Space Applications Institute

Marine Environment Unit

JOINT RESEARCH CENTRE
EUROPEAN COMMISSION

DESIMA Future Developments

Ardy Siegert and Wolfram Schrimpf

SAI-05 Project 'COAST'

2000

EUR 19632 EN

EUR 19632 EN

Printed in Italy

Preface

This report is produced in the frame of the SAI institutional Project COAST (Coastal Monitoring and Management). The Project COAST supports the implementation of the Community Strategy for integrated planning and management of coastal areas and of the new Water Framework Directive, here, in particular, water quality related issues (e.g. eutrophication).

Information on SAI's institutional projects in the context of Framework Program V can be found on the following Internet address:
http://www.sai.jrc.it/home_activities.htm.
Information on COAST related research and development activities are located at http://www.me.sai.jrc.it .

This report constitutes a Deliverable of Task 4300 (Scenario Specification, System Design and Implementation) of COAST.

The document provides a summary of the conclusions and experience made in the frame of the development, implementation and running of the DESIMA Demonstrator of a distributed information system. Since technology is developing extremely fast in these fields some of the tools of the DESIMA demonstration system seem to be already outdated. The present report takes into account relevant new developments. It can be very helpful for the definition and selection of appropriate software tools in the frame of future developments of distributed information systems not only in the context of COAST.

The DESIMA Demonstrator is available for public use under
http://desima.jrc.it.
User-id and password for running the two scenarios (oil spill and sea defence) can be provided by the COAST Project Leader on request.

Wolfram Schrimpf
COAST Project Leader

Contents

1.0 Introduction

This report aims to provide an overview of techniques available for building decision support systems which give access to distributed spatial data sources. The starting point for the discussion is the DEcision Support system for Integrated coastal zone MAnagement (DESIMA) which was developed to demonstrate how decision making could benefit from access to existing data sources.

The DESIMA demonstrator was successful in showing the benefit of the access to and interpretation of distributed data. Therefore it should be developed further into a mature, scalable and robust system.

1.1 General Architecture

To provide Internet access to distributed spatial data sources, the following elements are needed.

- Web browser
- Client side mapping tool
- Web server
- Server side component coordinator
- Mapping tools

Besides these elements the communication protocols used between the distinctive elements are of importance for building a distributed system.

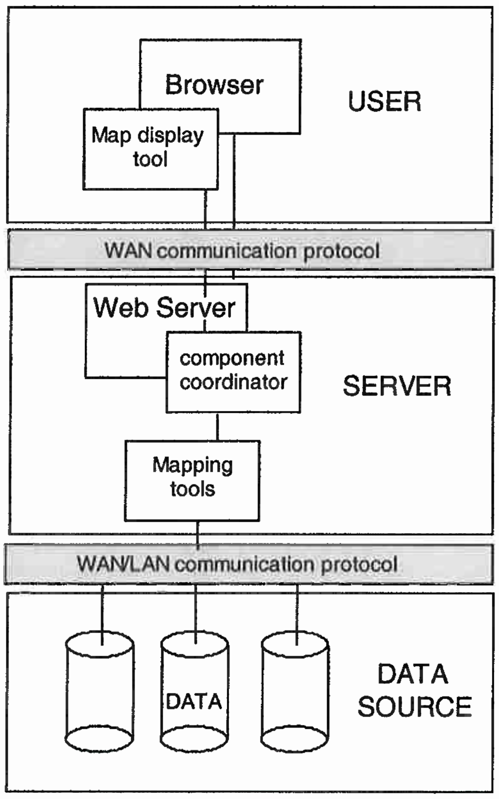

figure 1: elements needed for building internet access to distributed spatial data.

The communication between the User and the Server has a public interface and has different requirements than the communications between the server and the remote data source. The User can only make requests that are built into the user interface and the Web Server can be set up to reject any illegal requests. The relation between the Server and the User is by definition not trusted unless the connection is encrypted.

The data provider needs to trust the parties that access the database, the Server needs to be able to carry out database commands that require an access level above what would be permitted to normal web clients. Therefore a Virtual LAN or an encrypted connection would be most suitable for this connection.

The Server side component coordinator can be in the form of Server modules or some sort of distributed object standard such as CORBA or RMI. The characteristics of each of those possibilities will be described in chapter 3 of this report.

Chapter 4 describes the possibilities available for Mapping solutions. In the last chapters conclusions are drawn from the preceding chapters and recommendations for the finding the best combination of solutions are made.

1.2 Structure of the DESIMA demonstrator

In the current DESIMA demonstrator CORBA is used to provide the services of the component coordinator. The http-server serves the web pages, which provide the supporting information around the demonstrator. As soon as the applet is started the CORBA Object Request Broker (ORB) takes over the communication between the client, the server, and the data. The requests made by the client applet are packed in an Interoperable Object Reference (IOR) and sent from the applet ORB directly to the ORB-daemon on the server. The protocol used for this communication is the Internet Inter ORB Protocol (IIOP). DESIMA connects to the Data-server via IIOP, the data-server makes a connection to the DESIMA server to transfer the answer to the server. At the DESIMA server, the Mapping tool (Multiscope) transforms the data into the right projection, scale, color and data-format before it sends the transformed data to the client over IIOP.

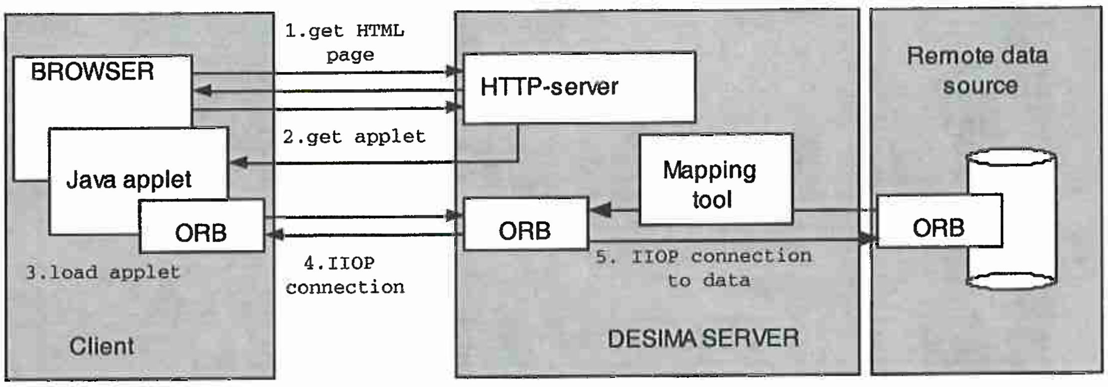

figure 2: Elements in the DESIMA demonstrator

For transfer of large data sets FTP is used instead of IIOP for the connection from the data source to the DESIMA Server.

1.3 Alternative Architecture

The HTTP-server could also be used for the communication between the component coordinator and the client, the information requested by the client would be encoded in the URL. The component coordinator would be an additional module on the HTTP-server for example Java servlets or mod_Perl.

1.4 Improving the DESIMA architecture

To bring DESIMA from demonstrator to a full blown production system the following problems have to be addressed:

- Image processing

The image processing part of DESIMA is currently done in Multiscope. The generic format of the data is first transferred to Multiscope format and then, sometimes after extra operations, Multiscope produces a file suitable for display in the Java applet. This procedure means that a dedicated program has to be written to ingest and process each type of data.

The CZCS data is moved from its database, transformed and stored in a new database. The trajectory it follows to be ingested in DESIMA is too slow to be carried out on line. As most satellite data is stored in binary files with a header file this means that for satellite data the DESIMA system does not fulfil its requirement of keeping data where it is.

- Orbix ORB

The instability of the Orbix ORB was one of the biggest problems tackled in the development of the DESIMA demonstrator. The product contained a lot of bugs. To investigate the possibility of replacement, the TAO and ORBACUS ORBs were tested. TAO is freely available as source code. It was not easy to install and compile it. Once compiled TAO had a rather big footprint (>1GB for the complete package). The overall impression of this distribution was that it was chaotic. The ORBACUS ORB is freely available for non-commercial use. This ORB was easy to install and compile. The files were organized in orderly directories, which made it easy to find the examples. The examples and manual were very clear. Online support on the ORBACUS mailing list was found to be quick and accurate.

The University of Prague has carried out a series of CORBA Comparison tests [7,8]. In which they compare most of the currently used ORBs. In this test the ORBACUS and OMNIORB show packet round times that are between 2 and 30 times quicker than Orbix.

- O_2 database

The O_2 database is not being sold anymore by Ardent Services, standard maintenance is currently still available but no enhancements or patch releases will be made. It is to be expected that this product will be considered obsolete within a year. Therefore it is not recommendable to use this product in a follow up phase.

The use of proprietary elements in the architecture of DESIMA would not be a problem if the API of each component was well defined and described in detail in the technical documentation. Lack of such documentation makes it very hard to replace modules from the original design.

For a decision support system the interpretation and processing of the data is the most important part of the process of making distributed data publicly available. To integrate a data-set into the DESIMA architecture is a time-consuming activity; an interface has to be defined for each type of data and filters have to be written to transform the data to a format that is readable by the GIS system available within DESIMA.

In the time that has passed since the first steps in the DESIMA project were set a lot of developments have taken place in the field of making geographic information available on the web. Commercial companies have made Webmapping tools and the OpenGIS consortium is working on standards, which enable the development of distributed map servers. In adapting the DESIMA architecture, the use of these standards could play an important role.

2.0 Selecting a web server

Web servers serve the files that form Web pages to Web browsers. Every computer on the World Wide Web (WWW) that contains a Web site must have a Web server program.

The main Web servers are Apache, Microsoft-IIS and iPlanets' Netscape-servers. Figure 3 shows the results of surveys carried out by Netcraft [13] since the beginning of the WWW in 1995 until March 2000.

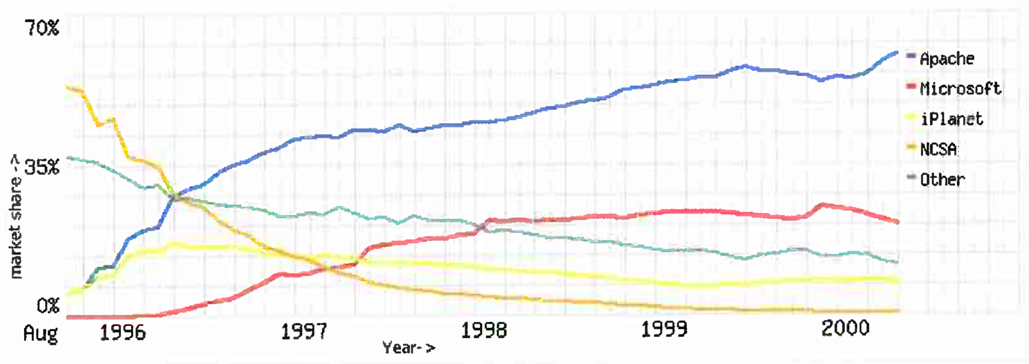

figure 3: Market share for the top web-server across all domains August 95 – March 2000 [2]

table 1: Marketshare of the top developers in March 2000 [2]

Developer	Number of servers counted	Market share %
Apache	7870864	60.05
Microsoft	2742931	20.93
iPlanet	955148	7.29

The Apache and Netscape servers run on a wide variety of computer platforms from Windows to a wide variety of UNIX flavours. Microsoft's Internet Information Server (IIS) runs on with Windows NT and Windows 2000. Most heavy loaded Web servers run Apache on UNIX. This combination was tested to be considerably more stable than IIS on Windows NT [11].

The Apache web server is the most widely used Web server. As it is open source extra modules can be added easily and it is available free of charge. The main arguments for using one of the commercial servers used to be easier configuration and support. Since Apache introduced the Comanche configuration tool and commercial support is available, these arguments have lost most of their force.

3.0 Selecting a Component Coordinator

3.1 Server Modules

The information in the following chapter is based on the book: Writing Apache Modules with Perl and C [1]. The interested reader is referred to this book for a more complete overview of these techniques.

3.1.1 Common Gateway Interface

The Common Gateway Interface (CGI) is a standard way for a Web server to pass a request to an application program. Every time a web server needs CGI the server must set up the CGI environment, read the script into memory and launch the script. The CGI protocol works well with operating systems that were optimized for fast process startup and many simultaneous processes such as Unix, provided that the server doesn't become very heavily loaded. However, as load increases the process creation bottleneck may eventually slow down the scripts. On operating systems that were designed to run lightweight threads and where full processes are rather heavy-weight, such as Windows NT, CGI scripts are showing very bad performance. CGI scripts exit as soon as they finish processing the current request. If the CGI script does some time-consuming operation during startup, such as establishing a database connection or creating complex data structures the overhead of reestablishing the state each time it's needed is considerable.

3.1.2 Web server application programming interfaces (API's)

The functionality of the server can be enhanced, by adding new modules directly to the server executable. Server API's provide access to the internal functions of the server. Disadvantages of this technique are the relatively steep learning curve and the risk and inconvenience that can be caused by a bug in the API module which could crash the whole server. The biggest disadvantage however is the limited portability of server API's

3.1.3 Server-Side includes

Server-side includes can be used to embed, special purpose tags or code inside HTML comments. Examples of advanced implementation of this technique are; Microsoft's Active Server Pages, Allaire Cold Fusion, and PHP. These solutions turn HTML into a miniature programming language complete with variables and database access methods.

Netscape servers recognize HTML pages that have been enhanced with JavaScript code (notice that this is different from client-side JavaScript).

Embperl runs on top of Apaches mod_perl module to fuse HTML with Perl.

The main problem with server-side includes is the absence of standardisation which makes it impossible to transfer pages to another web server without modifications.

3.1.4 Embedded Interpreters

High level interpretive languages can avoid some of the problems of proprietary API's and server-side includes. Embedded interpreters often come with CGI emulation layers allowing script files to be executed directly by the server without the overhead of invoking separate processes. An embedded interpreter also eliminates the need of making dramatic changes to the server software itself. In many cases an embedded interpreter provides a smooth path for speeding up CGI scripts because little or no source code modification is necessary.

Examples of embedded interpreters include mod_perl, which embeds a Perl interpreter, When a Perl script is requested, the latency between loading the script and running it, is dramatically reduced because the interpreter is already in memory. Similar modules exist for Python and TCL.

Sun Microsystems' "sevlet" API provides a standard way for web servers to run small programs written in the Java programming language. Depending on the implementation, a portion of the Java runtime system may be embedded in the web server or the web server itself may be written in Java.

Apaches' servlet system uses co-processes rather than an embedded interpreter. These implementations all avoid the overhead of launching a new external process for each request.

3.1.5 Script Co-processing

To avoid the latency of CGI scripts, they can be kept loaded and running all the time as a co-process. The FastCGI protocol was the first system to use co-processing. Once launched, FastCGI scripts don't exit after finishing a process, but go into an infinite loop to wait for new incoming requests.

3.1.5 Client-Side Scripting

To improve the performance of web-based applications some of the processing can be moved from the server side to the client side. In client-side systems, the browser is an active participant, executing commands and running small programs.

JavaScript and VBScript (Visual Basic script) embed a browser scripting language in HTML documents. When browser scripting is combined with languages with cascading style sheets, document layers, and other HTML enhancements, it results in "Dynamic HTML"(DHTML). The problem with DHTML is the total incompatibility between different browsers. The browsers built by Microsoft and Netscape implement different sets of DHTML features and features vary even between browser version numbers. Developers must choose which browser to support, or use workarounds to support more than one type of browser.

Java applets can be used run client-side applications on a wide range of environments. Improvements in the compatibility on different platforms combined with speed enhancements in the newer versions, have made this a very interesting technique.

Microsoft's ActiveX technology is a repackaging of its Common Object Model (COM) architecture. ActiveX allows dynamic link libraries to be packed up into "controls" shipped across the Internet, and run on the user's computer. Because ActiveX controls are compiled binaries and because COM has not been adopted by other operating systems, this technology is most suitable for uniform Intranet environments that consist of Microsoft-Windows machines running a recent version of Internet Explorer.

3.1.6 Integrated Development Environments

Integrated development environments provide a high –level view of the application. In this type of environment the emphasis is on the application logic and the user interface.

The development environment turns programs into a mixture of database access queries, server-side procedures, and client-side scripts. Some popular environments of this sort include Netscape's "Live" development systems, Apple/NeXT's object-oriented WebObjects, Allaire's ColdFusion and the Microsoft Frontpage publishing system. These systems, although attractive, have the same disadvantage as embedded HTML languages: Once committed to one of these environments, there is no backing out. There is no compatibility between the different vendors' development systems. The scalability of these tools is limited.

3.1.7 Summary

The techniques described above vary in portability, performance, simplicity and power. The 'best' technique to be used is highly dependent on the type of application. Always the choice will be a compromise between different characteristics. Table 1 [1] gives an overview of the strong and weak points of each of the techniques mentioned above.

Table 1: Comparison of Web Development Solutions [1]

	portability	performance	simplicity	power
CGI	+ + + +	+	+ + +	+ +
FastCGI	+ +	+ + +	+ + +	+ +
Server API	+	+ + + +	+	+ + + +
Server-side includes	+ +	+ +	+ + + +	+ +
DHTML	+	+ + +	+	+ +
Client-side Java	+ +	+ + +	+ +	+ + +
Embedded interpreter	+ + +	+ + +	+ +	+ + + +
Integrated system	+	+ + +	+ +	+ + + +

In this table, the "Portability" column indicates how easy it is to move a web application from one server to an other in the case of server-side systems, or from one make of web browser to another in the case of client side solutions. "Performance" means the interactive speed of the application as perceived by the user. "Simplicity" is a combination of the learning curve with the ease of development after the learning process. "Power" is an estimate of the capabilities of the system in terms of control over the application and flexibility to meet creative demands.

Client side Java has a good performance over the whole range of criteria. For a DESIMA application it would have to be combined with a server-side solution. Embedded interpreters have a good overall performance and power rating and scale better than integrated systems. Perl and Java solutions are currently the most widely used solutions. For DESIMA a combination of mod_perl and Java servlets would provide the speed of Perl combined with the Database interfaces, graphics tools and interoperability of Java.

3.2 Distributed object standards

3.2.1 CORBA

The Common Object Request Broker Architecture (CORBA) provides the specifications of a generic framework for building systems with distributed objects.

The Object Request Broker (ORB) forms the core of the specifications. It acts as a message bus between objects which may be located on any machine in a network, implemented in any programming language, and executed on any hardware or operating system platform.

The Interface Definition Language (IDL) is used to define the interface of objects independent of the programming language the objects were implemented in.

The Internet Inter-ORB protocol (IIOP) is a binary protocol for communication between ORBs.

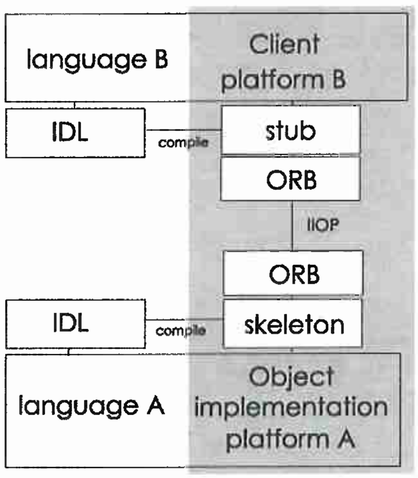

figure 4. Client and server objects can be implemented in different languages the stubs and skeletons take care of the interaction with the ORBs.

In addition to the above, CORBA also includes specifications for services that distributed objects may require, such as naming services and security measures.

3.2.2 Remote Method Invocation (RMI)

The Java Remote Method Invocation (RMI) offers elements for building a distributed object system in Java. It facilitates object function calls between Java Virtual Machines (JVM). The JVMs may be located on separate machines in a network. Remote Method Invocation works only between objects written in Java. Figure 5 shows the RMI architecture layers.

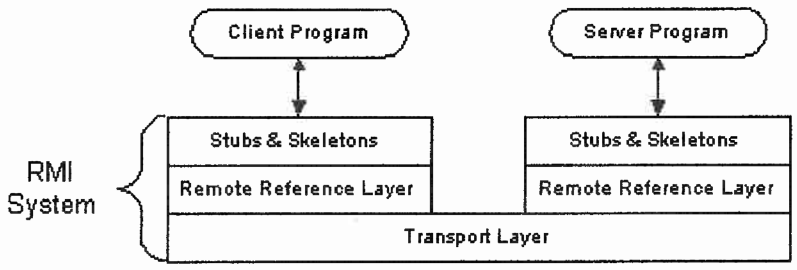

figure 5: RMI architecture layers [9]

The Transport layer is based on TCP/IP. It provides basic connectivity as well as some firewall penetration strategies. The Skeleton understands how to communicate with the stub across the RMI link. The skeleton talks to the stub. It reads the parameters for the method call from the link, makes the call to the remote served implementation object, accepts the return value and then writes the return value back to the stub. The Remote Reference Layer defines and supports the invocation semantics of the RMI connection. RMI can use the Java Remote Method Protocol (JRMP), as well as IIOP and HTTP.

3.2.3 CORBA versus RMI

RMI is easier to learn than CORBA. The CORBA specifications are very extensive delving into the details is sometimes overkill for the task at hand.
CORBA offers more interoperability. RMI is only for JAVA, while CORBA caters for a whole range of implemented languages. With the introduction of RMI-IIOP RMI and CORBA can be easily integrated and thus have become more complementary than competing. When working with legacy services implemented in other languages CORBA may be you best bet. While systems that are build from scratch RMI might be a better choice. Java provides extensive database interfacing and graphics tools.

3.3 Comparison of the protocols used for data transport

HTTP, IIOP and FTP are applications on top of the TCP/IP protocol. To understand the differences and functionality of these application-protocols some more knowledge of the workings of the Internet is needed.

TCP/IP Transmission Control Protocol/ Internet Protocol is the most widely used combination of protocols to send data over the Internet.

3.3.1 IIOP Internet Inter ORB Protocol

IIOP merely specifies how an ORB encodes the TCP/IP addressing information inside an Interoperable Object Reference (IOR) so the client can establish a connection to the server to send a request. The IOR provides the information to uniquely specify an object within a distributed system.

An IOR contains three major pieces of information.

1. Repository ID
2. Endpoint info
3. Object Key

The Repository ID is a pointer to the location of a detailed description of the interface. The Endpoint info contains an IP address and the destination port number. The Object key is used to identify the target object in the server for each request it receives.

3.3.2 HTTP Hyper Text Transfer Protocol

HTTP is an application protocol on top of TCP/IP. The client establishes a connection to the server, issues a request and receives the server's response. The server marks the end of its response by closing the connection. The file returned by the server normally contains hypertext links to other files that can reside on other servers. The client requests are simple ASCII lines and the server's response begins with an ASCII header followed by the data, which can be ASCII or binary. HTTP uses the Uniform Resource Locator (URL) to address a specific document or data object. The URL includes four address elements:

1. protocol
2. host
3. port
4. local entity reference

The text character of the URL makes for easy debugging and maintenance.

3.3.3 FTP File Transfer Protocol

The File Transfer Protocol copies a complete file from one system to another system. FTP uses two TCP connections between the client and the server: a control connection that is left up for the duration of the client-server session and a data connection that is created and deleted as necessary.

3.3.4 JRMP Java Remote Method Protocol

On top of TCP/IP, RMI uses a wire level protocol called Java Remote Method Protocol (JRMP). JRMP is a protocol that uses sockets and streams with a minimum of protocol overhead. RMI can also use IIOP which makes it possible to integrate RMI with CORBA. The third alternative for RMI is using HTTP to get through firewalls.

3.3.5 Comparison

The advantage of using just sockets and streams is that it is very efficient. On the other hand this approach makes it necessary to know the type and format of the data that will be transmitted and received.

HTTP requests are slower than those sent through direct sockets, because it imposes protocol overhead on the data stream. The advantage of HTTP is that it provides a standard means for serving and accessing data objects, and for identification of type and format of these objects.

IIOP also has the disadvantage of protocol overhead and the advantage of taking care of all type and format issues between client and server.

FTP is optimized to maximize reliable throughput and is therefore most useable for bulk transfer. It imposes less protocol overhead than IIOP but also leaves more work for the programmer in defining the data stream.

The URL and the IOR both give a description of host, port and content. They are comparable in functionality and protocol overhead.

Choosing between these protocols is a trade off between bandwith, programming efforts and maintainability. IIOP tends to require a non-trivial amount of run-time support to function properly. HTTP is the dominant Internet protocol, because it is simple, text based and requires very little run-time support to work properly. Additionally, many corporate firewalls block IIOP traffic while allowing HTTP packets into their networks. HTTP-servers are scalable, reliable and easy to administer.

4. 0 Selecting a Mapping solution

4.1 Open GIS Web Mapping Testbed

The Open GIS Consortium has taken the initiative for a Web Mapping Testbed, which aims to produce open geoprocessing Web technologies, capable of:

- Accessing multiple databases with Geographic information
- Fusing this Geographic information on a single web browser display
- Sending commands to multiple Geographic information servers to control rendering processes
- Querying the Geographic Information servers to obtain additional information.
- Updating map display features
- Providing an interface and viewing capability for terrain analysis, environmental conditions, real-time positional data tracking and analysis and terrain visualisation application.

The capabilities described above, fit seamless with the aim of DESIMA. Therefore it is worthwhile to investigate the possibility of using these standards to transform DESIMA into a mature and scaleable system.

The OpenGIS consortium distinguishes four distinct processing stages in the portrayal process between the data storage and the user-display. In figure 6 these four components are displayed.

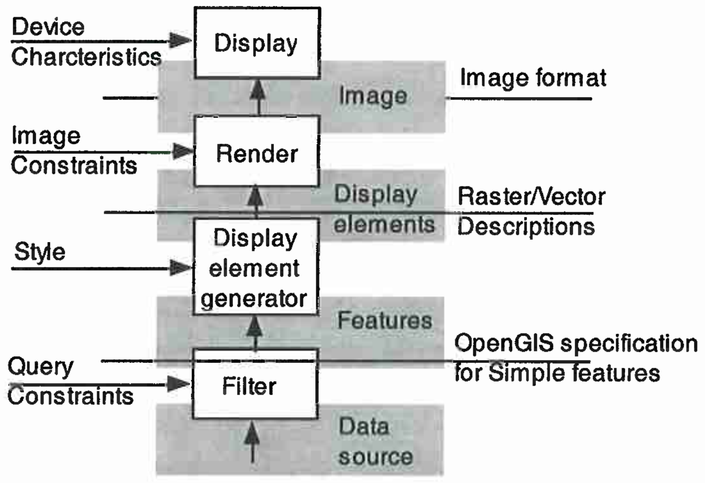

figure 6: The four processing stages between data and display [3]

The four processing stages displayed in figure 6 are:

Filter—this architecture component selects features (and, potentially, image and elevation data) from a data source according to a supplied query constraint and sends them across an interface to the next component. For example, a query might express that all features named "roads" and "streams" within some region of interest to be selected.

Display Element Generator—this architectural component will combine an incoming feature set, converts the features to graphic symbols, and apply a particular style definition. It will then send these display elements on to the next architectural component. For example, the "roads" will form a set of display elements that are solid black lines two pixels wide and two one pixel wide red lines on each side of the black line. It also has leeway to ignore the feature's geometry and substitute something as simple as a dot.

Render—this architectural component will render an image from the display elements according to some image constraints (such as what information is already on the display) and produce an image (in device coordinate space) that is capable of being displayed directly in the client application (such as a browser).

Display—this architectural component displays an image in a client application according to specified device characteristics.
Translated to the architecture of the DESIMA demonstrator the functionality of the Display Element Generator and the Render component are taken care of by Multiscope. While the Filter component is formed by custom written programs. The Display component in the DESIMA demonstrator is formed by Java applets.

Furthermore, web mapping can be defined in terms of what kind of information crosses the boundary between a client computer and a web server. The information stream can be divided into three main information streams:

1. **Pictures** such as maps in the form of JPEG or GIF
2. **Graphic elements** such as roads defined as a polyline, these elements are already in a projected reference system and with already defined symbolisation for geographic features. Some of the graphic elements could themselves be pictures in the sense of a bitmap or pre-drawn fragment of a map, so the graphic element case may also include a picture as a subset.
3. **Data** such as geographic features

Pictures

The map request can be made by means of an URL. The URL contains attributes such as: layers, styles, bounding box etc. An example of a request is given in figure 7.

```
http://www.opengis.org/wmt/myname?WMTver=0.9&REQUEST=map&La
yers=layer1&STYLES=style1&BBOX=10.1,20.5,12.4,23.6&SRS=4326
&WIDTH=400&HEIGHT=300&EXCEPTIONS=INIMAGE&FORMAT=JPEG&TRANSP
ARENT=TRUE&BGCOLOR=0xffffff
```

fig 7:Example URL for a maprequest [3]

In this URL the spatial context is defined in the attributes SRC, BBOX and WIDTH/HEIGHT.

SRC gives the Spatial Reference System based on the European Petroleum Survey Group tables.

BBOX - The Bounding Box: a set of four comma separated numbers to specify the minimum X, minimum Y, Maximum X, MaximumY ranges, expressed in (longitude and latitude) units of the SRS such that a rectangular area is defined.

The WIDTH/HEIGHT parameters in positive integers specify a rectangle of pixels into which the resulting map should be drawn and returned. If the aspect ratio of the BBOX and WIDTH/HEIGHT are different, the MapServer must assume that it is to stretch the returned map so that the resulting pixels could themselves be rendered in the aspect ratio of the BBOX.

Graphic elements and data

The Graphic elements and the data can be encoded in eXtensible Markup Language (XML). XML, like HTML is a subset of the Standard Generalized Mark-up Language (SGML). Both XML and HTML contain markup symbols to describe the contents of a page or file. HTML however describes the content of a Web page only in terms of how it is to be displayed and interacted with. For example <P> starts a new paragraph. XML describes the content in terms of what data is being described. For example, a <PHONENUM> could indicate that the data that followed was a phone number.

XML can be extended; it is a meta-language (a language for describing other languages). Each particular XML Language is defined by its own Document Type Definition (DTD). DTDs describe how the XML document elements, attributes and other data are defined and logically related in a document.A stylesheet dictates how document elements should be formatted when they are displayed. Different stylesheets can be applied to the same document, depending on the environment, thus changing its appearance without affecting any of the underlying data.

Thus in XML there is a distinction between the content and the formatting. This aspect makes it extremely useful for describing Graphics elements and data. The OpenGIS Consortium has developed a simple features DTD [3] defining things as :points, lines and polygons. Feature layers can be defined against this DTD.
The World Wide Web Consortium has written a standard for the definition of Scalable Vector Graphics (SVG) in XML. Furthermore XML provides good implementations for pdf (Portable Document Format) display, meta-data transfer and internationalization.

4.2 Commercial Web mapping tools

The DESIMA audience is formed by decision-makers, they normally have experience with GIS-packages. Therefore the Web mapping solution for DESIMA should not be an extension of existing GIS/desktop mapping packages. The level of expertise, training and involvement required for these packages may be too high for the web user.

The Client side of the web mapping tool should be easy to install. The side part of the web mapping tool should run on UNIX and the client side plug-in, if required, should be free. They should be Operating system independent and work at least on current versions of Internet Explorer and Netscape.

table 2: Web mapping products running on UNIX –servers [6]

Vendor	Product name	Estimated cost (Euro)	plug-in
APIC	APIC\Web	6400	?
ESRI	ArcIMS	8000	R
ESRI	ArcViewIMS	5000	NR
ESRI	MapObjectsIMS	5000	NR
Laser-Scan	Gothic Integrator	24000	O
MapInfo	MapXtreme 2.0	25000	O

R= Required
NR=Not Required
O=optional

table 3: Formats supported by the products of table 2 [6]

product name	Shape	Arc	DXF	DWG	DGN	Tiff	GeoTiff
APIC\Web	-	-	+	-	-	+	-
ArcIMS	+	+	-	+	-	+	+
ARC View IMS	+	+	+	+	+	+	+
MapObjects IMS	+	+	+	+	+	+	+
Gothic Integrator	+	+	+	-	+	+	+
MapXtreme 2.0	-	-	-	+	-	+	+

These products all support panning and zooming and database interaction. There are many more Web mapping products on the market but most of these only run on Windows-servers. The DESIMA application needs to be able to support multiple users and multiple threads, UNIX based operating systems are much more suitable for these type of applications, therefore the windows based web mapping tools were not taken into account in this overview.

Other interesting Commercial products with a more complete package for web access to distributed databases are Caris Spatial Fusion and CubeWerx solutions for Spatial Data Warehousing.

Caris Spatial Fusion is based on integration of the Orbix ORB with Mapping tools and databases and therefore closely follows the current DESIMA architecture.

CubeWerx uses the Open GIS approach combined with CGI and Javascript to build similar applications. It is part of the Open GIS Web Mapping Testbed. The CubeView on-line demonstration [12] is quick and stable. It works well on both Internet Explorer and Netscape and was used on Windows Macintosh and Unix platforms without problems.

4.3 Non-commercial web mapping products

The University of Minesota has developed a non-commercial MapServer that runs on Unix and Windows NT. It uses CGI to fetch maps from distributed sources. The programs are written in C, a Perl scripting tool; MapScript is provided to hide the complexity of the C API http://mapserver.gis.umn.edu/.

USGS has a wide range of tools available that can be used for integration into spatial data warehousing solutions.

Advantage of open source products is the accessibility of the source code to fix bugs if necessary. There are roughly three types of open source licenses:

- For truly "Public Domain" work all rights are abandoned and the material is offered without restrictions.
- The GNU Public License and licenses modeled on it impose the restriction that source code must be distributed or made available for all works that are derivatives of the GNU copyrighted code. Therefore products with this license cannot be mixed with commercial products.
- The Berkeley License allows proprietary adjustments to the code as long as the derived product acknowledges the writers of the included code.

It is important to read the licenses carefully to be aware of the conditions under which source code can be used, or adjusted.

5.0 Conclusions

The web mapping and distributed computing technology has advanced considerably since DESIMA was first designed. Due to this change, new products and standards were developed to support the development of systems like DESIMA. The development of these products and standard has benefited greatly from pilot projects like DESIMA. The web mapping tools available today bundle the experience of countless early projects.

CORBA seems more suitable for Intranet applications than for use over the internet. In the development of the DESIMA demonstrator often ftp was preferred over IIOP for sending bulk data from remote servers to the DESIMA server. IIOP often caused problems with firewalls. The interoperability between ORBs from different vendors is not very good. In contrast with the aim of CORBA the use of most CORBA related products is still quite complicated. If something breaks it is often due to bugs in the code. If there is only a binary distribution available, the programmer is dependent on support from the company that delivered the ORB.

Although CORBA looks like a very good solution for a lot of problems, the practical implementations of this protocol are not without problems. The ORBs available on the market still have a considerable learning curve, ask for maintenance, and IIOP traffic is quite often blocked by firewalls. The interoperability between ORBs of different vendors is still very problematic.

6.0 Recommendations

The development of DESIMA into a mature, scalable and robust product would require quite some adjustments on the current demonstrator. By taking advantage of newly developed techniques, the DESIMA system could be completely rebuilt with the same effort it would take to expand the current demonstrator. The system built with the new technologies would be more flexible than the expanded version of the current demonstrator.

For the Communication between User and Server the HTTP protocol currently provides the best compromise between bandwidth, reliability and maintenance. HTTP is more demanding on bandwidth than other protocols but it is simple, text based and requires very little run-time support to work properly. Additionally, many corporate firewalls block IIOP traffic while allowing HTTP packets.

The communication between the DESIMA server and the remote data sources should be determined by the protocol defined by the remote database. In the current DESIMA demonstrator, FTP connections were often preferred over IIOP.

The following combination seems to be the best solution to the DESIMA design challenge:

- Client side Java, Javascript or DHTML
- HTTP between user and DESIMA server
- Apache server
- mod_Perl and Java servlets
- OpenGIS - XML solution for Mapping
- A secure connection over a protocol the database supports, between the DESIMA server and the remote data sources.

Implementation of the OpenGIS specifications seems to be the best guideline for success. Engineering trade-offs are at the heart of making decisions in any technology. The simple solutions are often the best because they are easiest to maintain, repair and adjust.

Literature

[1] Writing Apache Modules with Perl and C, Stein and MacEachern, O'Reilly, USA, 1999

[2] The Netcraft Web server Survey, http://www.netcraft.com/Survey/, March 1999

[3] Web Map Server Interface Specification, Allan Doyle, OpenGIS project document 99-077r1, Open GIS Consortium, December 1999

[4] A Request for Technology In Support of a Web Mapping Technology Testbed, Open GIS Consortium, October 1998

[5] TCP/IP Illustrated Volume 1; The protocols, W. Richard Stevens, Addison-Wesley Professional Computing Series Reading MA, USA, 1994

[6] Don't hit Warp Speed with the wrong equipment, Frederick Limp, GEOEurope, December 1999

[7] CORBA Comparison Project Final Project Report, http://nenya.ms.mff.cuni.cz, Distributed Systems Research Group Department of Software Engineering Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic, 1998

[8] CORBA Comparison Project, Project Extension Final Report, http://nenya.ms.mff.cuni.cz, Distributed Systems Research Group Department of Software Engineering Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic, 1999

[9] Fundamentals of RMI, Java Developer Connection , SUN, http://developer.java.sun.com/developer/onlineTraining/rmi/RMI.html, 2000

[10] Java Distributed Computing, Jim Farley, O'Reilly, January 1998

[11] To Be Up or Not To Be Up, Juergen Schmidt, c't 08/2000 page 174, http://www.heise.de/ct/english/00/08/174/, 2000

[12] http://www.cubewerx.com/demo/cubeview/, 2000

[13] http://www.netcraft.com

List of Acronyms

API	Application Programming Interface
BBOX	Bounding Box
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
DESIMA	DEcision Support for Integrated coastal zone MAnagement
DHTML	Dynamic Hyper Text Mark-up Language
DTD	Document Type Definition
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
GIS	Geographic Information System
HTML	Hyper Text Mark-up Language
HTTP	Hyper Text Transfer Protocol
ID	Identifier
IDL	Interface Definition Language
IIOP	Internet Inter ORB protocol
IIS	Internet Information Server
IOR	Interoperable Object Reference
IP	Internet Protocol
JRMP	Java Remote Method Protocol
JVM	Java Virtual Machine
LAN	Local Area Network
ORB	Object Request Broker
RMI	Remote Method Invocation
SGML	Standard Generalized Mark-up Language
SRC	Spatial Reference System
TCL	Tool Command Language
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
VB	Visual Basic
WAN	Wide Area Network
WWW	World Wide Web
XML	eXtensible Mark-up Language