Proceedings of the TOOLS 29: Technology of Object-Oriented Languages and Systems. 1999, p. 251-260

TRUNIS: an Object Oriented Trunking Radio Telephone Network Information System
An experience report[1]

F.J. Zarazaga, J. Valiño, S. Comella, J. Nogueras, P. Muro-Medrano
Department of Computer Science and System Engineering
Centro Politécnico Superior, Universidad de Zaragoza
María de Luna 3, 50015 Zaragoza (Spain)
{javy,juanv,scomella,prmuro@posta.unizar.es}
{http://diana.cps.unizar.es/iaaa}

Abstract

The paper shows our experience in developing a trunking radiotelephone network information system using object-oriented technologies. The developed information system has the network management responsibility and it offers a complete set of utilities to the network operator to facilitate its work. It has been necessary to balance requirements from the management point of view (real-time interaction, statistics, etc.), requirements from the technical connection with the network (different technologies, buffering problems, etc.), and extra requirements imposed by the client (remote management option, easy to platform migration, etc.). Representation techniques based on object meta-knowledge have been used as a powerful tool to provide a flexible and easy to extend mechanism to cover application design and implementation phases. The adopted architecture and the main components and their orchestration are described. This paper includes learned experiences, development criteria, problems and choices of commercial products.

1. The project context

In recent years, competition among telecommunication companies has made necessary for most of them to extend their business area. To give a good communication service it is not enough, now it appears necessary to offer a set of added value services. In this sense, a power and easy to manage network information system has its own business contribution. Object oriented technology is gaining bigger interest within the developers of applications related with the operation, management and administration computer controlled networks (see [3], [6], [9] or [13] for relevant examples on this area), although this interest can be found in other domains such as can be seen in [7].

This paper describes our experience in developing a trunking radiotelephone network information system using object-oriented technologies. The system, named TRUNIS (an abbreviation for TRUNking Information System), is a R&D project developed by collaboration between the company Teltronic s.a. and IAAA research group from the Department of Computer Science and System Engineering at the University of Zaragoza (Spain).

Teltronic s.a. is a Spanish company dedicated to produce, market and install

[1] This work was partially supported by the *Comisión Interministerial de Ciencia y Tecnología* (CICYT) of Spain through the project TIC98-0587 and *Teltronic* s.a. trough the projects 95/0015 and 98/0131.

professional radio-communication systems. Their products have been installed in many countries along Europe, Africa and South America. It is in direct competition, in trunking networks, with big multinational companies like Motorola, Philips, Kenwood, Key, Tait, Fylde Microsystems, Nokia, etc. The IAAA is a R&D group composed by professors, associated engineers and undergraduate students from the college of engineering (industrial, telecommunications and computing) at the University of Zaragoza (Spain). The IAAA work in areas such as software engineering for information systems, object oriented and component based systems, distributed systems and open geographic information systems. Object oriented technologies have been adopted as the base of all works developed by the group, and represents its most important “know how”.

TELTRONIC S.A. is specialising in the development of hardware radio components and its corresponding embedded software. The development of a medium size information system application, based on a PC/workstation, was out of the scope of its development interest. On the other hand, the IAAA group had extensive experience in the involved technologies to develop that kind of product. That was the context where the collaboration was established, which was fomented by interesting public financial opportunities derived by the R&D collaboration with Universities.

The rest of the paper is structured as follows. Next section provides a basic introduction to trunking systems and its management. Section 3 shows the technologies and tools used within this project. Section 4 describes the architectural design and its major components. Section 5 focuses on our experience with the use of object oriented technologies and related tools involved the development of TRUNIS. This work ends with a conclusion section and the project continuity in the future.

2. Introduction to trunking systems

Trunking has long been a part of the Public Safety and business side of radiotelephony. It is based on the idea of offering more efficient use of a system repeaters and equipment. The term "trunk" actually came from the telephone industry. When a phone call is placed, the call is sent to a line where it is given the next available open line to its destination. This may happen several times on a long-distance call or once on a local system. The same is true for radio trunking.

An easier way to understand trunked radio is to think of it like standing in a line for a bank teller inside the bank rather than driving into the drive-thru. In a drive-thru every one picks a line for a specific teller machine. Some machines are used more than others and some are used less. While on the inside, everyone must stand in one line and wait for a teller to become free so that the customer can make the transaction. When that customer is gone, the teller is free to take the next person in line. While on the outside, people are still stuck on the line waiting to get to the next open spot in their particular line when there maybe 3 teller machines open in a different line. In a trunking network, "tellers" are radio frequencies, "customers" are radio terminals waiting for communication, "transactions" are calls, and the responsible of the co-ordination of all the system is the Central Node Control Unit.

Figure 1 shows a typical trunking network infrastructure that is mainly composed by a *Central Node (CN)* and a set of *Zone Base Stations (ZBS)*. The NC is in charge of the overall network control and it is itself composed by a *Central Node Control Unit* (CNC) and a set of hardware components to establish the communication paths (communication cards, commutation matrix, etc.). ZBSs are connected to the CN by links and have to provide the coverage for radio terminals. If a radio terminal is into a ZBS coverage, it could ask for a call,

the request is communicated to the CNC that decides if the communication could be established.

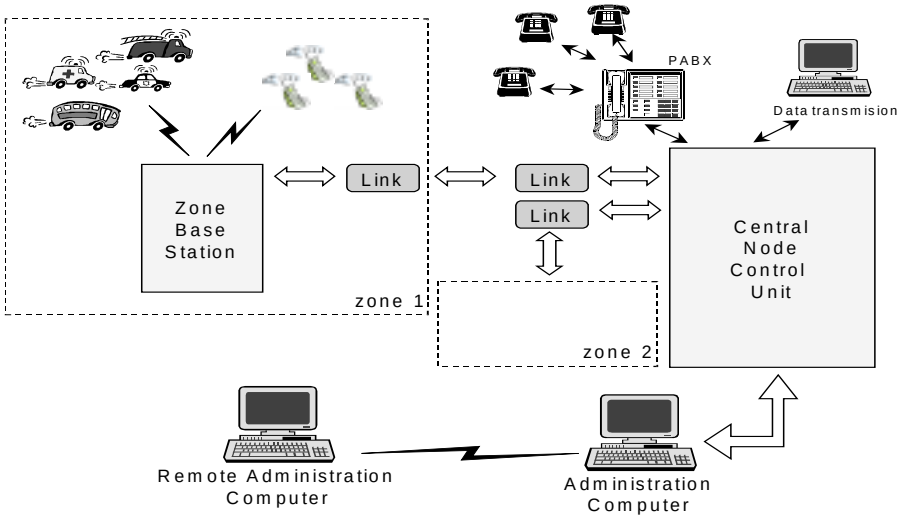


Figure 1: Trunking network infrastructure

TRUNIS is the interface between the network and its operator, this person, or set of persons, is who makes the supervision, control, maintenance and operation of a trunking network. It is installed into the central administration computer that is connected to the network through the CNC. A serial line is the physical link between them. This serial line could be a cable, a radio modem, a telephone modem, etc. A remote administration computer can be connected to the central administration computer via "point to point" modem connection, via Internet, etc. This remote computer could provide the same management functionality than the administration one. This remote access offers the opportunity to share the management work and allows to make remote supervision and demonstrations of the system, even in other city or country.

In the development of the management application, two groups of requirements have been identified:

- User services have their origin in the control, maintenance and operation of the trunking network, and could be grouped into three sets with very different characteristics:
  - Network infrastructure data and access
  - Use information: statistic, fleet and communication element configuration, invoicing, client configuration, etc.
  - Information system administration: user management, backups, etc. End users could be categorized and limited in its access to the application functionality.
- Requirements and constraints derived from the connection of TRUNIS with the trunking network. As it has been explained above, TRUNIS is connected to the trunking network through its CNC. This is an industrial PC (Pentium 133 MHz., with 4Mb RAM and 6Mb HD) with MS-DOS version 6.20 and a main control program written in C language.
  - There is only one point to communicate the network and its information system. This is a computer serial port that is connected to the network via modem, “serial cable”, radio link or others.
  - Information coming from the network and going to the information system must be integral and complete. This involves the design of a database to allow sharing the information among different services offered by the information system. There is an additional representation problem because the CNC uses a world representation designed to minimize storage and maximize access speed and it has a high penalty in

complexity and understanding. On the other hand, the information system must provide a data design with a high understanding level and low complexity. This is because it must be designed to be closer to human operators (with maybe a low technical preparation level), while CNC only "talks" with other hardware elements (cards, PCs radio terminals, etc).

- The computer used to control the CNC has all its databases kept in memory in order to minimize hard disk access and to be more efficient (trunking regulation establishes the maximum time interval to answer to a communication request). This makes the buffer capacity of this computer for historical information (callings, alarms, etc.) minimum and it is necessary to transfer that information to TRUNIS databases all the time.

Furthermore, to avoid human interaction the system must do, automatically, a set of periodical works (computer clock synchronisation, the transfer of network work data too bigger to be kept by the CNC PC, etc). Other additional requirement is the capacity of the system to be operated by more than one operator at the same time, using different consoles, even in remote mode via Internet or by telephone access. Figure 2 gives an idea of the high level functional architecture of the application.

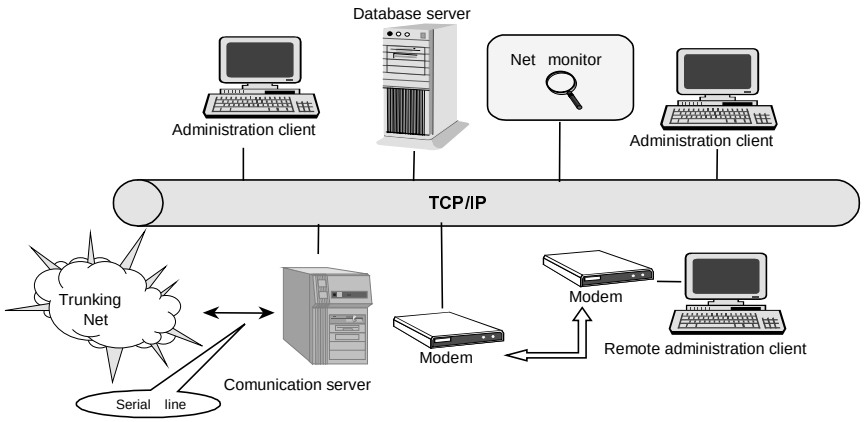


Figure 2: High level application structure

At the beginning, the system had to be platform and database server independent[2]. This strong requirement was relaxed and transformed to "cheap to migrate". Actually, the application is operating over a Microsoft NT operating system and the database server used is Oracle. But migration to other platform or database server should be no traumatic and with relative low time costs, this is because designs and implementations have been done having that problem in mind and the use of reusable design methods (object oriented methods) and languages (standard C++ and SQL-92) was fomented.

Finally, the client wants to "customize" the application in order to limit functionality so that the system could be sell to their clients as different versions of the application with different sets of capabilities.

3. Technologies and tools

Object oriented technologies have been the basis to succeed in the development of this application: Object oriented methodologies have been used for analysis and design process, and an object-oriented program has been selected for the implementation process. Analysis and design have been done using OMT [8] with a personal extension to support class and attribute

[2] In 1994, Java was a panacea that lived only in Sun research laboratories.

meta-information. Because of this extension, no commercial CASE application could be used and diagrams had to be built using a drawing application named Visio (version 3.0 and 4.0). Currently, as specification is changing during maintenance efforts, it is migrating to UML [1].

Since class and attribute meta-information is considered as one of the most powerful tools used in this development process, a special library designed to support this kind of meta-information has been used. This library is called *Facet Library* [12] and provides a strong knowledge representation scheme, specially built to represent typical information systems knowledge (data model information: attribute names, types, etc; GUI information: appearances, value representation sizes, etc; database information: table names, column names, etc.; ...). This design strategy based in the use of meta-knowledge facilitates that models developed during analysis could be moved through different developing phases with large similarities. The approach to use the same model during developing phases has been adopted in a bigger extend by other authors like in [5].

The programming language selected to build the application is C++ [10]. It offers powerful resources and many capabilities to use system libraries (GUI, database management systems and communications) and source code developed in other projects.

In order to support object persistence, *Facet Library* has an extension to provide a transparent object persistence mechanism especially designed to allow database server migration with minimum changes. This objective has been achieved limiting the number of classes that know the database server, and using SQL-92 [2] as unique database query language. To contrast this migration facility, two prototypes have been implemented, one using Informix database engine (over Silicon Graphics machine, IRIX 5.x) and another using Windows ODBC drivers (for ORACLE database). In the same way, to make a possible platform migration easier, a commercial multi-platform GUI library has been used. After a selection process where functionality and economic cost were considered, the final library selected was Zapp version 3.0 (from Rogue Wave software).

The application is currently operative over Microsoft NT version 4.0. with Oracle as database server. Initially, the application used Oracle workgroup server 7.3, but at present, last release of the application uses either Oracle 8 server version 8.0 or Personal Oracle 8 version 8.0. This change of database version has been done because Oracle does not sell version 7 licenses of its database engines. The fact is that two of the currently operative installations use Workgroup server version 7.3, and the other one uses Personal Oracle 8 version 8.0. The access mechanism to the database selected was the use of a C++ precompiler as it is faster than ODBC access mechanism. The pre-compiler used is Oracle Pro C/C++ version 2.2 (to access to Workgroup server version 7.3) or version 8.0 (to access to Oracle 8 server and Personal Oracle 8). The development has been done with Microsoft Visual C++ version 4.1. This compiler and version was imposed by the database selection because Oracle Pro C/C++ guaranteed its code generation only for this compiler and version.

As could be seen on the next point, the application has been developed to work in a network with different computers executing different functionality. Nevertheless, with an appropriate configuration, all application could be executed in the same machine at the same time. This machine should be a compatible PC with: Intel Pentium 133 MHz processor or upper, 64 Mb of RAM, 1 GB hard disk, 1 free serial port (usually one is used by mouse). It is also required a graphic card with enough capacity to give 16 colours for a 17" colour monitor with a minimum resolution of 1152 x 864 pixels. This resolution is necessary since there are some windows with a big amount of information and can not be properly shown with less resolution.

4. Architectural and component description

In this section we are going to present the architectural design of the application developed. This architecture is based on two major elements that are the concept of subapplication as management functional component, and the object data model that works as conceptual representation of the trunking network and its interactions.

4.1. Architectural design

During the development of this application, the first step was to design the architecture of the system. This architecture provided the major elements to be developed, their functionality and their relations among them.

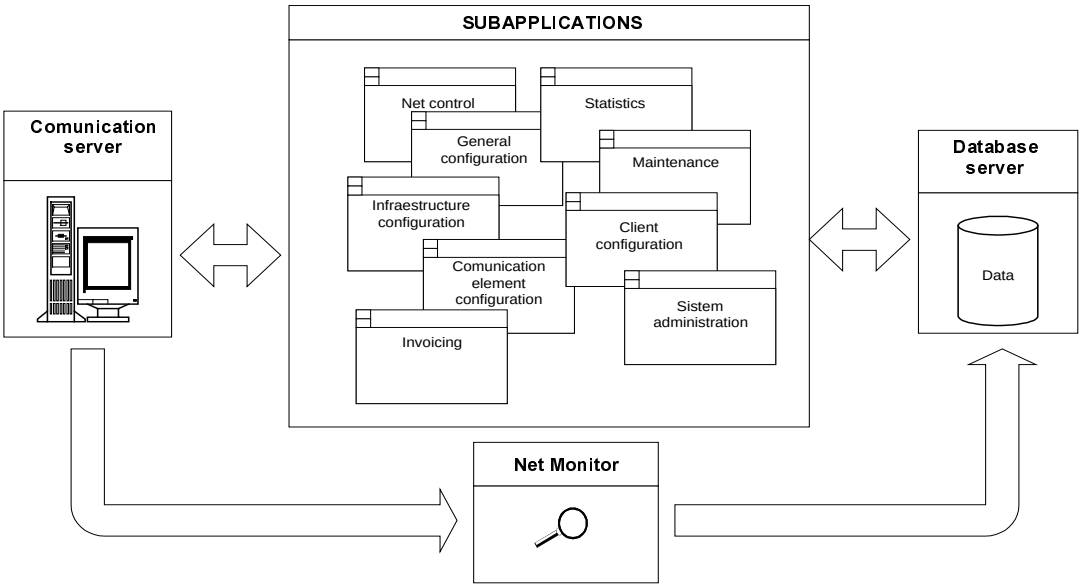


Figure 3: Architectural design

Figure 3 shows the architectural design developed for the application in order to satisfy all the requirements introduced before. The following elements can be identified:

- A database server: there must be only one database repository in order to have an integral and complete data management. This repository works as a server of the rest of elements, which work as clients of it. The database server is implemented using a commercial database server (Oracle) and its corresponding clients. Over these clients, a set of classes has been developed to offer an interface so near to the problem domain as possible.
- A communication server: this element is the only access to the trunking network in accordance with the requirements. It works like a server providing the lowest layers (from 1 to 4 using OSI standard [11]) of the protocol used to communicate the network with the application (this protocol was provided by Teltronic s.a., developed by their trunking team, and it is the same used to communicate their computers inside the network). It is an autonomous program, which captures a serial port and offers their services to clients via socket connection (IP and port could be configured via init-file).
- A network monitor: this element is necessary to "listen" the network every time and to execute automatic operations. Two programs compose the monitor. One of them is responsible for "listening" permanently the network and keeping data coming from it into

the database. The other program is responsible for executing automatic operations. Both are registered as an NT service too, but, while first starts as soon as the computer where it is installed is switched on, the other is programming to be executed one day of the week at a special hour both configured via init-file. These two programs work as clients of the database server and of the communication server.

- A set of subapplications which configures the administration client and represents the console for operator interaction with the information system (the concept of "subapplication" is described in the next point). It consists of an executable specialising to check the configuration of the application sold by Teltronic s.a. (see application requirements) and to allow the execution, via tool bar or via menu, of the corresponding subapplications, each of them is an autonomous executable program (see next section).

4.2. Subapplication designs

Administration client design is based on the concept of "subapplication". A subapplication is an autonomous executable program that offers a set of functionality closely related. This program could be executed only via the interface provided by the main application that works as a subapplication shuttle.

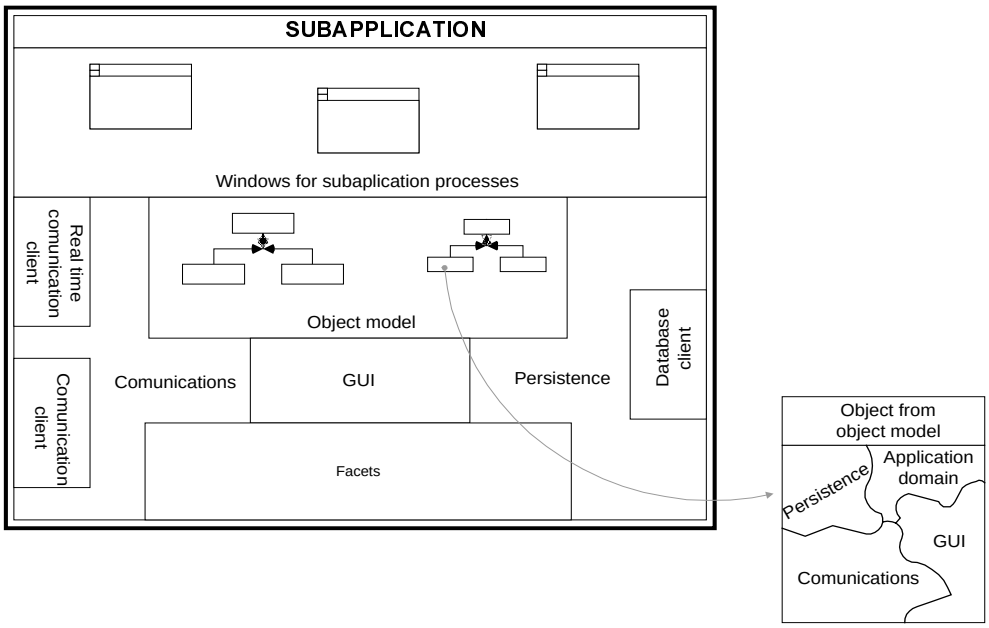


Figure 4: Subapplication scheme

Figure 4 shows the basic architecture of all subapplications. This architecture is supported by a powerful knowledge representation scheme [12], which allows integration and centralisation of the information system knowledge. Based on this kernel, some infrastructures have been built in order to provide utilities of general interest for information systems development: graphic user interface, database persistence and data transfer to the trunking network. These utilities involve the design of new objects and services for each utility, and new attributes specialized in those utilities. GUI infrastructure provides the necessary mechanisms to build application windows allowing the access to the data model information. The communications infrastructure role is to give appropriate tools to obtain the necessary object information when the network provides this information. Persistence infrastructure has the responsibility to provide objects with methods to manage their own persistence into a relational database.

4.3. Object data model

The object data model is the kernel of the application. It is integrated by all classes that represent elements from a trunking network organized in three major hierarchies. One of them, perhaps the most complex, represents the elements that could take part in a call inside the trunking system (see Figure 5). Another is composed by classes that manage the application administration (users, backups, etc.). And the third one is integrated by all classes connected with invoicing and client management. There are other minor hierarchies and several relations among classes from different hierarchies.

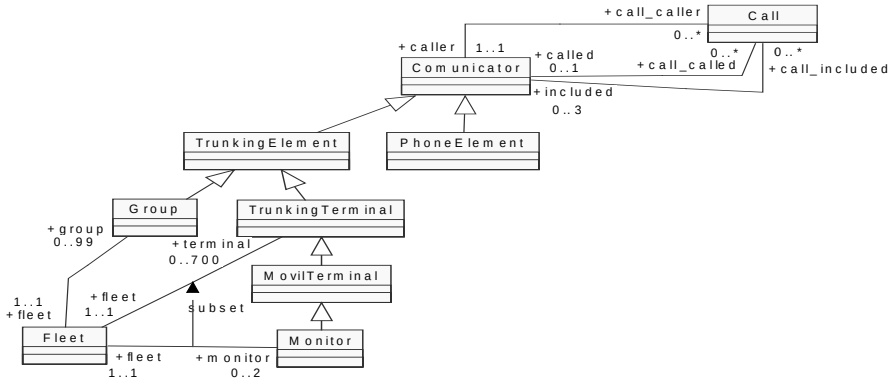


Figure 5: Object model sub-set

5. Related experience

This was the most complex project our research group had faced up to in its beginnings. This way, the results obtained have implied a remarkable improvement of our development process as well as our technical experience. The related experience is summarized below.

The use of object-oriented technologies

Object oriented technologies have been very useful in the development of this project, from the beginning to the maintenance phase. During the first steps, objects allow us build a system model understandable by Teltronic team, they did not know anything about objects, but they could see their network in our diagrams. Object design has made the organisation of the knowledge about the system, the division of the implementation work (up to 6 programmers have been working at the same time), the realisation of the test and the integration of elements easier. C++, as object oriented programming language, has offered a big set of capabilities to organize source code, making programmers relocation and maintenance labours easier

Facets libraries

Thanks to the facet libraries and the macros used to manage them, the coding task has been remarkably improved. The programmer must only describe what it is shown in the design model. For example, giving persistence to a class is so simple as the use of a set of macros (implicitly facets library). These macros are used to declare this class as persistent and to indicate the characteristics of the persistent attributes (data type, length, null values, unique...). Once this job has done, the programmer is automatically provided with a set of methods to interact with the database (load, save, update, delete, create...) as well as the basic methods to access to the value of the attributes or relationships with other classes. As you can see, coding is really easier now as it is almost a direct transcription of what you have done at design time. This way, the programmer needs only to concentrate on the business logic of the application avoiding tedious and repeatable tasks such as interaction with the database, get and set methods for attributes and so on. It has made easier for new programmers to understand source code, in conjunction with clear designs and well commented code.

Programming language limitations

To implement facets library, some of the most advanced C++ features have been used, more specifically, new standard language additions in template and member pointer areas. In some cases, it has been problematic to "understand" these new features and to "make" the compiler capable to support them. In this way, the extensive use of templates has made compiler time very high (to compile the complete application could costs about 48 hours using a PC 233 MHz and 128 Mb RAM), even using precompiled headers.

Problems with compilers

Source code files have many structures like `#if #else #endif` to include source code according to the C++ standard and the patched version which could be compiled by the environment used. Initially, four compilers (Borland C++, Microsoft Visual C++, SGI C++ and gnu C++) were used in order to detect the origin of some problems found in execution because it was no clear if problems have their origin in programmers or in compilers. It was necessary to use those structures because different compilers have different problems. Database server selection forced the use of Microsoft Visual C++ because C++ precompiler tool generated compatible code only for this one. Many hours have been lost detecting and solving compiler problems, and the exhaustive use of debug tools have been necessary.

Problems with Graphic User Interfaces libraries

One of the most expensive processes was to select an appropriate GUI library. According to initial requirements, the use of a multi-platform GUI library was decided in order to make the migration process easier. Several libraries were considered: Ilog views, C++ views and Zapp. The first one seemed to be the best option but it was rejected due to its expensive cost. The second one was bought but when the real work began, its functionality (in multi-platform services) was not enough for our requirements. Therefore, the final selection was Zapp.

Zapp is a GUI development library. It was chosen mainly by its multi-platform capabilities, its flexible design and because it is not a whole system framework as MFCs, therefore it does not determine any system architecture or deal with database interaction, system logic and so on. Zapp provides developer with basic GUI classes and a small amount of higher level classes. It also offers a graphic-developing tool, called Zapp Factory, which speeds up developing process but generates a quite awkward code. Unfortunately, we have found some problems using Zapp. Firstly, the library is difficult to mix with other class libraries such as STL. Secondly, Zapp aimed to be used in addition to Zapp Factory. Instead, we have needed to exploit every capability of Zapp using directly low level functions of the library in order to integrate the graphical aspects of objects inside the facet structure and building window hierarchies. For this reason, we had to face not well tested capacities of the library, and we have found some very hard to find and unreported errors. Finally, the documentation provided is not so complete .We have experienced this lack of clear documentation by developing ourselves a service, which was found later in the library but it was no documented.

Lessons learned

There exists a typical set of recommendations to the development of projects using object-oriented technologies that is extensive documented in bibliography. We would like to add two more lessons we have learned during the development of this project. On one side, we suggest to be suspicious about commercial tools (compilers, libraries, etc.) capacity when new technologies are involved. Test, test and test them. On the other side, when one project is going to be an industrial product and not only a laboratory prototype, you should try to minimize the risks of use new technologies by adding their functionality step by step. It is very useful to encapsulate them in order to be able to check them separately from the rest of the application by using black box tests and integration tests.

6. Conclusions

In this work, the development of a trunking network information system using object-oriented technologies, and the related experience with it, have been presented. We considered the development of this application a medium size project. More than 350 classes have been designed and implemented, including object data model (about 35 classes and 15 relations between them), all subapplications, basic facet infrastructure, services for database access, communications, GUI and print services, and general purpose (exceptions, strings). The team responsible of the project development was composed of 9 persons (a maximum of 7 working at the same time). With these characteristics, the use of object-oriented technologies and tools has been one of the most important elements in the success of the development of the project.

Nowadays, this application is working in several trunking operative networks installed in Alicante, Viladecans (Barcelona) and Gran Canaria (all in Spain). There are different configurations of the same application, and the evolution of each one has been different. This way, object oriented technologies allow us to make the maintenance of all of them easier as they provide us with the necessary encapsulation mechanisms to change some application aspects transparently to the rest. In the same way, the addition of new functionality is currently the other line of work. The improvement work has two major development lines, one is extending the invoicing and client configuration services in order to integrate them with the network operator company business organisation, and the other is providing new services to network clients like "mini" information allowing them to access to their "own" information.

As Brian Henderson-Sellers says in [4], object technology is still immature although developing rapidly, in this way, the develop of a medium size application using object oriented technologies has provided our research group with a very important "know-how" in the use of them, their benefits and pitfalls. This knowledge is being applied in the development of new projects and in teaching our students at the university.

7. References

[1] G.Booch, J.Rumbaugh, I.Jacobson*: "The Unified Modeling Language User Guide"*, Addison-Wesley 1998.

[2] C.J.Date, H.Darwen, A guide to the SQL standard, Third Edition ; Addison-Wesley, 1993.

[3] EHPT, a company that develops software telecom management applications: http://www.ehpt.com/index.html

[4] B.Henderson-Sellers: "A book of Object-Oriented Knowledge. An introduction to Object-Oriented Software Engineering", Prentice Hall 1997.

[5] R.B.Jackson, W.C.Giauque, J.V.Hansen: *"Concepts of single-paradigm object-oriented development, with application to a manufacturing information system"*. IEEE Transactions on Systems, Man and Cybernetics. Pages: 583 - 598, Sept. 1996, Vol. 26, Issue: 5.

[6] C.Kroll: *"Object oriented network operation, administration and management"*. Proceedings of the IEEE Symposium on Computers and Communications, 1995. Pages: 50 - 56.

[7] M.Missikoff: *"An object-oriented approach to an information and decision support system for railway traffic control"*. Proceedings of the First International Conference on Knowledge-Based Intelligent Electronic Systems 1997. KES '97. Pages: 633 - 641 vol.2.

[8] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy, W.Lorensen, *"Object-Oriented Modeling and Design"*, Prentice Hall, 1991.

[9] R.H.Stratman: *"Development of an integrated network manager for heterogeneous networks using OSI standards and object-oriented techniques"*. IEEE Journal on Selected Areas in Communications. Pages: 1110 - 1120, Aug. 1994, Vol. 12, Issue: 6.

[10] B.Stroustrup: *"The C++ Language, third edition"*, Addison-Wesley. 1997.

[11] A.S.Tanembaum: "*Computer networks*". Prentice-Hall 1996.

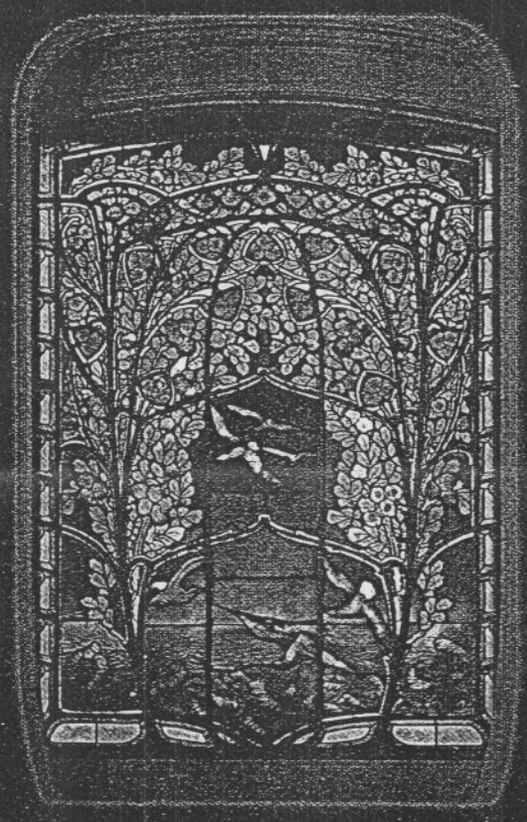
[12] J.Valiño, J.Zarazaga, S.Comella, J.Nogueras, P.Muro-Medrano: *"Utilización de técnicas de programación basadas en frames para incrementar la potencia de representación en clases de C++ para aplicaciones de sistemas de información"*. Proceedings of the CAEPIA'97. Málaga, Spain, Nov, 1997 (Spanish).

[13] S.Yamazaki, K.Kajihara, M.Ito, R.Yasuhara: "Object-Oriented Design of Telecommunication Software". IEEE Software, January 1993.

Proceedings of the TOOLS 29: Technology of Object-Oriented Languages and Systems. 1999, p. 251-260

TOOLS 29

EDITORS: Richard Mitchell, Alan Cameron Wills, Jan Bosch, Bertrand Meyer



TOOLS CONFERENCES

JUNE 7-10, 1999

NANCY, FRANCE

TECHNOLOGY OF OBJECT-ORIENTED LANGUAGES AND SYSTEMS

IEEE COMPUTER SOCIETY