thesis no. 03-61, ITC dissertation no. 110.

Morales, J., Ferreira Pires, L. and van Sinderen, M., 2002. Model driven geo-information systems development. In: Proceedings of the Sixth International Enterprise Distributed Object Computing Conference EDOC 2002, IEEE Computer Society, Lausanne, Switzerland, pp. 155–166.

OGC, 1999. The opengis abstract specification overview. Version 4, Open GIS Consortium, Inc.

OGC, 2002. The openGIS Abstract Specification Topic 12: OpenGIS Service Architecture. Version 4.3, Open GIS Consortium, Inc. Editor: George Percivall.

OGC, 2003. OpenGIS Reference Model. Version: 0.1.2, No. OGC 03-040, Open GIS Consortium Inc. Editor: Kurt Buehler.

OMG - Architecture Board, 2003. MDA guide. Version 1.0.1, Object Management Group.

Perry, D. E. and Wolf, A. L., 1992. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes Vol. 17(No. 4), pp. 40–52.

Quartel, D., 2003. ISDL meta-model and repository. Draft, Enschede, The Netherlands. Arco project No. Arco/WP1/N001/V01.

Quartel, D., Ferreira Pires, L. and van Sinderen, M., 2002. On architectural support for behaviour refinement in distributed systems design. Transactions of the SDPS Journal of Integrated Design and Process Science Vol. 6(No. 1), pp. 1–30.

Szyperski, C., 2002. Component Software: Beyond Object-Oriented Programming. 2^{nd} edn, Addison-Wesley Pub. Co., Essex, England.

van Sinderen, M., 1995. On the Design of Application Protocols. PhD thesis, University of Twente, Enschede, The Netherlands. Ph.D. Thesis, no. 95-04.