

GNOME for business appliances: case study and architecture proposal

Igalia
Gutenberg 34B 2
15008, A Coruna
Galicia, Spain
<http://www.igalia.com>
info@igalia.com

ABSTRACT

Igalia is a galician company (Spanish state), at the north-west corner of the Iberian peninsula, devoted to the development of solutions based in free software. Last year we have developed a management system for a SME¹ enterprise using GNOME development platform. In this paper we'll explain our experiences during the project and the conclusions about the GNOME needs as a development framework for business appliances. The development last almost six months, including analysis and design of the application. The code has been licensed for our client under GPL and it's going to be available as a new free software project for the community on may, 1st. Nowadays the system has been working for almost a year in a distributed installation, deployed over 8 offices of the client. During the development we had to take some decisions about the use of the GNOME technology, trying to keep the focus in a general solution but having in mind the budget. The lack of a standardized server side component system or a multi-tiered standard architecture is still an problem to bet for GNOME. The use of GNOME as a development framework for this kind of systems will make the selection of GNOME easier for developers. To achieve this goal, we think GNOME project should discuss a complete development platform and architecture for management systems. We will show in the paper a proposal for this architecture, completely based in GNOME concepts and technologies, that would let us give to the users solutions related with general management systems (ERP/CRM)².

1. INTRODUCTION

Nowadays nobody doubts that GNU/Linux will be one of the main technologies for the desktop in the near future. GNOME project has greatly contributed to this situation.

¹Small and Medium Enterprise

²Enterprise Resource Planner/Customer Relationship Management

Under the umbrella of GNOME, a lot of wonderful libraries and applications have arisen; all this software has turned it into a great development platform.

In Igalia, we selected GNOME as our main development platform for multiple reasons. The most important reason for selecting this technology is that it is a standard, free development platform.

We are a company devoted to the development of **Free Software** solutions. Past year (2002) we have developed a business management application; in this paper we will show the work and the conclusions obtained during the project. The project goal was to create a system to cover the requirements of an automotive glass replacement and repair company, called *Auto Arte*.

In the last part, we will also put forward our ideas about the needs of specification for the GNOME server side developments. Trying to standardize the development of applications, GNOME has released the HIG³. This document points out the way we should develop the interfaces. The arrival of a component system, Bonobo, has shown the architectural framework for the applications. These technologies try to create a specification of the design GNOME applications should have. Business management applications usually will have a server side part, and we think this kind of developments should be made also following some kind of standard specification. This will allow the community to create software that could drive us to build a framework. The advantages of this framework will drive forward the use of GNOME as a development platform for business applications.

2. THE DEVELOPMENT OF A BUSINESS APPLIANCE

2.1 Brief history

Igalia is a Computer Engineering company devoted to the research and development of solutions in the field of Information and Communication Technologies. Free software technologies consulting is one of the main activities of Igalia-GNU/Linux and their associated applications.

In February, 2002, the company *Auto Arte* get in touch

³Human Interface Guidelines

with us, and ask us for a system able to cover all their management requirements. They knew nothing about **Free Software** but they had experiences with the problems of the proprietary solutions; this situation let us offer a **Free Software** solution. *Auto Arte* is an automotive glass replacement and repair company, it owns garages and warehouses all around Galicia. The requirements of the management comprise:

- POS⁴ for garages.
- Centralized management of the garages and warehouses.
- Communication using narrowband lines.
- Accountability requirements.
- GPL license.

We look for a solution using the software available in that moment but we couldn't find any application that would carry out the requirements of their management operations. There was some projects which targets coincide with ours but they were in a initial phase of development: *GNUe*, *ASPLFact*, *FactulaLux*, etc. The assesment of the business applications made with **Free Software** let us think that it was a field where much effort was needed. Although we considered the collaboration with that projects, our deadline was a big trouble.

After the evaluation of the options we decided to build our solution using the development platform we have selected, **GNOME**; **GNOME** had an office initiative we could use and it was the most interesting development platform.

The use of **GPL** license was agreed with the client, and it was discussed the release of the code and the advantages this situation will give to their company. Although the company required a custom-made solution the, development was made thinking in the future of the application as a general ERP solution.

The project lasted 6 months, from March to August of 2002. The state of the **GNOME** project in that period was a little bit fuzzy: the release team was doing a hard work to bring out **GNOME 2**.

2.2 First approach (Java/SOAP/GNOME)

During the first stages of the project, the analysis of the application was carried out, the requirements were found out and the technology tests were accomplished.

The operations that are done in this type of business have some special features. This features have to do with assurance companies and consortiums in the invoicing process.

The first architecture we tried was a multi-tiered one, having in mind the traditional client-server pattern. Essentially, it the following 3 tiers:

- Client: the interface of the system and the control of that interface.

⁴Point-Of-Sale

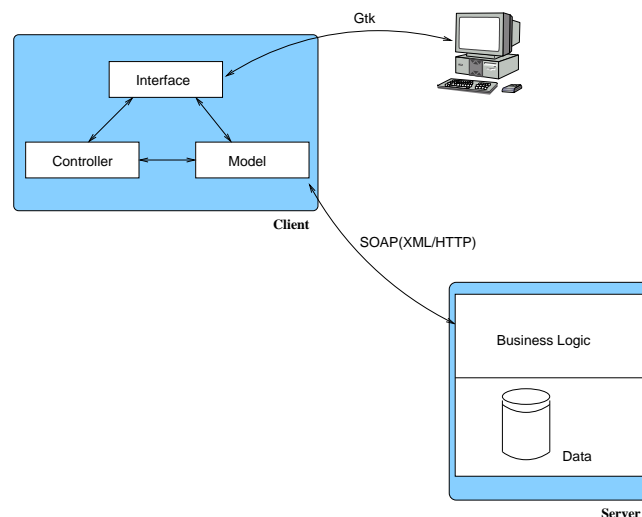


Figure 1: Graphic of the architecture of the first approach

- Server: the business logic and the data access.
- Database system; server to store the information of the application.

The communication between client and server was designed using **SOAP**⁵, the XML RPC protocol.

We start attempting to use **GNOME** technology in all the tiers, client and server, of the system but we found no information about the development of the server-side using **GNOME**. Thus, we decided to use a development platform that had information and libraries to be used in the server side. It also should support **SOAP**. We assess the choices and decided to use **Java**. It had a good **SOAP** support and there were a lot of patterns and architectures recommended for the server implementation.

So the test of the architecture (a prototype) shown in the figure 1 was developed. A test database, a **GNOME** interface and a **Java** server using **SOAP**. The **SOAP** server was deployed using *Apache* and the **GNOME** library *libsoup* was used to implement **SOAP** in the client side.

The selected database was *PostgreSQL*. The distribution of the application was done in this layer because it was thought at design time that this was the easiest way. To do this job we selected *PGReplicator*, which is a project that implements an asynchronous replica engine, and provides some data ownership models and many conflict resolution algorithms.

When we did the tests of the architecture some troubles arose:

- Awful performance. The reason of this performance was the patterns used to implement the data access in the server, that was done using **DAO/Accessor** in

⁵Simple Object Access Protocol

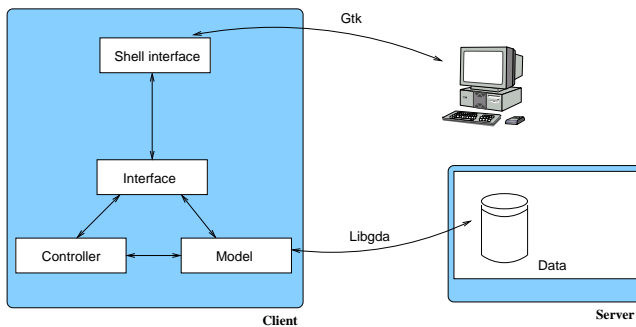


Figure 2: Graphic of the architecture of the final solution

Java; and the most important reason was the SOAP transformation. The data that traveled from the server to the client had to be transformed to XML, serialized, sent to the client and deserialized. SOAP can not work well for big amounts of data, because the computation with a big XML tree is very expensive.

- GNOME SOAP library was still under development. `Libsoup` was under development at that moment and was quite buggy.
- PGReplicator has an offline replication problem. The system to do the replicas required that all databases were up and connected when the process is being done. This was not possible because the nodes of the system were not reliable.
- Mixed technologies. The use of two technologies, Java and GNOME, made us to be worried about more libraries, documentation, environments, etc. This would require a lot of maintenance work.

2.3 Final solution (GNOME/libgda)

As a result of the problems described in the previous section, we decided to change the architecture. When we did this we had to take into account the milestones of the project, it was important to fulfill them.

In the figure 2 we can see the planned architecture. The general decision was to use just GNOME development platform. We outlined to do an multi-tiered architecture using CORBA between models and the controllers/interfaces. The deadline made us to start with a two tier architecture, but still ready to use CORBA in the future in order to build a multi-tiered solution.

The code was sorted out using a MVC⁶ pattern. As we said in the previous paragraph, the model has the access to the data and the business logic. The controller has the information about the interface management.

We used GNOME 1.4, and the language C. At that moment, GNOME 2 stable version was not still released. The database access was done using `libgda` (0.2.96). This was the last stable version of the library. The application trusts `libgda` to isolate it from the database.

⁶Model-View-Controller

The development environment was: GNU/Emacs and glade. Libglade was used to build the interfaces.

The solution to do the database replication was to develop over PostgreSQL. The system used a SQL sentence log that is used to send the operations in each database. The node of the center (central office of the company) has the responsibility of sending the useful data to each node. The sent is done using SSH connections to secure the data transmission.

This stage of the project lasted four months. The system was deployed in August (2002). The use of C language increased the debug time because of the memory management. Nowadays, the system has been working for almost a year.

2.4 Release of the code

When the deployment of project finished we started the work to publish the code. The plan stated the milestones until the first public release of the code. The work in the plan had to do with generalization and removal of some custom-made parts.

On 1st May, 2003, we published the code using sourceforge.net tools. The name of the Open Source project is Fisterra⁷. Although the documentation was in spanish the code was done in english, thinking about the future internationalization of the project. All the code was licensed using GPL.

Current release number is 1.4, still not a general solution for all kind of business. It's a public release of a system used to manage an automotive glass replacement and repair company. If somebody wants to use it to manage his business, first should check if the present features fit his/her requirements.

The features of the current release are:

- Clients, companies, agents, garages, consortiums and enterprises management system.
- Work order system: repairs, detailed piece information, vehicle, insurance (company, policy, receipt) and accident (photo management).
- Invoice management system: invoice emission, consults, collection, charge management (bank note automatic generation), general administration of enterprise positions and account conciliation.
- Warehouses management system: piece and stock management, multiple shops, inter shop transferences and order management.
- Account information system: VAT book, invoices management, summaries listing.
- Cash management system.
- Administrative management of the requirements to perform repairs in the garages.
- Multiple printing invoice format.

⁷www.fisterra.org

- Distributed database system using PostgreSQL, it can work offline and using low bandwidth lines interconnection.

The future minor features (release 1.6) will be:

- Complete configuration data administration.
- Improvements to the replication system.
- Modifications to the invoice structure.
- Database access tune.
- Others proposals from the developers community.

We hope this release will be ready before the end of the summer. The next major features (release 2) will include:

- Gnome 2 support (HIG).
- Implementation of the new architecture, we expect GNOME project could specify this architecture.

The project has already been downloaded and examined for a lot of people, we hope this could be a good start line to do work related with business applications in the GNOME platform. The target will be to build a collection of tools to do business management, the project will try to cover the general requirements of an CRM/ERP solution.

2.5 Project conclusions

After the final stage of the project, the development team has some doubts about the used architecture and the ideal structure of this kind of applications using the GNOME platform. The main conclusions are:

- Business appliances have the need of a client/server architecture. We have to take into account that we are developing applications in a environment where multiple users work with the same information following a workflow. The use of an intelligent server (more than a simple database level) has too many advantages to be discarded.
- GNOME lacks server side implementation specification. We should ask ourselves: is GNOME just a development platform for the desktop? Or even we can ask: would it be enough for GNOME to be a development platform for the desktop? In general we think the answer is no, GNOME should have an architecture for component-based computing to reach the level of other development platforms. We think that even if we answer yes to the questions GNOME should specify which technologies can be used and how they should be used in this kind of projects. The specification will reinforce the business applications developments, and the spreading of the platform could get better. The specification of the server will have some advantages:
 - Standardization of the developments.

- The reuse of code will be made easier.
- Growth of the general quality of the server developments.

- Client/server communications: SOAP, CORBA, etc. It will also interesting to study the specification of the standard communication system: which protocols should be used and when? This is a important part of an architecture.
- GNOME as a complete development platform. We think the effort to build specifications for the sever side and the communications should be strengthened. It will be very interesting to create a sever side specification because the advantages referred before. We even think that the specification of the recommended architectures with technologies different from GNOME has less interest and could cause some troubles. We could have troubles with two libraries that implement the same functionalities, one from GNOME project and the other from the technology used in the server side. This issue should be considered inside the community.

3. NEEDS AND PROPOSALS

3.1 GNOME technology and business appliances

As we could see in the previous section, the server side specification is an interesting documentation to develop business appliances. It will be an interesting work to bring GNOME near the server side.

We can think about the features that the architecture should have before starting its definition:

- Object storage. The specification should have the rules to build a data access system. Persistence of the objects should be transparently implemented. The developer would have to point out the database structure and the relation with the classes. We can not forget the need of a good cache system and a connection manager. At the moment libgda does some of this tasks.
- Business objects. It will be important that we could isolate the business logic in objects that have some standard properties and behaviors. Like in the GTK+ environment our objects will need common properties and operations.
- Transactions support. This kind of applications usually runs as a workflow, a document that goes through some stages. For this kind of operations a transaction system is very important to control the working.
- Fault tolerance. Is is important that the architecture could control some errors, the system should do well in this situations. We could implement supervision trees to control the state of the objects. We could also control the state of the servers placed in different hosts.
- Scalability. One of the preferred features will be a way to take into account the performance of the system. A transparent object distribution would allow us to move the objects between hosts trying to improve the performance.

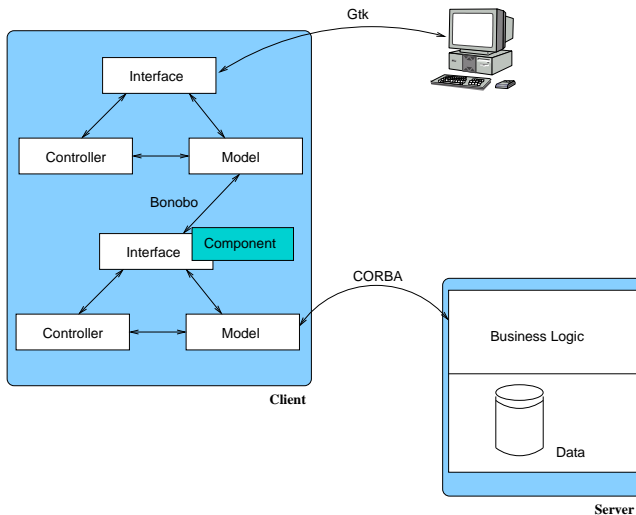


Figure 3: Graphic of the proposed architecture

- Security. It is necessary to build an infrastructure that let us control the access to the system, data and services. This should be done easily in the configuration of the server.

We think the most interesting general architecture of the server would be a container. The container of *GTK+* has shown that it is a very flexible and advanced solution. We could use *GTK+* container and add some extra controllers over it. Thanks to the separation between *GTK* and *Glib* we could develop and use the container without including visual libraries.

3.2 Architecture: initial proposal

During the publication period we started a work to develop a prototype of the wanted architecture. The results of this work will be released in the web of the project. We are working in the first approach of this new architecture. The work is being done using **GNOME 2** libraries.

In figure 3, the graphic of the proposed architecture can be seen.

The parts of the architecture are:

- Client/server architecture. As we have explained, this architectural pattern is very interesting in this kind of applications.
- Server. To design the server we used a multi-layered architecture. The main layers are two, the business logic and the database system. We can't forget the features: distribution, transactions, security, etc. We will leave any implementation of this services to next stages, the prototype will just implement the main features. We will use *libgda* to do the data access and *gobject* to implement the classes.
- Communications. The communications between client and server will be carried out using CORBA.

- Client. In the client side we will use the MVC pattern to structure the code of the Bonobo components. The model will use the CORBA interface to access the server objects. The components will be connected using Bonobo, the composite pattern fits well here. Interfaces will be built using libglade.

3.3 Conclusions

As we explained in the conclusions of the project, the most important ideas have to do with the specification of the server side for the **GNOME** applications. Business appliances have the need of a client/server architecture and we think **GNOME** should specify which technologies can be used and how they should be used.

The framework of the server should have some features:

- Object storage.
- Business objects.
- Transactions support.
- Fault tolerance.
- Scalability.
- Security.

We think the **GNOME** community should discuss about this issue and propose a guide to the developers of this kind of applications. This will let the community to create software that could drive us to build a framework for the server side developments. The advantages of this framework will drive forward the use of **GNOME** as a development platform for business applications.