ABSTRACT

The emergence of web-based geoportals for accessing, and subsequently visualizing, geo-spatial information has created new areas of research for cartography. The design of many new geoportals is based on distributed architectures, a framework that is well-suited for integrating cartography into larger, multi-domain projects. The concept of a distributed architecture describes a network of loosely-coupled services that communicate via standard interfaces. Within this framework, a cartographic service can be integrated into an overall system linking multiple databases and services. Because the cartographic services are developed independently from the other services in the network, they are able to focus on providing high quality visualizations based on well-known and proven cartographic principles. This paper examines the role of cartography in the development of geoportals and how current cartographic research can be influenced by the recent focus on distributed processing.

INTRODUCTION

Monolithic Applications are OUT - Distributed Environments are IN. This statement should not take many Web application developers by surprise. In the last years, we are seeing a steady decline of stand-alone applications that function independently of outside influences. In there place we are finding an ever-increasing amount of applications that are built upon the principles of distributed processing in which a single application is composed of a distributed network of services. These services can be built independently and function independently from other services; but, when placed together in a network of client-server relationships, they can be used to build a multi-faceted, multi-domain application. The advantages to this system are that each domain scientist can now focus directly on their area of interest and develop the best possible services for their needs. This means that cartographers can now concentrate on services that provide high-quality map rendering functionalities that follow solid cartographic principles. No longer will cartographers have to worry about developing all the services functionality necessary for use in a Web-mapping application, and vice versa, non-cartographers can use open map-server technologies developed by cartographers for visualising their information via the Internet.

As part of the expanding use of distributed processing, this paper will explore the concept of independent services functioning as part of a service network. It will ask, and answer questions that are important for cartography, such as: how exactly can cartography benefit from the use of distributed processing? How can we contribute to the overall development and use of GI services in distributed processing? And in an attempt to expand the discussion of distributed processing, how do notions of coupling, standards usage, and interoperability come to play in this new work environment. In particular, this paper will focus on the development and use of cartographic services for the production and management of atlases as well as the rendering of individual map elements.

DISTRIBUTED PROCESSING AND MODULARITY

These service networks are a long way away from traditional methods of software development. In Figure 1, Tsou and Buttenfield (2002), examine different possible architectures that can be used for application programming. Traditional architectures, shown in the left-most diagram, involve single monolithic centralized software packages that come pre-packaged with the individual components hard-coded together. An individual user is not able to manipulate the software in any meaningful way.
To begin the discussion of distributed process, we must understand what we mean when we talk about distributed processing and services. First, we will examine the term *modularity* and how this concept is the backbone of distributed processing. And second, we will continue to look at Figure 1 in order to explore and define the relationships between services.

The concept of modularity is not just known in computer science but also in other sciences, e.g., engineering and management. It is therefore interesting to find that the basic definitions are the same in all disciplines. Modularity can be defined as a way of organizing complex products and processes efficiently [Baldwin & Clark 1997] by splitting complex tasks into simpler ones that can be handled independently [Mikkola & Gassmann 2003]. The resulting parts can be produced and handled separately and it is possible to use them in different environments, either in the same product or even in another project (*reusability*). These parts are the services in a service network. The benefit of modular software design includes a clearer structure, more efficient implementation and upgrades, easier software maintenance and reusability. Each service in a network carries out a certain task. Services can be developed separately and linked into one program. Once the services are linked with each other through so called *interfaces*, they can be implemented and modified independently [Litvin & Litvin 1998].

Now getting back to Figure 1, we see that the middle diagram represents the client-server architecture approach in which application components function either as a *requester* (i.e., client) of information or a *provider* (i.e., server) of information. In the past, this was a rather static approach to distributed processing, allowing each component to function only as a client or a server, but rarely both. In basic two-tier client-server architecture, the client is usually located on the users desktop and the servers, such as a data server (using a DBMS), are located on a more powerful machine that is able to service many clients. The client-server technology has long been a stable architecture for application developers. Indeed, the third (right-most) diagram in Figure 1 can be viewed as an extension of the traditional two-tier client-server architecture. Although as seen in the diagram, this architecture is based on multiple, distributed ‘nodes’ that interact with one another to build applications. In this model, each node can be both a client (i.e., a requester) and a server (i.e., a provider) depending on the configuration and intended uses.

What the third diagram does not show, and what will be described in this paper, is the amount of interdependence between the individual nodes, services, or clients. In a simple application of networked services built upon the principles of distributed processing, each of the individual components can be developed separately, function separately, but only be able to communicate with other components via static interfaces. Such a *tightly-coupled* approach still allows domain scientists to focus on their particular area of expertise, and thus develop the services that best suit their needs. But in the end, the coupling with other services is not done dynamically, but statically for every interface between the services. New services can be added to such a network, but there is a minimum level of coding necessary to make this happen.

**Cartography within a distributed system**

One example of a tightly-coupled system is the Statistical Atlas of the European Union (STATLAS [http://www.statlas.org](http://www.statlas.org)). STATLAS was developed as part of an EU-funded project to provide access to, analysis of, and visualization for Europe-wide statistical and spatial data. Statistical data was supplied by Eurostat’s New Cronos: REGIO database consisting of the main aspects of economic and social life in the European Union. The administrative levels of classification used the first three levels of the Nomenclature of Statistical Territorial Units (NUTS). Spatial data for these three NUTS levels was also provided by Eurostat in the form of the GISCO database. The GISCO and REGIO data sets are designed to function together, but are nonetheless maintained in separate DBMS’s on separate servers.

Along with the databases, the STATLAS project faced the task of integrating theme specific analytical services into the network. For this reason, we opted to follow a three-tiered architectural approach as shown in Figure 2. In this model, each of the three functional domains is classified according to the following functionality:

![Figure 1: Three types of mapping architectures (by M.H. Tsou and B.P. Buttenfield)](image-url)
• User Domain: provides the interface to a user component and interacts with the Mediation Domain. In the case of the STATLAS application, the User Domain consisted of a Graphical User Interface that acted only as a client; requesting information from the Broker and receiving information either from the Broker or the Visualization Environment of the Mediation Domain.

• Mediation Domain: provides the main communication part of the application. It mediates the service calls from the user domain to the thematic domain based on meta-data exchanged with the system components of the thematic domain. The core of the application, the median domain of STATLAS consisted of a Broker, for directing the flow of information; a specification and metadata module, for guaranteeing proper direction of flow as well as up-to-dateness of information; and the visualization environment iMap, which handled the cartographic rendering as well atlas management of STATLAS.

• Thematic domain: incorporates the networked thematic components to be coupled. By means of request and/or retrieval services, these system components are connected to the mediation domain. Within the STATLAS application, there were two main components to the Thematic Domain: a DBMS and a statistical toolbox. The DBMS consisted of two separate databases, one for the statistical data and another for the spatial data described above. The statistical toolbox consisted of a set of three modules for providing statistical analysis: descriptive statistics, data mining, and spatial econometrics. In future implementations of this atlas, this domain could be expanded to include other thematic services, such as GIS, flood analysis, meteorological services, etc.

![Diagram of STATLAS application](image)

**Figure 2. Architectural Model of the STATLAS application.**

**Tightly-Coupled Distributed Services**

There is no doubt that the STATLAS application is an example of a tightly-coupled distributed processing system. Between each component, a certain amount of coding was needed. Tightly-coupled systems are often a product of proprietary technology. While it is not the case that all proprietary technology leads to closed systems, it is often a by-product as a result of increasing the functionality provided in lieu of using international standards (e.g., ISO and OGC) for providing the interfaces between distributed services. A good example of this is the Cartographic Markup Language (CartoML) used in the iMap visualization environment. This map description language is much more advanced than any current styling languages offered by a standard organization, such as the Styled Layer Description (SLD 1.0) supported as an Implementation Specification of the OGC. In this case, the iMap visualization environment is capable of more functionality than is possible using the OGC standard services, i.e., WMS, WCS, and WFS. In order to be able to take full advantage of iMap, a new communication method was needed for passing the appropriate parameters to the mapping engine. Thus, CartoML was born.
CartoML is an approach for managing various map types by packaging cartographic information into an easily readable and understandable XML-based document. Cartographic design inherits from centuries of refinement, resulting in rules regarding the layout of maps and map elements. These rules make it a challenge to design satisfactory cartographic notation that takes into account all cartographic variables. Everything from the correct thickness of a line, the proper dimensions of proportional symbols and which colors work well together, are variables that a map-maker knows beforehand must be satisfied. What STALTAS developed with CartoML is giving common-place (and not-so-common-place) cartographic variables and parameters a unique place to nest within a well-structured document that has a logical and meaningful format. The advantage of using a markup language for this task is that the information is necessarily nested at different levels of complexity and it fits in very well with the tree-branch structure inherent in XML.

An example of placing cartographic variables within a nested tree-branch structure can be seen in Figure 3. In this example, a line feature is given two attributes - width and color. We know that a line feature can have these two attributes because it is specified in the schema (a). Viewed in the CartoML syntax, the values for each of these attributes is easily readable (b). And the final view of this data is the map itself (c).

![Example CartoML description for a line map element](image)

**Figure 3:** Example CartoML description for a line map element

The description of a simple line feature is obviously not beyond an SLD, but iMap and CartoML were designed to go beyond just simple feature drawing. Some of the advanced features include user-defined transparencies, anti-aliasing, bezier curves, texture mapping, and advanced point symbol functions as seen in Figure 4. Besides the rendering of the actual map, iMap functions as an atlas manager for, among other thing setting the projection system, coordinates, grouping similar maps (based on either geographic area or thematic content), as well as layer management (Fig. 5). All these properties are described in CartoML.

![Image on the left is an example of point features drawn with volumetric properties and transparency. The image on the right is an example of drawing point symbols as multivariate bar charts.](image)

**Figure 4:** The image on the left is an example of point features drawn with volumetric properties and transparency. The image on the right is an example of drawing point symbols as multivariate bar charts.
Figure 5: Management hierarchy of the iMap visualization engine as described in CartoML.

A tightly-coupled solution such as is found in STATLAS offers many advantages, particularly in the increased use of proprietary functionality that can really target the intended use of the service. The positioning of the iMap visualization environment into the Mediation Domain was a conscious decision based its intended use. No matter what other services were called, i.e., spatial data server, statistical data server or statistical toolbox; iMap would be used for every update or redraw of the map. Thus it was determined to be a core service of the application. Also, for performance reasons, it was more efficient to have a static interface directly between iMap and the GUI. The only way to do this was to tightly-couple those two services together.

An alternative to a tightly-coupled solution is a loosely-coupled solution. The next section of this paper will cover this alternative in its discussion of open distributed processing.

“MAP ACCESS SERVICE” IN AN OPEN DISTRIBUTED PROCESSING ENVIRONMENT

Open distributed processing

Distributed applications like STATLAS provide a particular service for a user, and users have a static set of services they can access. Open Distributed Processing (ODP) provides a much richer dynamic environment for a user to find and invoke services. Open distributed Processing represents a combination of two concepts - openness and distributed processing - seen by many as a significant aid to gaining an organizational flexibility needed by today’s IT infrastructures (Joyner). The vision of ODP is an information technology that can support cooperation between geographically and organizationally distributed work groups and will ease the use for both users and programmers. They should not be concerned with the nature and means of distribution. Programming and use of such a distributed application should appear exactly the same as if it were not distributed at all. This is one main characteristic of ODP that can be achieved by so called transparencies.

Transparency

In computing and networking, an application that supports different logical actions through the same user or application interface is transparent. For example, the Network File System allows users to access files stored on a remote machine as if they were stored locally through the same file/folder hierarchy. Transparencies can contribute to ease of use and ease of programming in many ways:

- Access transparency - Regardless of how resource access and representation has to be performed on each individual computing entity, the users of a distributed system should always access resources in a single, uniform way.
- Location transparency - Users of a distributed system should not have to be aware of where a resource is physically located.
- Migration transparency - Users should not be aware of whether a resource or computing entity possesses the ability to move to a different physical or logical location.
- Relocation transparency - Should a resource move while in use, this should not be noticeable to the end user.
- Replication transparency - If a resource is replicated among several locations, it should appear to the user as a single resource.
- Concurrent transparency - While multiple users may compete for and share a single resource, this should not be apparent to any of them.
- Failure transparency - Always try to hide any failure and recovery of computing entities and resources.
- Persistence transparency - Whether a resource lies in volatile or permanent memory should make no difference to the user.

Opportunities from ODPs - Openness

The first section of this paper already described the advantages provided by distributed systems. ODPs now bring additional opportunities due to their openness. From the organizational or management point of view we can benefit from openness in two ways. The term evolution defines the ability of an open system that can be maintained even when external requirements change. Unboundedness is a characteristic that describes the absence of limitations to the provision of information and services for the end-user. An end-user is able to have access to information, sources or even formats he probably did not expect when entering the system.
From the technological point of view, open systems can be characterized by the following attributes:

- **Portability** is the capability of implementing programs without any change on different computer systems.
- **Interoperability** is the ability of joint working between interconnected computing systems.
- **Interconnection** is the ability to move information between computing systems using communications.
- **Distributability** extends the interoperability idea to allow processes and information to be provided and migrated automatically to the most convenient point of an interconnected set of computing systems.

**Open architecture within ORCHESTRA**

The IST - 6th framework integrated project ORCHESTRA (Open Architecture and Spatial Data Infrastructure for Risk Management) tries to improve the efficiency in dealing with risks by developing an open service architecture for risk management that is based on de-facto and de-jure standards. In order to realise this vision, the project will:

- Design an open service-oriented architecture for risk management. Special attention will be paid to an integrated service and data approach including both their spatial, temporal and thematic characteristics.
- Develop the software infrastructure for enabling risk management services.
- Develop services that are useful for different thematic risk management applications (for instance forest fires or floods, man-made risks).
- Validate the ORCHESTRA results in a scenario that involves different risks, both natural and man-made, in a cross-border situation.
- Provide software standards for risk management applications, and to provide additional information about these standards in the form of a book.

(Source: “Orchestra fact sheet”, http://www.eu-orchestra.org/)

As mentioned above, ORCHESTRA will, whenever possible, rely on already existing de-facto and de-jure standards. The ORCHESTRA consortium agreed on using the OpenGis Service Architecture standard, provided by the OGC (Open Geospatial Consortium) and ISO (International Organization for Standardization) as base for the Orchestra Architecture (OA). This Standard relies on many other ISO and OGC standards; one of them is the RM-ODP - an ISO Reference Model for Open Distributed Processing that formulates the ideas of the previous section. A reference model for the OA (RM-OA) is currently under development.

**OA requirements on a “Map Access Service”**

The Institute of Cartography is one of 14 institutions involved in the development of ORCHESTRA, and mainly responsible for the service description and implementation of a mapping service for the OA. A Map Access Service is a service that symbolizes geographic “raw data” provided in vector, grid or raster formats, according to cartographic specifications. This service does not provide a human interface, it can be seen as an intermediate service in a service chain in between of a raw data provider on the one side and a geographic viewer on the other side. Apart from the requirements that are defined by the RM-ODP (portable, interoperable, interconnectable, distributable), services within the OA should also feature the several other attributes.

**Generic:**

As displayed in Figure 6, the instantiation of the OA, the ORCHESTRA Application Architecture uses so called thematic services that will rely on the OA services. The thematic services will be developed in different thematic or application domains that require the OA services to be highly generic. In this context a service is generic, if it is independent of the application domain.
Figure 6: Relationship between OA services and thematic services

**Semantically interoperable:**

"Semantic interoperability emphasizes the importance of information inside enterprise networks and focuses on enabling content, data, and information to interoperate with software systems outside of their origin. Information's meaning is the crucial enabler that allows software to interpret the appropriate context, structure, and format in which the information should reside at any given moment and inside any given system. This information ubiquity is the beginning phase of a truly information-driven organization." (Pollock and Hodgson, 2004) Seen from a geo-viewpoint, semantic interoperability:

- is the ability of a user to access, consistently and coherently, similar (though autonomously defined and managed) classes of digital objects and services distributed across heterogeneous repositories.
- provides systems for cross-correlating items of information across multiple sources to solve problems.

**Existing and missing standards for the implementation**

The implementation of the core functionality for a “Map Access Service”, like the symbolization or the actual map rendering, is not the main problem we have to deal with in our future development. Within ODP architecture we have to abandon the use of any proprietary interface and switch to *de-jure* or *de-facto* standards which are available so far. In our case this is a little drawback for CartoML, which we have already developed as an interface for symbolization that fits the needs of cartographic design. On the other hand there are already a huge number of standards available that help us to fulfill the requirements of the RM-ODP like portability or interoperability. Especially the OGC has made an enormous effort in the last years to come up with standards that are accepted and already implemented by numerous GIS and mapping applications. The following list shows the most important ones that will likely be used or act as guidelines for the implementation of a “Map Access Service” within ORCHESTRA:

<table>
<thead>
<tr>
<th>OGC</th>
<th>ISO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Map Service (WMS)</td>
<td>ISO/DIS 19128 - Web Map Service</td>
</tr>
<tr>
<td>Styled Layer Descriptor (SLD)</td>
<td>ISO/FDIS 19117 - Portrayal</td>
</tr>
<tr>
<td>Web Feature Server (WFS)</td>
<td></td>
</tr>
<tr>
<td>Web Coverage Server (WCS)</td>
<td></td>
</tr>
<tr>
<td>Web Map Context Documents (WMC)</td>
<td></td>
</tr>
<tr>
<td>Filter Encoding (Filter)</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: OGC and corresponding ISO standards relevant for a “Map Access Service”
Semantics is a quite new topic in the IT domain that is will be used within ORCHESTRA. Within the GIS world, there are no adequate concepts yet that have been implemented in a specific standard.

**CONCLUSION AND FUTURE OUTLOOK**

Where do we go from here? Well, it is clear that one path of Web- or Internet-based cartography lies in Open Distributed Processing. This is the path that will lead to a treasure trove of on-line mapping services that are developed by cartographers and for use by everybody. By using simple techniques based on modularity – that is breaking things down into smaller, discrete parts – applications can be created dynamically. And if this is done transparently, then the end-user will have no knowledge of what is going on behind-the-scenes.

One key element of this is the use and further development of international standards and specifications that describe the use of interfaces to be used between services. In the GI world, the OGC, as well as ISO/TC 211 and CEN/TC 287 (European Committee for Standardisation) are working towards this end. At the moment, they provide a basic style description for cartographic output as a standard (SLD). Although this description is now static in that it allows the user to describe only basic map elements. The two alternatives for overcoming this deficiency is to either:

1.) support the further development of the SLD portrayal standard in the hopes of adding advance cartographic theory or,
2.) use a proprietary description language such as CartoML.

In either case, a service is needed to access these symbolization descriptions and apply them to geographic information. That is the purpose of the proposed Map Access Service in the Orchestra Architecture. Lessons learned from the development of iMap during the STATLAS project are being used for the current design phase of this new service. Based on the visualization environment iMap developed for STATLAS, the new Map Access Service will be designed by cartographers to provide high-quality map-rendering for use in a distributed environment.

**References**


