Development of a Distributed System on an Internet/Intranet Environment and Its Application to Environmental Multi-source Data Integration

MARGARETH SIMÕES P. MEIRELLES, D.Sc.
Department of System Engineering
Rio de Janeiro State University and
National Soil Research Center, Embrapa Solos
Rio de Janeiro, RJ
BRAZIL
Tel: +55 (21) 587-7442

OSCAR FARIAS, D.Sc.
Department of System Engineering
Rio de Janeiro State University
Rua São Francisco Xavier, 524, bl. D, 5018
Maracanã, 20559-900, Rio de Janeiro, RJ
BRAZIL
Tel: +55 (21) 587-7442

Abtract: - This paper relates our experience in developing DISI^{2}E - a distributed system on an Internet/Intranet environment; built from modules the same way children play LEGO. These modules can be any kind of file visualized by a browser: pictures, html files, audio files, animations files and other types of files visualized via a plug-in, like executable programs (CGI), applets and queries to databases. The user through an interface very similar to the Windows Explorer selects these modules. Then, these modules, following the object-oriented paradigm, are related basically in part-of relations (containment relationships, like regions on a map) or some of them – simpler – are taken as attributes of more complex ones. All the user needs to operate DISI^{2}E are *http (hyper text transfer protocol) servers* – to publish the distributed information – and *browsers*, enabling users to navigate through the distributed system visualizing non-structured information, querying databases and executing programs.

Keywords: - Distributed information system, hypertext, object associations, environmental, database

1. Introduction

DISI^{2}E is a Distributed Information System on an Internet/Intranet Environment originated from a research grant between CEPEL (Electrical Energy Research Center) and UERJ (Rio de Janeiro State University). The goal was to build an Internet based distributed system with a graphical user interface based on hierarchical maps, associating information and services to a given topology. In that way, CEPEL, the largest Brazilian research center on electrical energy could, for example, build geographical databases with equipments (e.g., transformers) of the energy suppliers and query these databases to answer questions directly related with these equipments, like specification data, operational data, administrative data and so on. Besides that, it would be interesting, for example, to have animations showing the technicians how to maintain or even how to assemble - or disassemble - a given equipment. Formularies to be filled and sent to the person associated to administrative tasks, like how to order equipment maintenance, were also desired. Some users also wanted to test a given equipment by executing some program. At the same time, the system could be used to maintain a database related to land use and the geographical disposition of the electrical network, and so forth.

The system requirements could be thought, in a broader sense, as the possibility of aggregate generic isolated objects that describe the resources of a company and to relate these objects in a logical manner. Most of the time these are part-of or containment relationships [2,4]. If we could allow the users to establish arbitrary associations between objects we could solve our problem and, more than this, we would have a system with the possibility of being applied in several distinct areas: commercial, educational (e.g., geography classes), etc. Another important system requirement is that we need a distributed system on an Internet/Intranet environment, i.e., the objects could, at principle, be located at any machine on the Internet address space.

2. Logical Solution For Disi^{2}e

We have envisaged a solution very similar to a child's toy called LEGO. With LEGO blocks children build complex objects based on simple ones represented by some elementary blocks. Simple objects are associated on each other, to build more complex ones. Similarly, we could think of the resources of a company, represented by objects. These objects can be regarded as a set of files of different types, describing physical resources, personal resources, procedures, methodologies, and so on. All we have to do is connect these objects in a logical way to have a model of the company. The hypertext technology gives us the possibility to connect these objects together and to establish relations between them in a very general way. Our solution links the objects using hypertext references, but in a more limited way. Basically, we have two kinds of relations between objects: i) a containment relationship, like the ones that we see in maps, where, for example, a country encompasses several states. This can be a recursive (finite) relation, since each state by its turn, encompasses several districts and municipalities; ii) a kind of attribute relationship, where a certain object can have associated descriptors, that give additional information related to the object. For instance, equipments, like a transformer, can have several descriptors (files) associated with them. One of these descriptors could be an animation showing how to perform a maintenance routine; another descriptor could query a database to show the operational data related to that transformer; a third descriptor could execute a program, that, based on operational data recovered from a database, can diagnose the transformer; a fourth one could be a file with a technical description of the transformer, etc. With these two kinds of relations we can, for example, distribute the facilities and equipments of a company in maps representing regions and, then, describe these facilities and equipments based in the descriptors.

Our solution also needed to allow some kind of navigation between these maps, and to allow the user to inspect the object descriptors. Then we have represented the objects in a containment relationship as image maps objects, a well-known technique from the World Wide Web (WWW), so that we can use an http browser, like Internet Explorer or Netscape Navigator, to go from one map to another. The descriptors were represented as hyperlinks in the object being browsed, that, when pressed, they call another object describing, or related to, the first one. Descriptors can be any kind of file visualized by a browser: pictures, html files [3], audio files, animation files and even other types of files visualized via a plug-in (e.g., CGM - Computer Graphics Metafile), like executable programs (CGI), applets, and database queries. The interaction with databases is found in the descriptor. So it can be embedded, via SQL statements, in an applet referenced by a html file, or, in a html file that communicates directly with a http server that hosts PHP. In the former case the database queries are accomplished through JDBC (Java Data Base Connectivity) and, in the latter, through embedded SQL statements in html files.

3. Implementation Of Disi^{2}e

We have developed a software tool called "Developer View", which is somewhat similar, in pictorial terms, to Windows Explorer. The Developer View allows the user to associate the objects - html files, sound files, image files, animation files, SQL queries, CGM files, CGI programs, and other file formats directly interpreted by an http browser - that, together, represent a distributed system. These objects can be located in any directory of the computer system. The associations between these objects are built through a graphical interface developed in C++ Builder [1]. They are restricted to the two types of relations mentioned above: containment and attribute. The net result of all these associations is a web site, written in html. All files that compose the web site: the original objects given by the user, plus the association between these objects - reflected in the html files - are put in an htdocs directory of an http server. Finally, the user points his browser to the root object of the htdocs directory on the http server and begins his navigation through a true distributed system on an Internet/Intranet environment.

When creating the site, the Developer View automatically builds all the directory structure needed, obeying the paths defined by the user. Some very useful functions help the developer task. They are: *i) Autosave* - when enabled, the Developer View saves the work done by the developer in building the site at previously defined time intervals; *ii) Lock/Unlock objects* - allows the developer to maintain some control over the objects that are already set to be published and the objects still in development. Locked objects cannot be edited; *iii) Dependency Verification* - sweep all the structure of the site that is being

generated, seeking for duplicated objects, which are not allowed, and for invalid associations. All the inconsistencies found are registered in a log.

The Developer View permits the fast development of sites, and also reduces the effort to maintain sites, since it comprises site edition.

4. Tools Needed By The User

All the user needs to operate DISI^{2}E is *http (hyper text transfer protocol) servers* – to disseminate the distributed information – and *browsers*, so that people can navigate through the distributed system (built by the user at his will) visualizing non-structured information, querying databases and executing programs. Of course, the user needs to have at hand all the objects that he wants to take part on his distributed system and also the resources to make these objects functional. So, if he wants to access a database and submit queries, he will need a Data Base Management System (DBMS), a database and a mechanism to query the database (e.g., an applet using JDBC). The same is true for objects of other types, like CGM animations, sound files, etc.

5. Some Applications Of Disi^{2}e

The DISI^{2}E system can be applied on several contexts. A company can use it to disseminate to its clients and employees information related to its operational infrastructure, for example geographical location of equipment and other resources, like photos and animations describing the right way of repairing or maintaining these equipments, technical and operational data of equipment, etc. In another context DISI^{2}E can be used to teach Geography, relating written information, sounds (folk songs), animations (typical dances) to points on a map. And the user (teacher) is entirely free to include all the information he wants in the system, like economical and resource information.

Also, a heterogeneous data generated into a geographical database could be integrated and queries to answer questions directly related with the environmental aspects could be done, like bio-physical data (soil, geology, geomorphology, vegetation data), social-economical data, erosion data, land use/ land cover data and so on. In a land degradation diagnoses or monitoring process, it would be interesting, for example, to show different satellites images or even how to apply conservation techniques in agriculture field - or access maps in different scales of detail, going from the watershed to a micro-watershed located in one critical municipality. Formularies containing social-economical aspects of an area (educational level of the owner, type of conservation practices used, number of cattles, and so on), to be filled and sent to the person associated to administrative tasks, could also be desired. Some users also may want to have information about a neighbour municipality. At the same time, the system could be used to maintain a database related to land use and geographical information dynamically updated by satellite images and having many different cartographic scale and projections. Figure 1 presents an example of application of DISI^{2}E system on an Environmental (Erosion Monitoring) Problem involving heterogeneous and distributed information at Taquari watershed in Mato Grosso do Sul, Brasil.

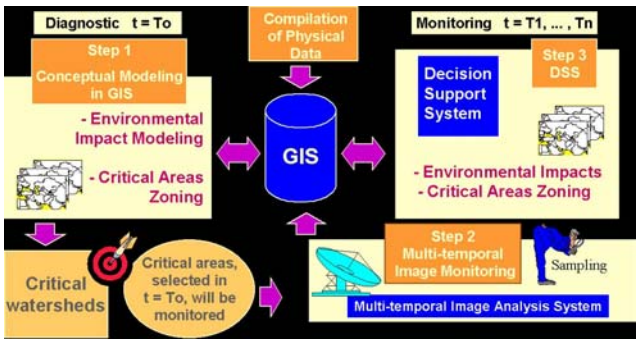

Figure 1 – Example of an Application of DISI^{2}E system on a Distributed Heterougeneous Environmental Database

6. Conclusions

Based in DISI^{2}E a user can build a distributed system in an incremental way, adding to the systems all the objects that can be normally visualized by a browser. New types of files, not yet invented, will be naturally added to the system, since we hope that the browsers will be capable of dealing with them. The system implicitly uses all the Internet standards, what gives it great flexibility. It is a true multi-platform distributed system, and is prepared to cope with future progress on the Internet.

References

[1] Charlie Calvert, et al, *C++ Builder 3 Unleashed*, SAMS Borland Press, 1998.

[2] Grady Booch, *Object Oriented Design with Applications*. The Benjamin/Cummings Publishing Company, Inc, 1991.

[3] HTML 4.0 Specification, W3C Recommendation, revised on 24-Apr-1998.

[4] Robert C. Martin, *Designing Object Oriented C++ Applications Using The Booch Method*. Prentice Hall, Englewood Cliffs, New Jersey, 07632, 1995.