The Web ToolBox Architecture : www.meteo.fr

Sylvie Thépaut, Yann Génin, Michel Edwell
Météo-France – Toulouse

1. The context of work:

Internet is a challenge. The keyword is reactivity. The internet team has to try to reach two requirements :

- To present all the production in a modern way on the Web.
- To offer a robust and secure system.

It looks quite impossible to say now what the end-users requirements will be in 6 months time, thus the architecture design should be generic enough to enable the plug-in of new functionnalities. We can identify several levels of services:

- The end-user is accessing the Web Server, he is looking for very high added value products and will download simple Html Pages from our server.
- The end-user is interested in a more specific product : for example an aircraft pilot looking for Wind forecast. We then need a dynamical service: the user builds up a request and the data are delivered to him in an appropriate format.
- The End-user is a professional looking for raw data through an extranet service.
- The data can be downloaded from a Web Server, or from a ftp server.
- The data can be sent by mail.
- The data can be produced upon request or on a regular basis.

The architecture should enable the widest range of applications scenarios and offers the flexibility to add some new ones. The architecture should take advantage of all the existing applications and add some value to them. And this conclusion leads us to design a 3-tiers architecture.

2. A 3-tiers architecture:

A 3-tiers architecture is based on the idea of components. The components communicate through a middleware. There are three main kinds of components:

- The presentation components which present the information to an external source and obtain input from that source: A web browser, a thin client application.
- The business components which contain the logic of the application.
- The data components which contain the logic to access the data.

The component behaves like a good object: It offers a service to the other components that only see its interface and do not care about its implementation. It should not do too many things, because it will be too difficult to re-use in another context and it should not be too small, otherwise a simple action will require a call to many components and become too complex. Here we face the challenge: find the right granularity for a component.

3. The use of Java/Corba:

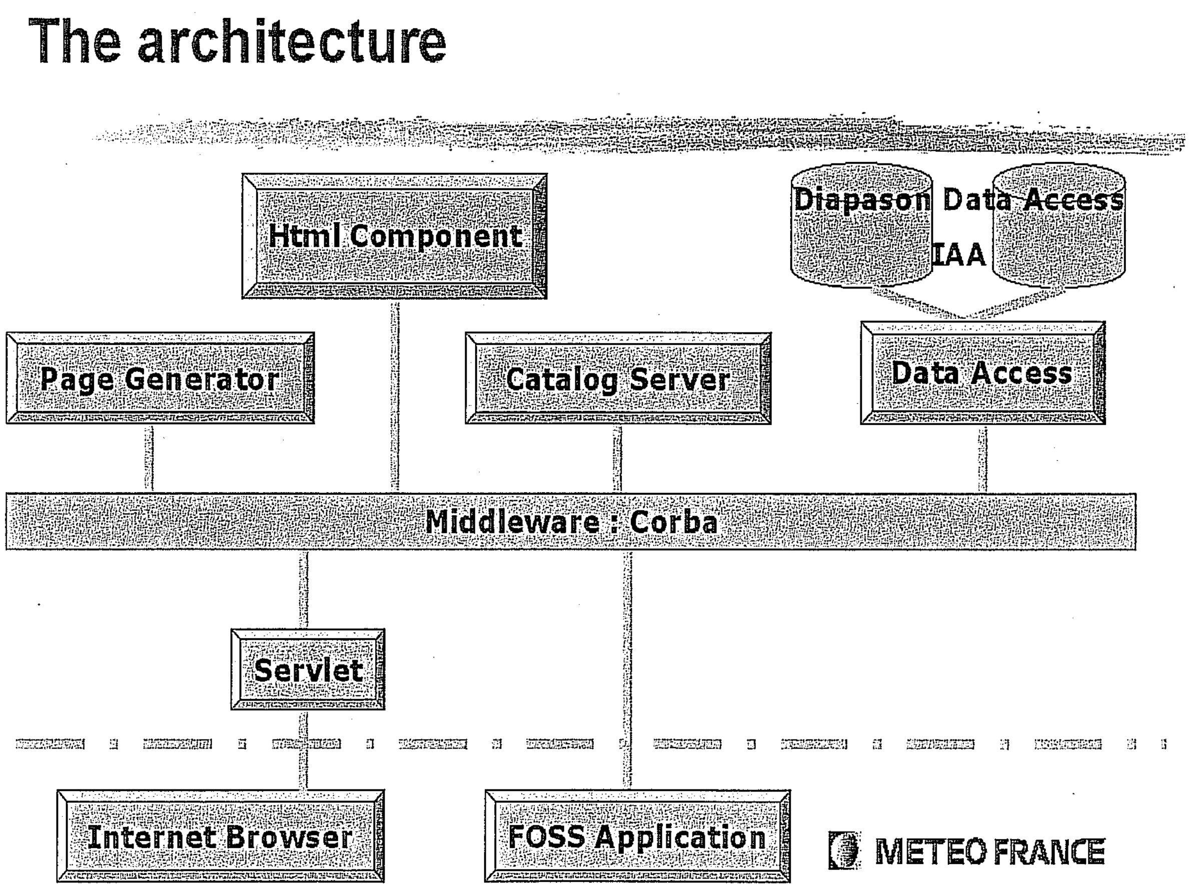
For the implementation, we decided to use an object oriented language well-known in the Web community: Java. As a middleware, we use Corba that enables our components to be distributed. Corba offers some basic load balancing and some fault tolerance. These choices appear to be good. Java is very useful in such a context.

4. A set of robust tools:

One main decision was to take advantage of all the existing applications, and to use a set of robust and well known tools:

- IAA: A Météo-France software to access our operational databases. This software is based on corba and the Object Oriented Technology.
- Metview : An ECMWF software to manipulate and plot some data. The output format is PDF.
- SMS/XCDP: An ECMWF software to handle our automatic production.
- Patrol : a BMC software for the supervision.

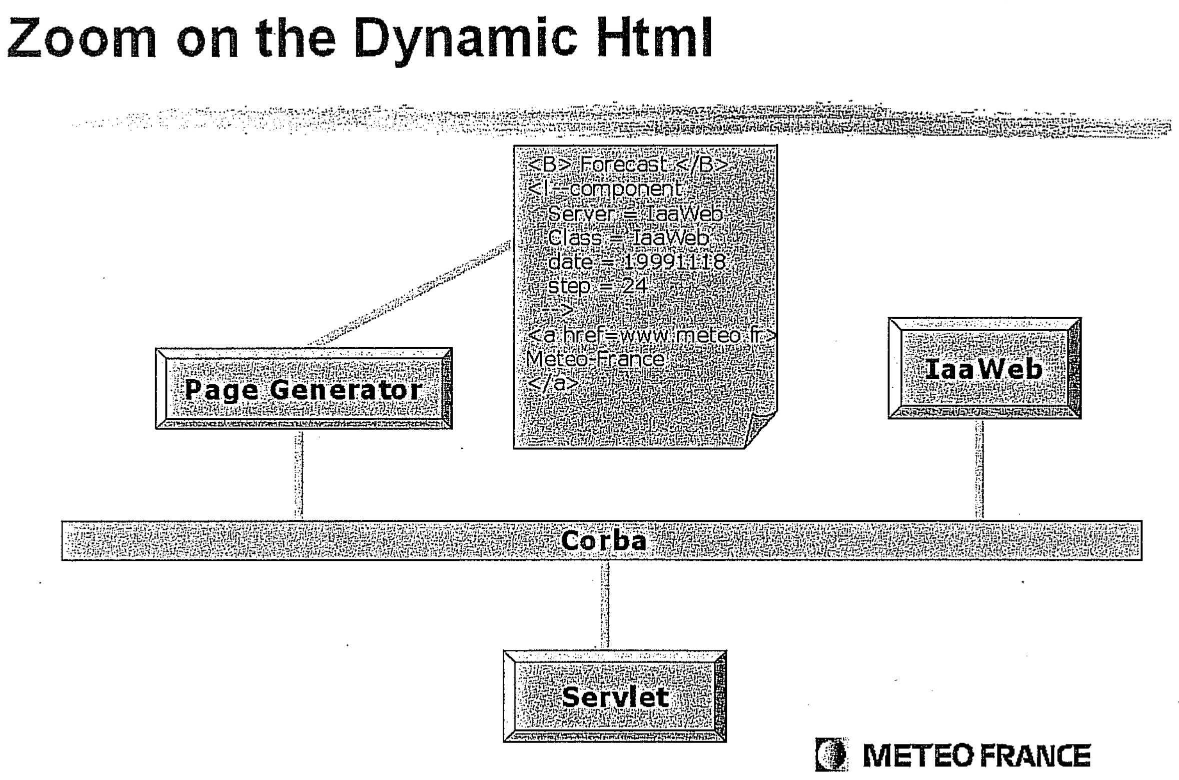
5. The architecture :



The backbone is corba. All components are plugged to it.
An end-user behind his Web-Browser has access using the HTTP protocol to a generic servlet. This servlet receives the request and asks the Page Generator to treat it. In the 3-tiers Architecture model, the servlet is the presentation component, and the Page Generator the

Business component. The Page Generator prepares the output by calling several other business or data components. When ready the output is sent to the client by the servlet.
An end-user using our extranet service FOSS[1], is immediately using the IIOP[2] Protocol to communicate with the other components. The Foss application, a presentation component, presents the user his own catalog. This catalog is downloaded from the Catalog Server. The application enables the user to retrieve data and save them on his machine.

6. Zoom on the dynamical Html :



```
<B> Forecast </B>
<!--component
   Server = IaaWeb
   Class = IaaWeb
   date = 19991118
   step = 24
 -->
<a href=www.meteo.fr>
Meteo-France
</a>
```

The main actors in the creation of dynamical Html are a generic servlet on our Web Server and the Page Generator. The servlet receives a request from an end-user and invokes the Page generator to treat it. The request references what we call an HTML Template. The template is written in pseudo HTML. It contains some normal tags written as such in the HTML output and some special tags that are interpreted by the Page Generator. In this example, The Page Generator calls the IaaWeb Server and asks for the execution of an IaaWeb component. The IaaWeb component gets some forecasts data through a Data Access Component and presents them in an Html Format. The Page Generator replaces the tags by the output. The page Generator is acting as a builder tool, it can invoke nested components and handle some exceptions.
The IaaWeb server is behaving like a factory of IaaWeb Components. The life time of a component is very short : it is created upon request of another component by its factory. The

[1] FOSS : French One Stop Shop.
[2] Corba Protocol.

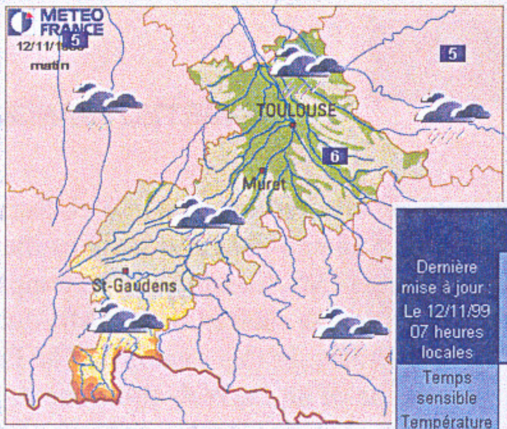
client component gets a reference to it and calls its execute method. When the execution ends, the client will acknowledge, and the component will be deleted.
The difficulties but also the success of such an architecture is the ability to add safely new components.

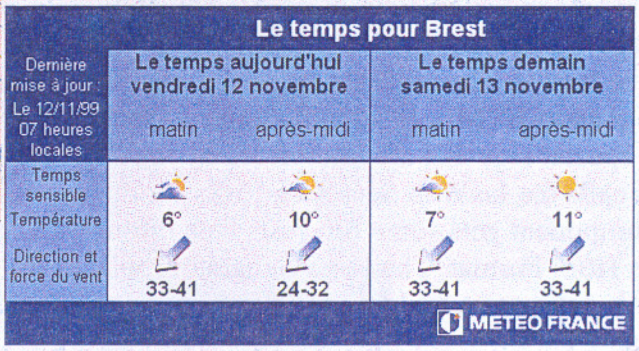
- The servlet was designed generic enough that we do not need to modify it or add a new one when we add new components.
- To keep the system homogeneous, we decided to keep a very simple and generic skeleton for a component. The component receives a request, executes it and makes the result available for downloading.
- The underlying technology is encapsulated, so a developer does not need to be a Corba expert before writing his first component.
- The Object Oriented technology allows our components to inherit from general functionality like fault tolerance in the corba package, messages and information for supervision.

7. Some examples of components:

A 3-tiers architecture allows a perfect splitting between the data and their representation. In this example a single Data Access component (Here, SympoWeb: Access component to the Symposium database) is combined with 2 different representation components. The fist one will present the data as pictograms on a map, and the second one as a weather time evolution in simple Html format.

Example of components





Le temps pour Brest				
Dernière mise à jour : Le 12/11/99 07 heures locales	Le temps aujourd'hui vendredi 12 novembre		Le temps demain samedi 13 novembre	
	matin	après-midi	matin	après-midi
Temps sensible				
Température	6°	10°	7°	11°
Direction et force du vent	33-41	24-32	33-41	33-41

METEO FRANCE

METEO FRANCE

8. The Future:

The architecture is now up and running.
The next challenge will be to maintain this site. It will be useful to add a graphical templates editor. An administrator could then be able to see the list of the existing components. With the mouse he could create his own product, and put it into production.
We have also to think about "clever" navigation. We could dynamically modify the services offered to a user depending on his previous navigation. We already tried some software[3] that could be used in such a concept.
We also started some tests on the Wap[4] technology, some Wap components have been successfully plugged to the architecture.
Right now, we are implementing 3 different sites : the idea is to present the same data with 3 different representations. The production of these 3 sites puts emphasis on the benefits we get from the architecture, and from the reuse of already existing components.

[3] Jrules –ILOG
[4] Wireless Application Protocol