OKAPI project : a climatological production environment

Maryvonne KERDONCUFF [1], Nicolas FONROSE [2]

[1] Météo-France, Climatology department
42 av. G. Coriolis
31057 Toulouse Cedex
France

maryvonne.kerdoncuff@meteo.fr

[2] Valtech, Architecture unit
5 Av Marcel Dassault
31500 Toulouse
France

nicolas.fonrose@valtech.fr

Résumé – Avec ce projet Météo-France vise à mettre en place une plate-forme de production climatologique et différents types de services 'client' :

- services pour les usagers internes de Météo-France
- services pour les usagers des secteurs Recherche et Education
- services pour les usagers commerciaux.

Le système Okapi permet d'accéder aux différentes bases de données climatologiques (données d'observation, données radar, sorties de modèles de prévision numérique, …) et d'obtenir toutes sortes de produits climatologiques élaborés comme des roses de vent, des produits statistiques, des cartes, …
Au début du projet, l'objectif principal était de disposer d'un environnement de production évolutif qui soit capable d'intégrer les nouveaux besoins : accès à de nouvelle bases de données, élaboration de nouveaux produits, …et de faciliter l'accès aux utilisateurs par Internet. Le projet a ainsi choisi les technologies dites de 'l'Internet' comme Java, Corba, HTML, XML. Le système est extensible, il permet d'intégrer des logiciels de traitement existants (fortran, C) ou bien encore des outils commerciaux comme les SIG (Système d'Information Géographique).

Abstract - With this project Meteo-France intends to set up a climatological production platform and different types of customers services :

- services for Meteo-France internal users
- services for Research and Education users
- services for commercial customers.

Okapi allows access to different climatological data bases (observation data, radar data, numerical models output, …) and to all kinds of climatological elaborated products such as wind rose, statistical products, maps etc. …
From the beginning of the project the main objective has been to create an evolvable production environment which would be able to integrate new requirements : access to new data bases, new products etc …, and which would be easily accessible to customers via the Internet. The project has thus chosen 'internet' technologies, some of them being Java, Corba, HTML and XML.
Okapi is based on a multi-tiered architecture, it privileges re-usability with component technology. The Okapi system allows extensibility with the integration of existing processing software (Fortran, C) or commercial tools (GIS).

1. Okapi project objectives

The main objective of the Okapi project is to give easy access to Meteo-France data and products to different kind of users :

- Meteo-France internal users,
- commercial customers,
- institutional users (Research and Training sectors).

The other objectives are to have a system able to easily evolve, and to be open towards other external systems.

2. Prospective study

The project started in 1999 with a prospective study whose aims were to :
- help for the redaction of ‘requirements document’,
- study for different available logical architectures,
- define a strategy to move from existent to future environment,
- development planning,
- a beginning of business analyse.

Moreover the prospective study enlightened that production is always based on the same concepts :
- metadata and data bases access
- different levels of production access
- implementation of ‘business’ processing
- data and products layout
- customer and production management
- production automation

The above concepts are available for any kind of production, so we will see that Okapi system is susceptible to be used for other types of production than meteorological production.

Concerning the technical aspects :
- we pointed out the difficulties for deployment of existent applications (‘heavy weight’ client/server architecture)
- we decided to privilege re-use (components technology) and portability.

3. Towards a new system

3.1. Functionalities

The new system had to be able to :

- access all meteorological metadata and data bases :
  . standard meteorological observations
  . radar/satellite data
  . numerical models output
  . forecasted data
  . lightning data
  . etc
- implement any kind of ‘business’ processing :
  . statistical processing
  . physical processing : ETP Calculations, hydrological evaluation, …
- implement any kind of layout :
  . simple ASCII or binary format
  . tables
  . graphical : curves, cartography, …

- automate the product elaboration,
- allow production management : identification of the user, accounting of the products.

The system must also offer different kinds of ‘client’ applications which allow to :

- select ‘standard’ products via a catalog,
- fill product parameters : the parameters can be different from a product to another one
  . temporal reference for products
  . geographical reference for the products with more or less sophisticated criteria according to the product.
  . complementary parameters necessary to elaborate the product,
- order for the products,
- get product price if necessary,
- get back the products in the user local environment,
- visualize the products,
- generate new products from basic ‘components’ (as Lego system).

3.2. Architecture and technologies

There are different levels in the system architecture ; we will distinguish high level architecture and lower level architecture.

3.2.1. High level architecture

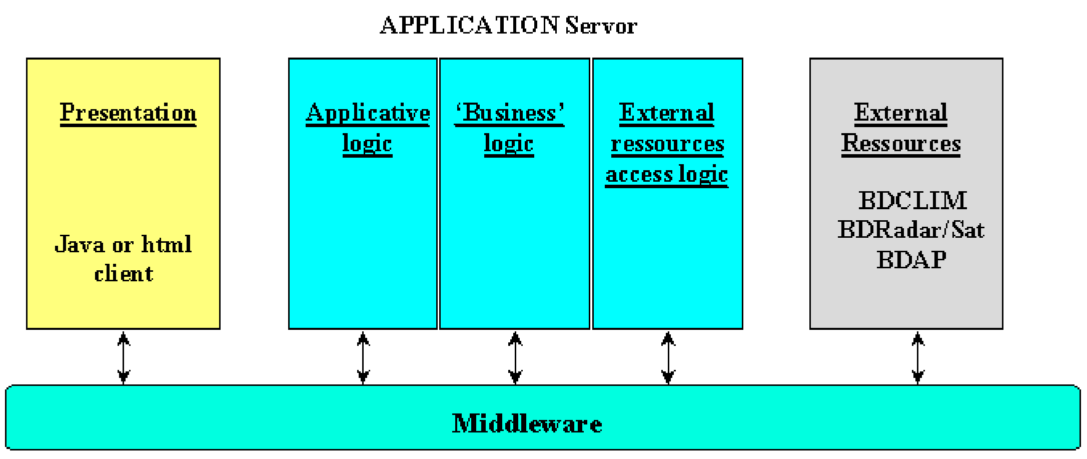


FIG.1 : the multi-tiers architecture

The high level Okapi architecture is a multi-tier architecture using Corba as the middleware. This architecture has a lot of advantages, especially it allows to clearly separate :
- external resources : fundamentally data bases for Okapi
- application server (called ‘production platform’ in the Okapi system) which contains :
  - external resources access logic : data bases access components
  - ‘business’ logic : ‘business’ components
  - applicative logic : requests management component
- presentation : in the Okapi system there are different client applications : ‘thin’ Web client and ‘heavy weight’ Java client.

3.2.2. Lower level architecture

Here is the description of lower level architecture for the production platform. We have modelled what a meteorological product is, using UML modelling language ; we have described a product as being a ‘tree’ made of three types of nodes. Each type of node has an associated kind of basic ‘business’ components :
- data base access component,
- ‘business’ processing component,
- layout component.

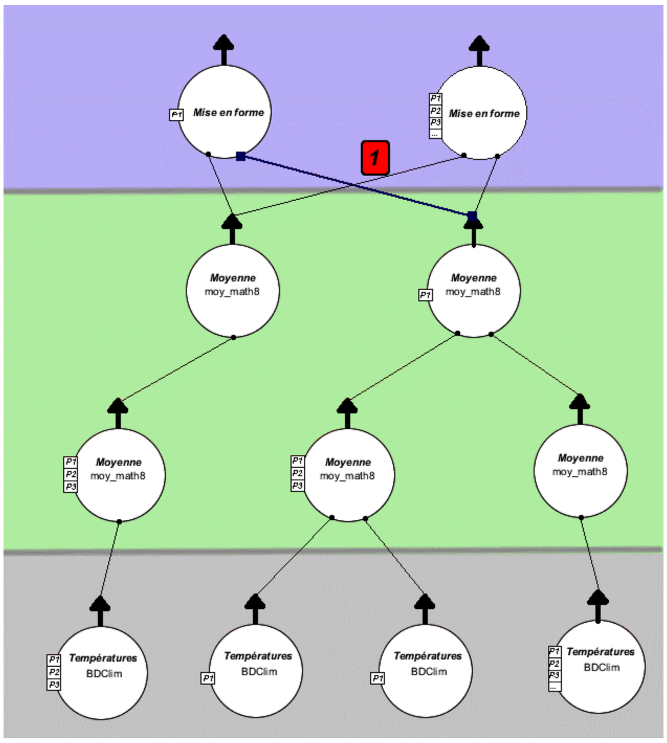


FIG 2 :
This schema shows how a meteorological product is modelled in the Okapi system. There are three layers :
- on the lower, the data base access components
- on the middle, the 'business' processing components
- on the upper, the layout components
This initial model has then been extended to take into account several layouts for a same product : simple ASCII file, graphic layout, …

3.2.3. Technologies

Technologies used for the production platform are Java, Corba and XML. Some pre-existing Fortran or C pieces of software are integrated using JNI (Java Native Interface).
We distinguish two sorts of clients in Okapi system :

- Web client which uses HTML, Javascript, JSP (Java Server Pages),
- Java client which uses Java Swing library.

3.3. Training and technical assistance

An important training phase was necessary for the Meteo-France project team : enterprise architecture with Java, UML analysis, developing with Java, Java Server Pages, Corba, XML, … and after this training phase it was absolutely necessary to benefit from a high level technical assistance for architecture, components approach of the system, and technical Java assistance for Meteo-France developers.
With this project Meteo-France wanted to keep the control of the whole system. The investment in training and technical assistance has been very important during the two first years, but after there are real benefits with a very evolvable and reactive system.

4. The new system

4.1. Description

The new system is logically composed of :

➢ a production platform : 'the products factory' which is able to elaborate any kind of product (illustrated by FIG.3). It contains :

- **Infrastructure components,** the main being :
  'Request management' component,
  'Product elaboration' component,
  'Products catalog' component.
- **Business components** :
  'Meteorological stations selection' component,
  'Data bases access' components : 6 (at the present time),
  'Business processing' components : 20 (at the present time),
  'Layout' components : 10 (at the present time).

➢ several client applications :

- Climathèque : the Internet application for commercial and institutional use (allows access to standard products),
- OkapiMet : the Intranet application for internal users (allows access to more standard products),
- OkapiStudies : the 'heavy weight' Java client for studies ; it offers a graphical shop to generate graphically and dynamically new type of products from basic 'business' components,
- Other client applications are expected ….

➢ some 'gateway' components :

- provide the interface with commercial management : authentification, estimated price before ordering, products accounting,
- provide 'client' services.

Figure 4 shows how the system is integrated in the Meteo-France production context.

4.2. The architecture and the components approach

To build an evolvable system, the architectural and design step is primordial. The project team may have the feeling that this phase of no real software production is rather long, but it has saved time for the following phases of the project and especially for its evolution.

In the case of Okapi the choice of the high level architecture (multi-tier) was rather quick, but the choice of the architecture for the production platform and the business modeling was longer. However early enough during the project we built prototypes to experiment with the different design choices and to allow the project to have 'something' to show…
In this kind of project another problem is to evaluate the components granularity : of course the first criterion is to maintain an object-oriented approach and to give functional consistence to each component. Okapi components granularity is rather small ; the system contains now about 60 components, but the number of 'business' components will increase.

To avoid this high number of components to turn into an administration nightmare we have built a system that make it very simple to host several components into a single process, and to start or stop them on demand. This makes it very easy to add new components to the system.

At the present time it is very easy to add a new product in the Okapi system ; the only thing needed is to create the description of the new elaboration process in the products catalogue (using an XML syntax), if needed to build some new 'business' components and to update the 'client' view of the catalog.

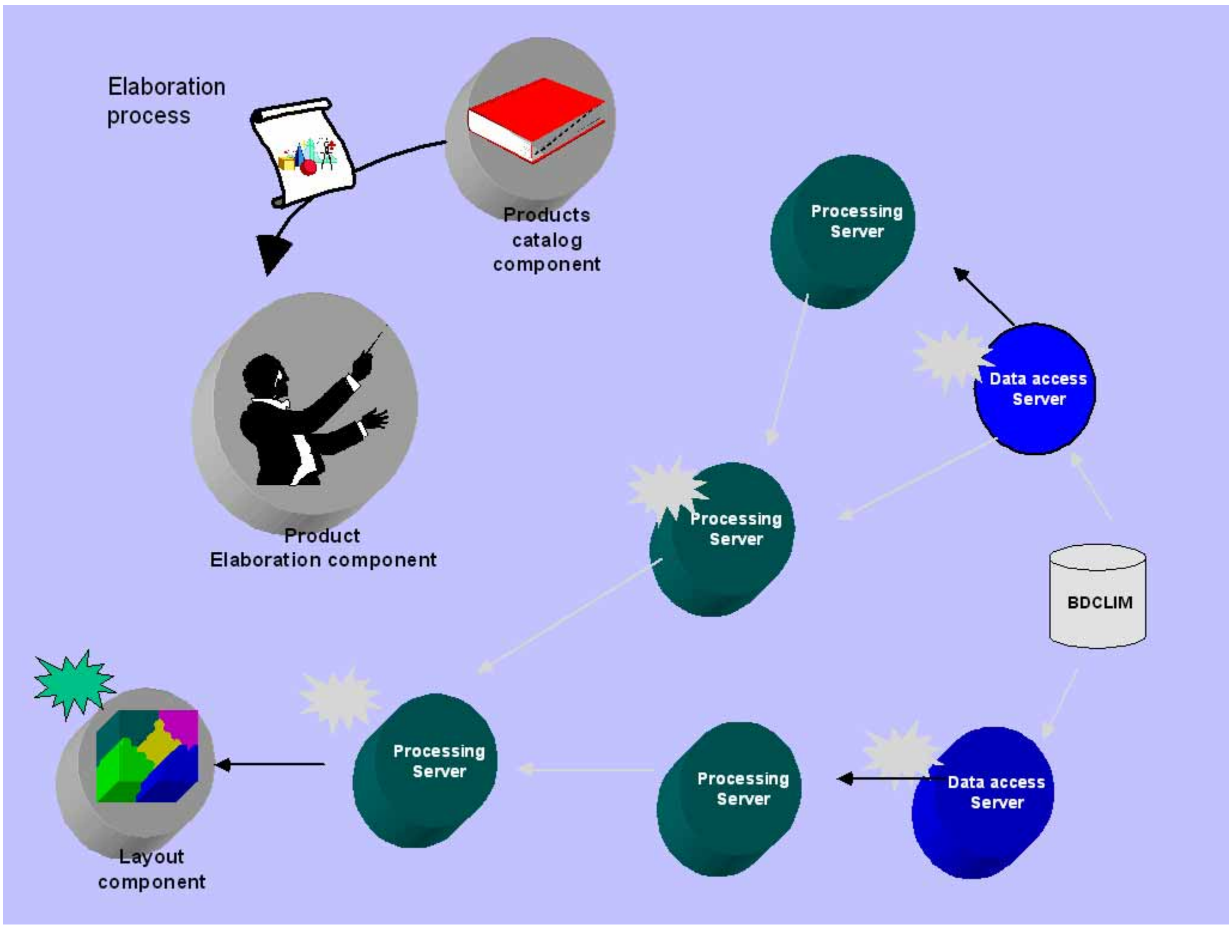


FIG 3 : Zoom on the production platform and how a product is elaborated by the 'Product elaboration component' which is the 'orchestra manager'

4.3. Use of XML in the Okapi system

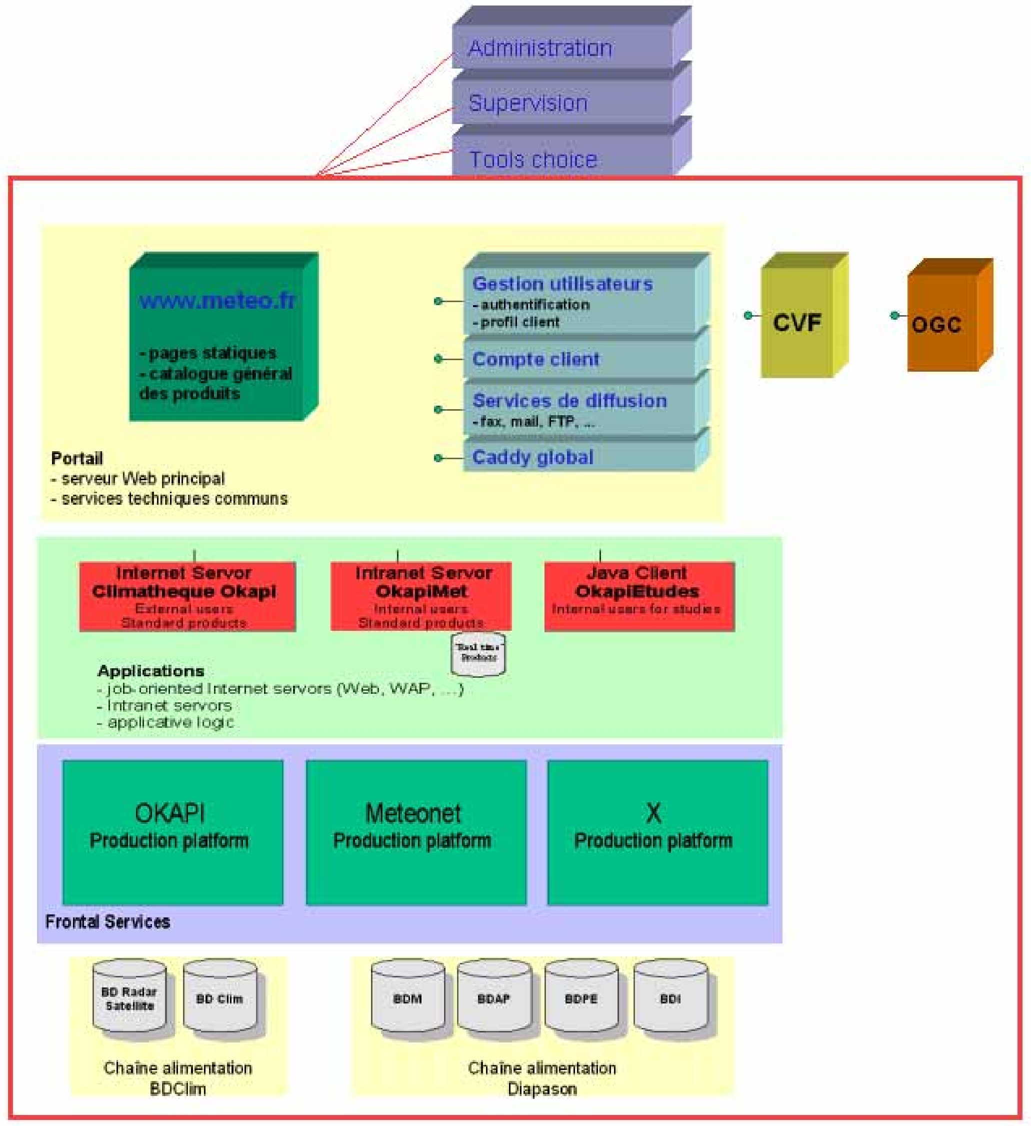
The Okapi system makes an intensive use of XML technologies for very different aspects :

1- Products catalog
The structure of the catalog is 'tree-like' and is described with XML ; the main advantages are that we can update it very easily without interaction on the applications and on the production platform.
It is also possible to have multiple catalog structures in order to personalize client application according to a client profile.

2- Requests syntax
The requests sent by client applications to the production platform are written with an XML syntax and are managed by the 'Request management' component. The modularity of XML makes it very easy to add new parameters to the request, to handle the 'business' parameters of new products, or some configuration parameters.
Recently we have added in the system the possibility for the user to manage some contextual parameters :

- getting compressed files for his products
- editing a different address on the products PDF file.
- asking the system to get back products by email
- …

All these contextual parameters have been integrated in the requests syntax.



Global View

Portal - Gateway
- main Web servor
- common technical services

Applications
- job-oriented Internet servors (Web, WAP, ...)
- Intranet servors
- applicative logic

Frontal Services
- production platforms

Data Bases

METEO FRANCE

FIG 4 : Global view of production environment

3- Data stream description
From the beginning of the project, one of the main objective was that any ‘business’ component should be able to interpret any data stream exchanged in the system. So we decided to add ‘data stream descriptors’ to each data stream and to use a common XML syntax for these descriptors. The modelling was rather long and we finally modelled data streams as hypercubes with a free number of dimensions.
Only the header, ie the descriptors, is using XML, the data themselves being in any format : ASCII, Binary, XML, special binary meteorological format as BUFR or GRIB, …
We then developed a java package called ‘iterators’ which makes it possible to read and write data streams which go in or go out every ‘business’ component.

The project uses a single DTD (Data Type Description) for meteorological data streams descriptors, no DTD for the product catalog structure, and XML Schema for the validation of requests.

4.4. Re-use

There are different levels of re-use that can be made of the Okapi system.

Re-use of the high level components :
- the production platform,
- the ‘gateway’ components,
- the client applications : Climathèque, OkapiMet, ...

Re-use of the lower level components :
- the ‘business’ components (data bases access, processing or layout components),
- the commercial interface component.

We already have some experiences of re-use in the context of cooperation with other projects. A business layout component has been integrated in the Java software of an international project in only two days. Some other ‘business’ components integration are forecasted in this project.
A part of the Intranet client application has been integrated in a Visual Basic application of Meteo-France.

In a near future re-use will be improved thanks to the use of WebServices technology and the SOAP protocol. The integration of these technologies will require very few developments thanks to the highly component oriented nature of the Okapi system.

4.5. Portability

Nearly all components of the Okapi system are developed with portable technologies. Only some ‘business’ components (encapsulated Fortran or C pieces of software) have to be set up on special operating system (such as HPUX). Okapi uses a distributed technical architecture with different operating system (at the present time Linux and HPUX), but the system can be set up on NT or other operating systems for which a Java virtual machine is available.
For the Web applications, no deployment is necessary for the client application, so we have a good reactivity to solve the problems or to evolve the application.

4.6. Load increase

The logical and the technical architecture of the Okapi system allow to easily manage the load increase. All components can be replicated and the load of the elaboration of each meteorological product can be dispatched across all the business components.

The main possibilities to handle a load increase are :
- duplication of the infrastructure components (request management and product elaboration),
- duplication of business components.

4.7. Operational status

The Okapi system is now operational. The delivery step is going on during summer 2002, and the client applications are expected to be available in September.
The system is supervised by the ‘Computer Centre’ of Meteo-France.

5. Conclusion

We more and more have to take into account the design, architecture and infrastructure implementation of a system if we want to be evolvable and reactive, able to answer quickly to the customers requirements. This

was a big investment that required hard work from the Meteo-France team to learn many new technologies and the external technical assistance to ensure clean design of the architecture.
But now the benefits are real in term of business and the developments are quicker and quicker. It is not only much more easier to create new products but it also leverages existing algorithms that were not easily re-usable. Most of the flexibility of the Okapi system features comes from the huge work spent in modelling and in architecture design.

The Okapi system has very few technical constraints thanks to the technologies used (Java, Corba, XML) and to distributed technical architecture. This system can be set up on any operating system for which a Java Virtual Machine is available.
The object-oriented and components approach allow to add new functionalities without upsetting all the system. In a near future we intend to insert in the Okapi system :
- graphical and cartographic aspects : the possibility for the user to make cartographic navigation to select the meteorological stations, and to elaborate cartographic products,
- automation of the production : it will be possible for the user to ask for a product elaboration at a specified frequency (each day or week at such time…).

We have some experiences of real re-use of Okapi components at different levels and for different needs.

At last we hope to make Okapi an open system :
- cooperation with other projects exchanging some business components,
- giving a user guide of the Okapi system, to allow other teams to develop components which will be integrated in the system,
- moving towards WebServices technologies and connecting the Okapi system to other external systems.

SUMMARY SESSION 5