David Rousseau

Suivi par:
M. Thierry Denoeux

UTC
Génie Informatique
Ingénierie des Connaissances et des Systèmes d'Information
Printemps 2008

DATA MANAGEMENT IN A BIOMEDICAL APPLICATION

VICOMTech
Visual Interaction and Communication Technologies Center
Mikeletegi Pasealekua, 57
Parque Tecnológico
E-20009 Donostia - San Sebastián
SPAIN

Responsable: Céline Paloc
cpaloc@vicomtech.org

I wish to thank:

- ✓ Every member of VICOMTech, for their welcome, their integration, their help and their kindness.
- ✓ Céline Paloc, who gives the UTC students like me the opportunity of doing their internship in a place like VICOMTech.
- ✓ Iván Macía, who was taking on his little time to explain, teach me and giving me everything I needed to fulfil my tasks in the best conditions.
- ✓ Iñigo Barandiaran, responsible of my tasks, for his availability each time I needed.
- ✓ The whole team for the nice relationship and atmosphere we work in.

Table of content

List of figures

Technical Summary

During my internship, I was part of a biomedical research and development team developing software for implant surgery planning. It consists in evaluating the possibility of en implant surgery, studying the jaws and teeth structure. The software provides 3D and specific 2D visualisation, with a few tools for analysing the patient mandibles.

This software uses a large amount of date: X-ray image files and study and patient (and more) information. There was no database, nor data structure, nor manager.

First the data structure was designed, and then the database, to keep all the data.

Next, the manager was implemented, with a few functionalities (sorting, selecting, loading, saving).

At last, the different inputs and outputs: importing and exporting (xml format), and creation from the specific medical image data files (DICOM).

In a nutshell, I used and designed a few structures: database, programming objects, XML file, DICOM. I implemented the exchanges between these structures around the user interface. The interface provides functionalities for managing the data.

Presentation - Environment

The international and local groups

VICOMTech is member of two big groups: ik4 and INI-GraphicsNet. Both groups are Research and Development headed in the European economic competition.

Ik4 regroups companies leading the Research and Developments in the new technologies sectors from the Basque country. There have activities in 8 main areas: Mechatronic, Micro-technologies, Industrial management and production, Biotechnology, Information Technology, Energy, Materials and processes and Environment.

INI-GraphicsNet (International Network of Institutions for Advanced Education, Training and R&D in Computer Graphics Technology, Systems and Applications) builds worldwide one of the greatest centres for new Media and forms of communications - including the related of information and application technology.
Present in 7 countries (Germany, Italy, Korea, Portugal, Singapore, Spain, USA), the research of the INI-GraphicsNet focus mainly on missions from research to development of technology and application prototypes. The network cooperates intensely with local universities.

The company

VICOMTech (Visual Communication and Interaction Technologies Centre) is an applied research centre for Interactive Computer Graphics and Multimedia, located in the Technology park of San Sebastian. VICOMTech is a non-profit association, founded by the INI-GraphicsNet Foundation, with the Fraunhofer-IGD as founder member, and the Basque Television, Radio and Broadcasting group EiTB.

VICOMTech does research in the following application areas:

- Digital TV and Multimedia Services
- Biomedical Applications
- Tourism, Heritage and Creativity
- Interaction for Education, Leisure and E-Inclusion
- Industrial Applications

The Biomedical Applications Department, in which I was involved for my internship, carries out research and development for healthcare and biotechnology sectors. It is important to note that VICOMTech is a non profit organisation. This mainly means that VICOMTech is not allowed to commercialize any final product. VICOMTech works for other companies or state organisation.

The team

The head of this department is Céline Paloc (Dr. in Biomedical Engineering).

The work team responsible of the project I worked on is composed of 3 researchers:

- Iván Macía Oliver (Automatic and Electronics Industrial engineer)
- Iñigo Barandiaran Martirena (Computer science engineer)
- Diana Wald (Computer science engineer)

In my work, I was quite autonomous: I was responsible for my “module”, the documentation and conformity with the rest of the project. Of course, I was helped as soon as I needed it, thanks for their availability. The relationships in the team were exempted of hierarchy. We were more colleagues, what was really nice and made the working atmosphere pleasant.

Tools

I was assigned a computer with Window XP, connected to the network and to the internet, apparently without any restrictions. For the development work, the whole team uses the same tools:

- Microsoft Visual Studio .NET 2003.
- Qt, a graphical user interface library.
- Visual ToolKit (VTK) and Insight ToolKit, two libraries used for medical images processing.
- CMake, a cross-platform system for build automation, including all the different necessary libraries easily.
- Tortoise SVN, a Version Control System that we all used together for our project.
- PostgreSQL local server for the implementation of the database, part of my internship.

The project

For confidential reasons, I cannot give the real names of the clients, the project or the software. We will call it the MedApp, for medical application.

MedApp is a software application that digitalizes and displays patients´ CAT scan (computerized axial tomography), making implant surgeries much easier.

MedApp project is an application for doctors performing implant studies. It allows the doctor to visualize the jaws of the patient in a configurable in 3D view, evaluate and simulate positioning a future implant.

The clients asked for a complete prototype, ready to be commercialized. This is one of the biggest current projects in VICOMTech: the conception includes the research, the development and the documentation such as functional specification, developers guide, security and user manual.

The requirements

Actually, the project isn’t really new: such software already existed. But this software was basic and, as it was developed by sub-contactors, the code wasn’t available.
The clients asked VICOMTech to do research on the different possibilities with new technologies, and when they saw the results, they asked for the complete product.

The users are doctors who are not very experienced with computers and new software, especially 3D visualisation software. As they have trained on the old version, the new version has to be very similar to the old one, so the doctors wouldn’t feel lost, even if containing more and better features.

So the new software interface has to be very close to the old one, but in the same time, include the new client requirements. And as they know much more about the medical side than the computer side, the requirements were very practical and functional.

I found it very interesting to have direct requirements from the “potential users”, so implementing them was sometimes quite a challenge.

My work

When I arrived, the basic jaws visualisation was done, and the first functionalities being implemented. My work consisted in building the **study manager**: The **study** to be edited by the doctors is first chosen among all the others, and is loaded, and later saved again. They can also be exported, imported, created. So I was to build all this.

The MedApp software is actually constituted of two big parts: the **editor manager**, providing all the tools for the doctor to work on the implant surgery (=**study**), and the **study manager**, providing all the studies management. I only worked on the **study manager**.

If we follow one **study**: First it’s created from the radiologists images, getting the patient information, and saved. Then, the **study** is chosen (among all the others in the list) and loaded, edited and saved again. This can repeat as many times as necessary. Then at last, it is exported when the doctor is finished with it. Note: an exported **study** can be imported again.
And all this is done around the **study manager**.

The study manager, the creation module, the import and export module, the saving and loading process, was what I worked on.

My organisation chart

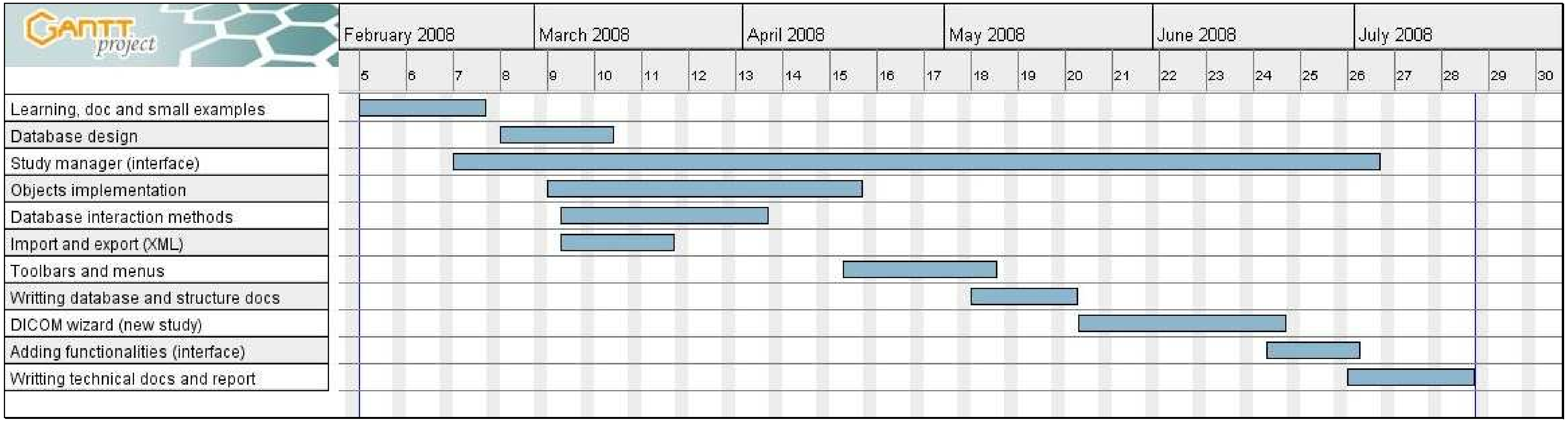


Figure 1 - Organisation chart
Figure 1 – Organigramme du projet

The application

I. Structuring the data

a) Description

At the beginning of my internship, the team had already set up the visualisation of the mandibles and setting up the first tools (editor).

Actually the visualisation is build from medical DICOM files (Digital Imaging and Communication in Medicine) that are sets of medical X-ray images. So the 3D representation of a DICOM belongs to a study, which is engaged for one patient for one series of implants.

These studies had to be precisely defined and all other kind data needed by the doctors in the study (human side) or for the edition software to work (technical side).

Technically a study is composed, on the one side, of a set of X-ray image files and, on the other side, information concerning the study (incl. patient, doctor, clinic) and every internal study configuration (i.e. implant positioning). I specifically worked on the study information.

And this step had not been done until then, so I spend a long time evaluating the needs, organising the data and discussing the issues. So that was my first objective: build the database.

b) Designing the database

The MedApp uses a lot of information; most of this information has to be organised, sorted, kept temporally, saved or even exported. The database has been build as described below.
(Each ***"keyword"*** represents a table)

The software creates a ***Study***.
It is made to evaluate the possibility of having ***Implants*** added to one ***Patient***.
An ***Implant*** is of a specific implant ***Type***.
Each implant ***Type*** is part of a specific implant ***Family***.
Each implant ***Family*** is from a specific implant ***Vendor***.
A radiology ***Clinic*** takes care of taking the radio pictures (DICOM), which are required for a ***Study***. A ***Study*** is always supervised by a ***Doctor***.
In a Study, the user builds an ***Arcade*** and needs to save it among his data.
This ***Arcade*** is defined by eight control points (in 3D) in order (***ControlPoint***).

The team chose to implement a PostgreSQL database, because of it is light, easy to implement, to use, and free to use in our commercial situation.

c) The database schema

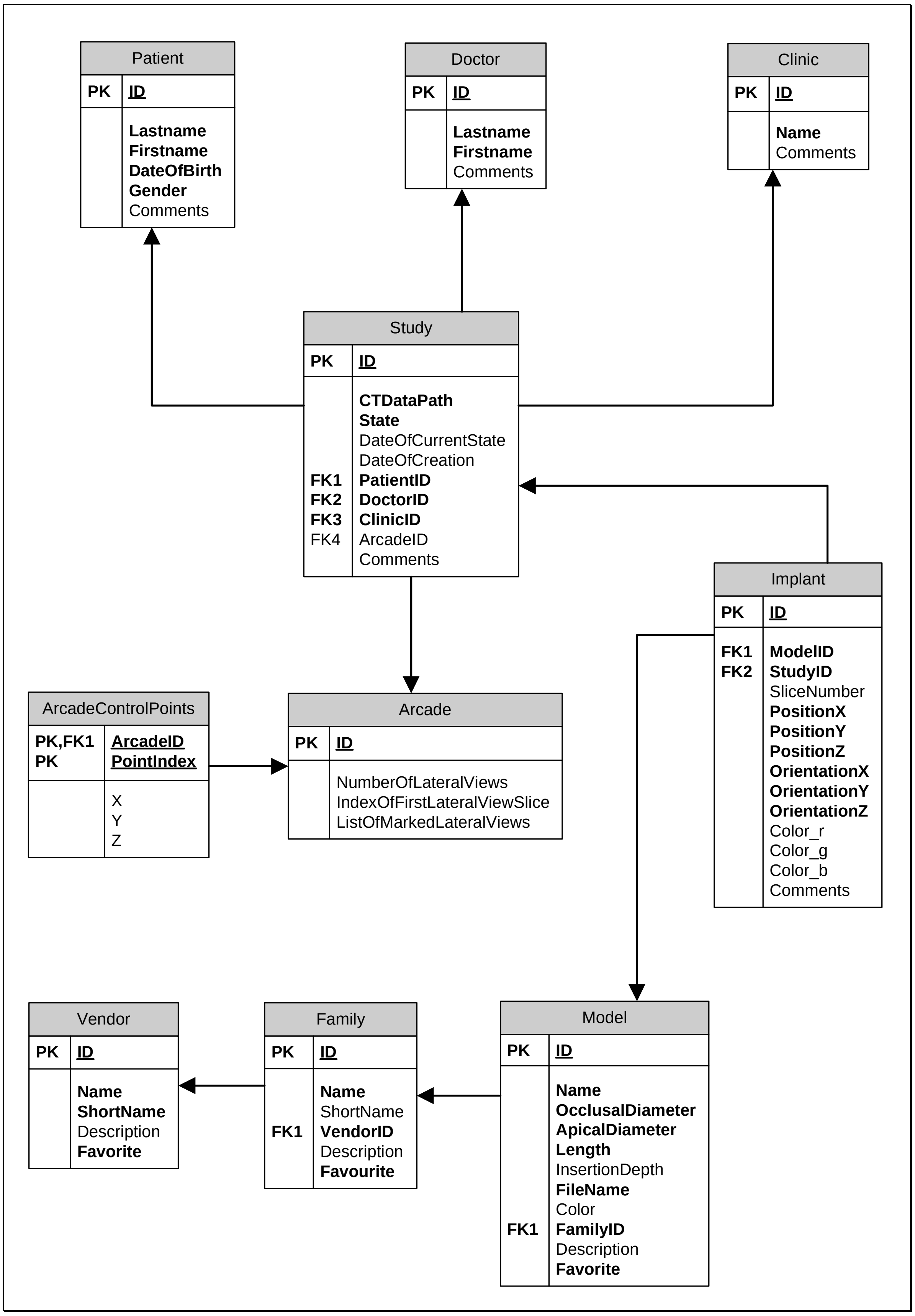


Figure 2 - Database schema
Figure 2 – Schéma de la Base de données

II. Implementing the application

a) Functional description

Once the database is designed, we come to the representation and the use of this data.
First of all, the chosen study is selected among the other studies of the database, so they must be shown in a view that allows the user to make the difference between them and identify the one he wants easily.

Then he needs to perform different actions on it (them), such as adding a new study, edit a study. The study manager has been developed for this purpose. The study manager is the part of the program responsible of representing the content of the database, allow every user action and send the selected study to the Editor visual tools. The Editor contains all the functions that the doctor can work with, and is developed by the rest of the team.

Saving and loading a study are obviously required.

Moreover, an export function is required, so the finished studies can be archived, to empty regularly the database and provide a way of transferring and storing the studies externally. And in the same time, an importation shall reinsert an exported study in the database.

As the requirements are extensive, I’ll only present the major functionalities and windows.

b) Study creation (DICOM)

Although this functionality was among the last ones I implemented in my internship, I will present it now for a better comprehension of the raw medical data. This is done when a patient just has had his X-ray exam and the doctor recommends an implant surgery. The doctor loads the DICOM (Digital Imaging and Communication in Medicine, which are the X-ray image files) and starts a new study with it.

On the technical side, all the information from the DICOM has to be translated to our standard data structure. We use our object structure for a start, and then save it to our database structure. The images are extracted and saved to a specific local directory. Although the DICOM format is a known format in medical applications, it is tough to read it because it is a (messy) mix of textual and visual data. One series of images belongs to a single patient, and to a single X-ray exam, but one DICOM can contain several series from supposedly the same patient
That’s why the doctor has to choose a series within a DICOM within a directory. (There are three levels of containers).

For an easy way to import a DICOM series, we build a wizard for assisting the doctor in every step. Here are the main pages of it:

First step: Choosing the directory. The MedApp browser checks the presence of DICOM files in the selected directory.

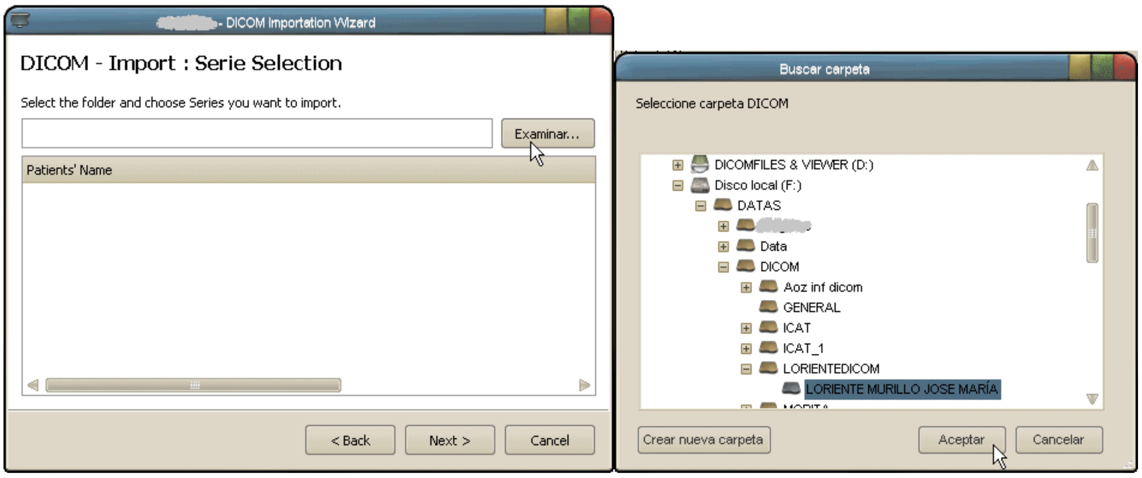
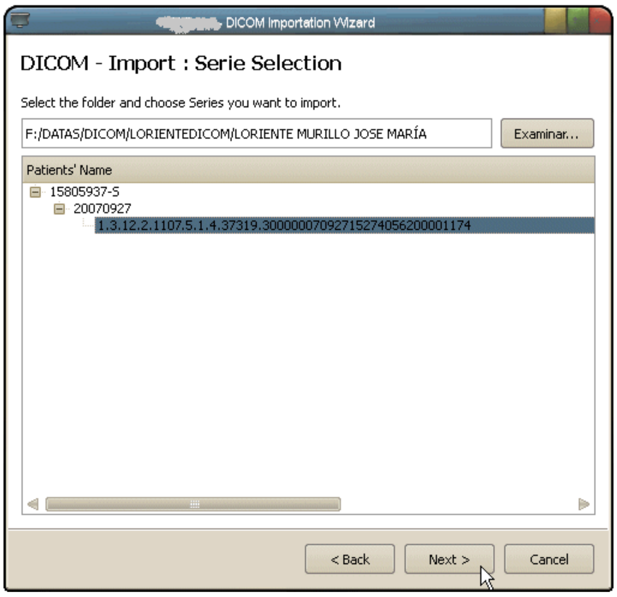


Figure 3 and 4 - Select directory
Figure 3 et 4 –Choix du répertoire

Second step: choosing the series. The whole directory and subdirectories are scanned, and every series founds, as a hierarchical tree (“DICOM” >> “patient” >> “series”).


[redacted] DICOM Importation Wizard

DICOM - Import : Serie Selection

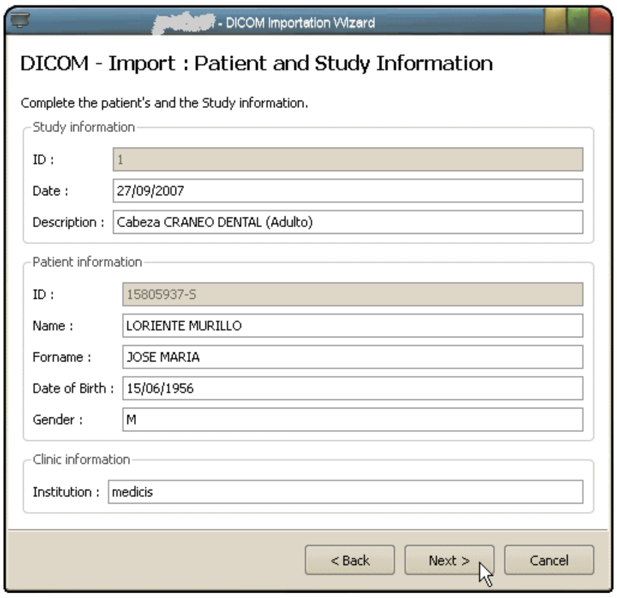
Select the folder and choose Series you want to import.

F:/DATAS/DICOM/LORIENTEDICOM/LORIENTE MURILLO JOSE MARÍA

Patients' Name
- 15805937-S
  - 20070927
    - 1.3.12.2.1107.5.1.4.37319.30000007092715274056200001174

Figure 5 - Select series
Figure 5 – Choix de la série

Third step: editing the information before creating the data structure with it. It allows the user to correct or complete the DICOM information. It validates all the fields for conformity and insures the study integrity.


[redacted] - DICOM Importation Wizard

DICOM - Import : Patient and Study Information

Complete the patient's and the Study information.

Study information

ID :	1
Date :	27/09/2007
Description :	Cabeza CRANEO DENTAL (Adulto)

Patient information

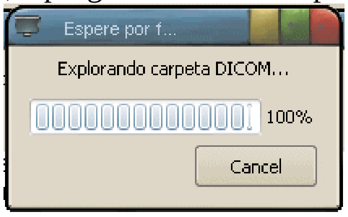
ID :	15805937-S
Name :	LORIENTE MURILLO
Forname :	JOSE MARIA
Date of Birth :	15/06/1956
Gender :	M

Clinic information

Institution :	medicis

Figure 6 - Editing information
Figure 6 – Confirmation des informations

The wizard also includes a start page and a completion page. As loading and reading of large volume of medical files could take some time, a progress bar shows up during processing.


Espere por f...

Explorando carpeta DICOM...

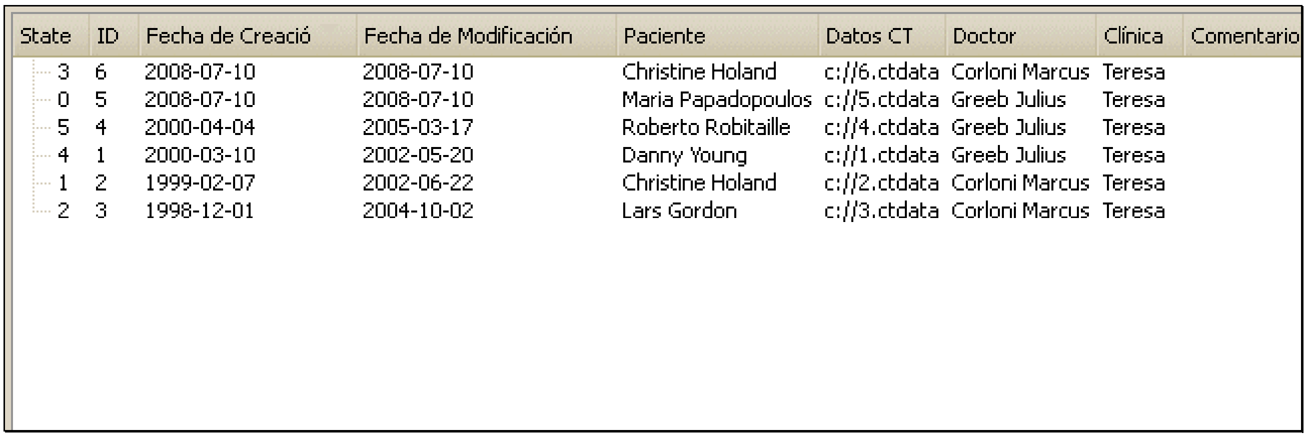
100%

Figure 7 - Progress bar
Figure 7 – Barre de progression

c) Displaying the content of the database.

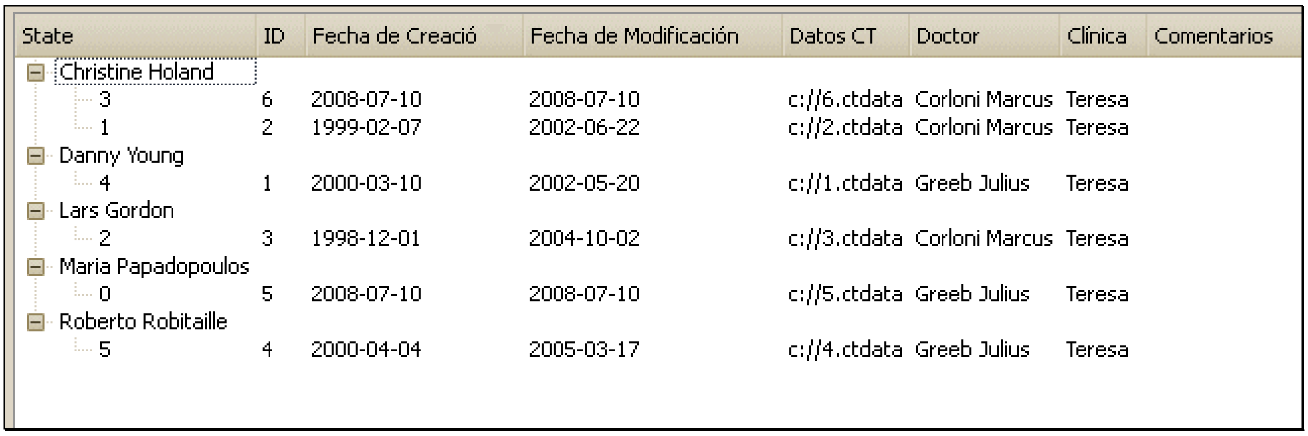
We had to follow the user interface of the old MedApp, but in the same time, we wanted to improve visual aspect and performance. So different solutions were imagined, tried, and then we discussed them in team meeting, and we decided together what the best solution was.

Among the different views, we tried lists, tables, trees, but we found out best was the simplest and the easiest. Here are the two most pertinent views:



State	ID	Fecha de Creació	Fecha de Modificación	Paciente	Datos CT	Doctor	Clínica	Comentario
3	6	2008-07-10	2008-07-10	Christine Holand	c://6.ctdata	Corloni Marcus	Teresa	
0	5	2008-07-10	2008-07-10	Maria Papadopoulos	c://5.ctdata	Greeb Julius	Teresa	
5	4	2000-04-04	2005-03-17	Roberto Robitaille	c://4.ctdata	Greeb Julius	Teresa	
4	1	2000-03-10	2002-05-20	Danny Young	c://1.ctdata	Greeb Julius	Teresa	
1	2	1999-02-07	2002-06-22	Christine Holand	c://2.ctdata	Corloni Marcus	Teresa	
2	3	1998-12-01	2004-10-02	Lars Gordon	c://3.ctdata	Corloni Marcus	Teresa	

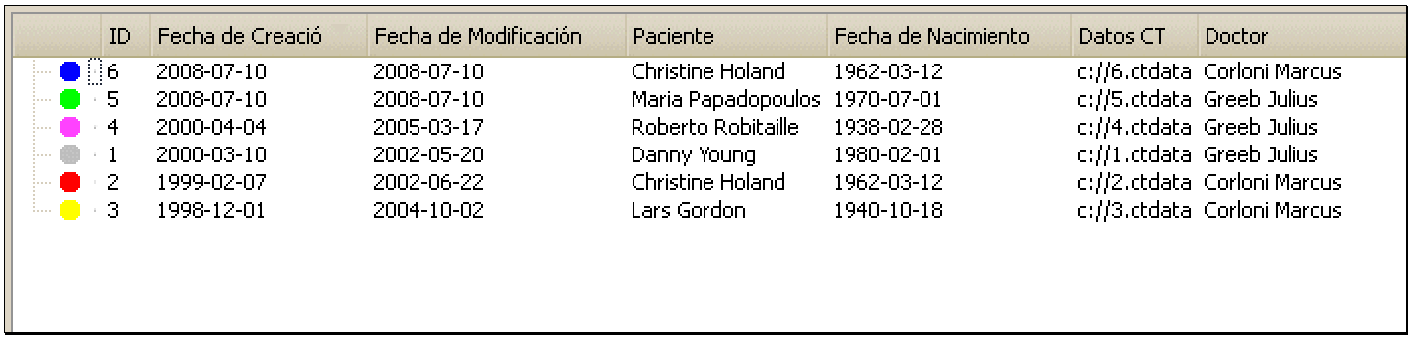
Figure 8 - Table view
Figure 8 – Vue en tableau



State	ID	Fecha de Creació	Fecha de Modificación	Datos CT	Doctor	Clínica	Comentarios
⊟ Christine Holand							
3	6	2008-07-10	2008-07-10	c://6.ctdata	Corloni Marcus	Teresa	
1	2	1999-02-07	2002-06-22	c://2.ctdata	Corloni Marcus	Teresa	
⊟ Danny Young							
4	1	2000-03-10	2002-05-20	c://1.ctdata	Greeb Julius	Teresa	
⊟ Lars Gordon							
2	3	1998-12-01	2004-10-02	c://3.ctdata	Corloni Marcus	Teresa	
⊟ Maria Papadopoulos							
0	5	2008-07-10	2008-07-10	c://5.ctdata	Greeb Julius	Teresa	
⊟ Roberto Robitaille							
5	4	2000-04-04	2005-03-17	c://4.ctdata	Greeb Julius	Teresa	

Figure 9 - Tree view
Figure 9 – Vue en arborescence

We finally chose the table view. Here is our optimised version (each colour represents a study status):



	ID	Fecha de Creació	Fecha de Modificación	Paciente	Fecha de Nacimiento	Datos CT	Doctor
🔵	6	2008-07-10	2008-07-10	Christine Holand	1962-03-12	c://6.ctdata	Corloni Marcus
🟢	5	2008-07-10	2008-07-10	Maria Papadopoulos	1970-07-01	c://5.ctdata	Greeb Julius
🟣	4	2000-04-04	2005-03-17	Roberto Robitaille	1938-02-28	c://4.ctdata	Greeb Julius
⚪	1	2000-03-10	2002-05-20	Danny Young	1980-02-01	c://1.ctdata	Greeb Julius
🔴	2	1999-02-07	2002-06-22	Christine Holand	1962-03-12	c://2.ctdata	Corloni Marcus
🟡	3	1998-12-01	2004-10-02	Lars Gordon	1940-10-18	c://3.ctdata	Corloni Marcus

Figure 10 - Optimised table view
Figure 10 – Vue optimisée en tableau

d) Creating objects

The objective is to provide objects to manage the data in object oriented programming, during the doctor's work on the studies in: these objects will contain the data that will be modified in the editor. This means building a class (object) structure out of the different database tables.
The schema of these objects is quite close to the database schema, except we only need one instance of the Study, the Doctor, the Clinic, and the Arcade. That makes things a little different on the relations between objects, but otherwise it is mostly the same structure.

Here is a quick schema of this structure:

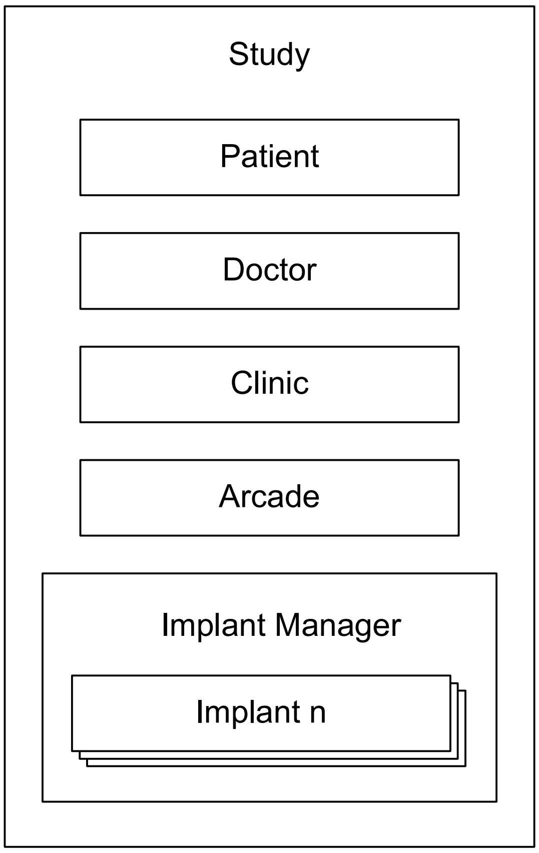


Figure 11 - Object structure schema
Figure 11 – Schéma de la structure objet

As we can see on the schema:

- A study object contains one patient, one doctor, one clinic, one arcade and one implant manager object.
- The implant manager object contains all the implant objects that belong to the study.

The study is the main object we manipulate. All actions on a study (Load, Save, Import, Export) are to be performed by the study itself (internal methods). The study, as a container, has the responsibility of distributing the actions to its components.

e) Methods to load and save the data.

Basically, loading a study is building an instance of the object structure by reading the corresponding information in the database tables. Saving is the opposite, updating the database tables from the information of the object structure. The interaction with the database is achieved with a specially designed database manager. It handles all requests to the database. Each object is responsible for its own loading and saving.

Loading a study is made simple because the entire study object's data is retrieved in a single elaborated database request. (See Annexe 1)

Here is a sequence diagram of the method we chose to load a study:

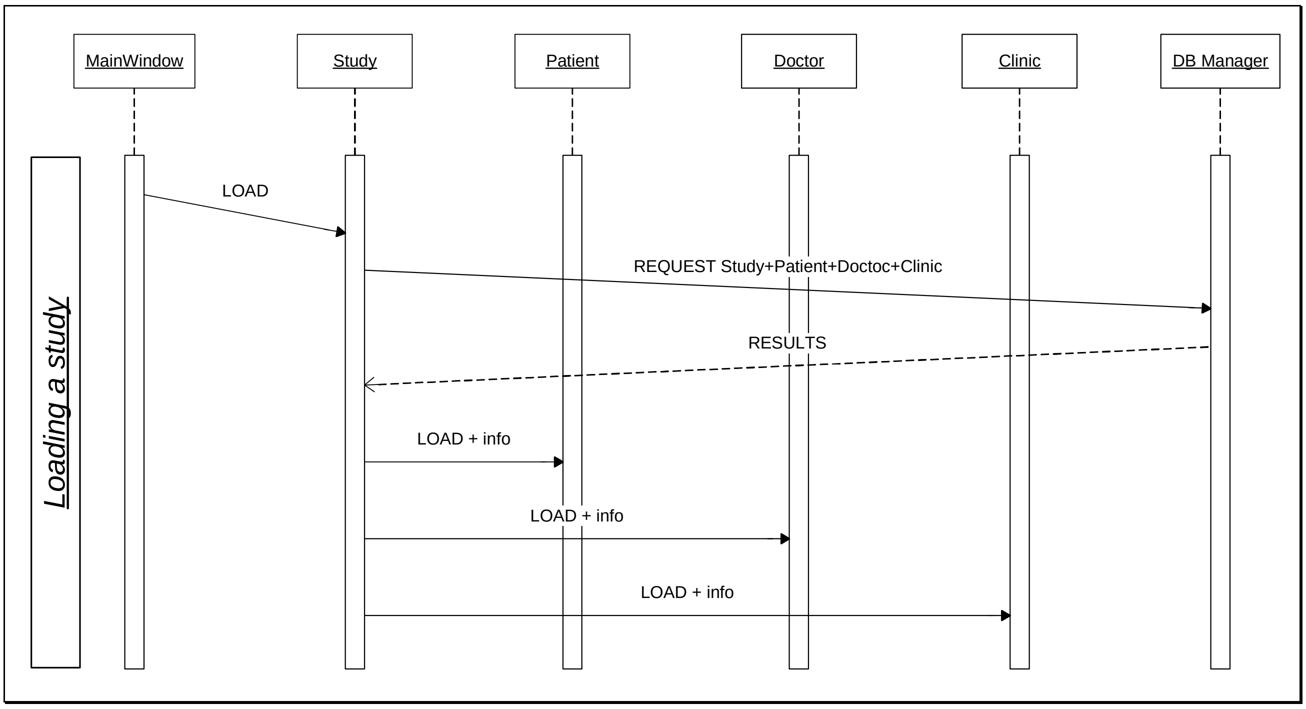


Figure 12 - Simplified loading sequence
Figure 12 – Sequence de chargement simplifiée

Remark: The diagram has been simplified: implants, arcade are not mentioned.
Remarque: Le diagramme ne mentionne pas les implants et l'arcade.

Saving is more complicated, mainly for safety reasons. The data to be saved has to be complete, up to date, valid and accessible. For example, we have to check whether the object being saved has already been saved or not. We can't save a study without its patient. On a technical side, the objects and the table aren't managed the same way. We need to get the database identifier of what we inserted for verification and because it is required for the next insertion, for table interconnection.

Here is the sequence diagram of the saving method:

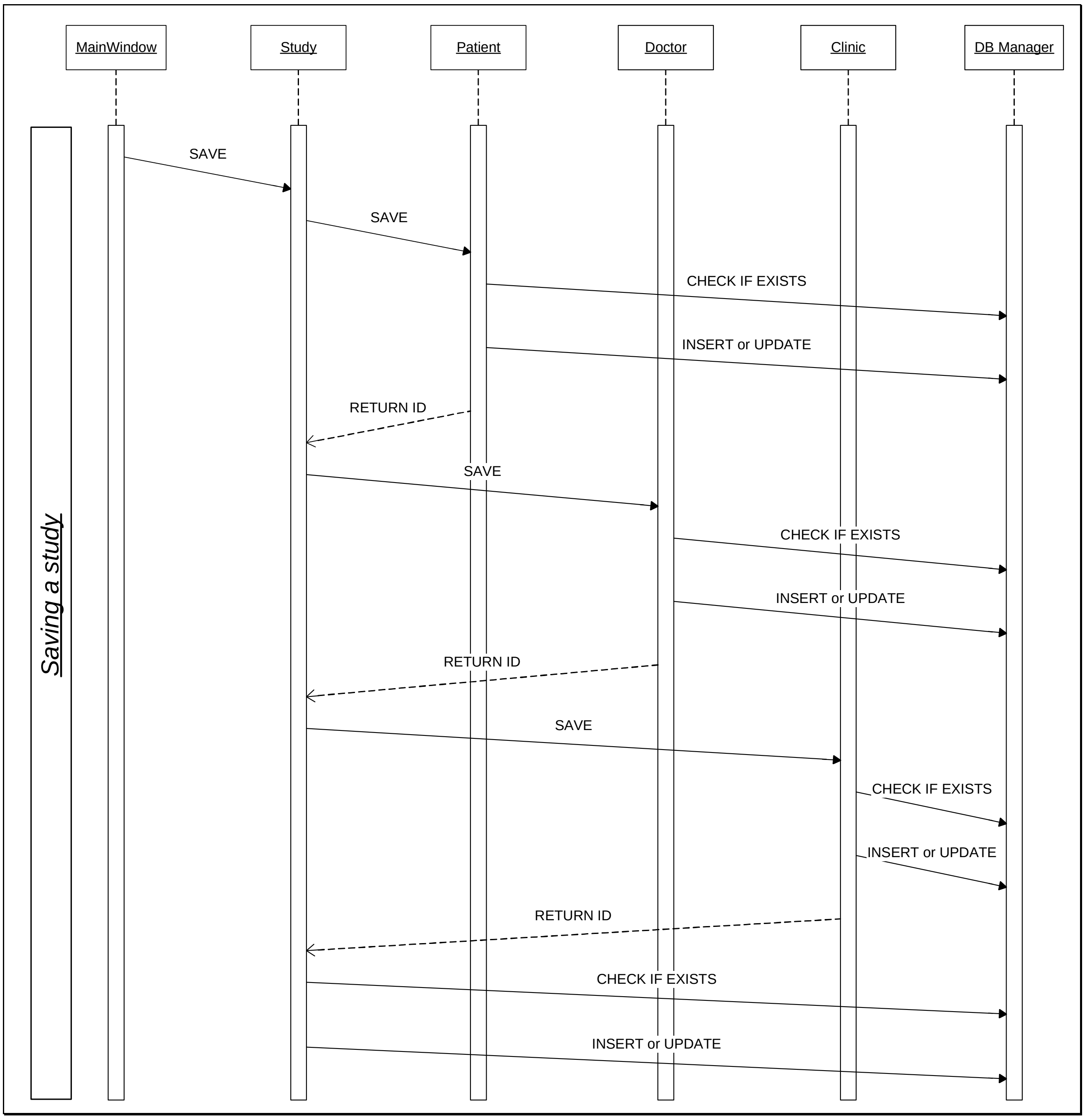


Figure 13 - Simplified saving sequence
Figure 13 – Sequence de sauvegarde simplifiée

Remark: The diagram has been simplified: implants, arcade are not mentioned.
Remarque: Le diagramme ne mentionne pas les implants et l'arcade.

f) Menus, toolbars and other functionalities.

The menus and the toolbars provide the actions to be executed, one in a windows traditional way (menus) and the other like quick access to the main actions directly above the study listing. The whole interface has been (re)built imitating the old disposition in an improved version.

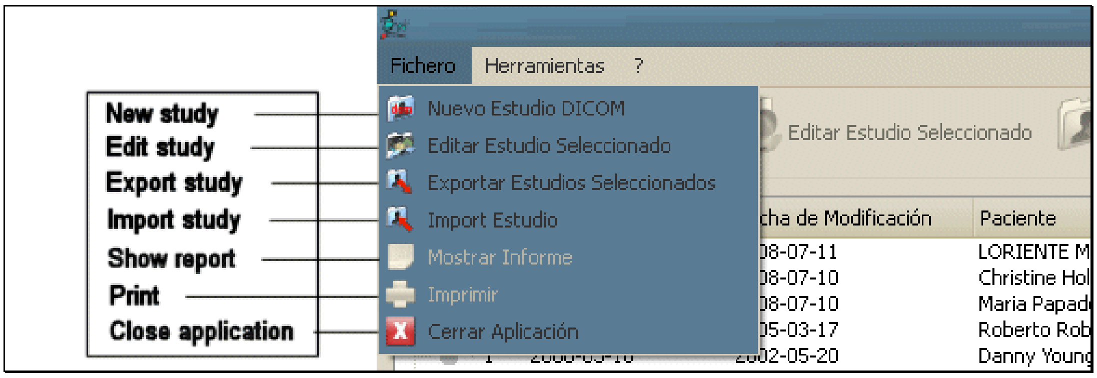
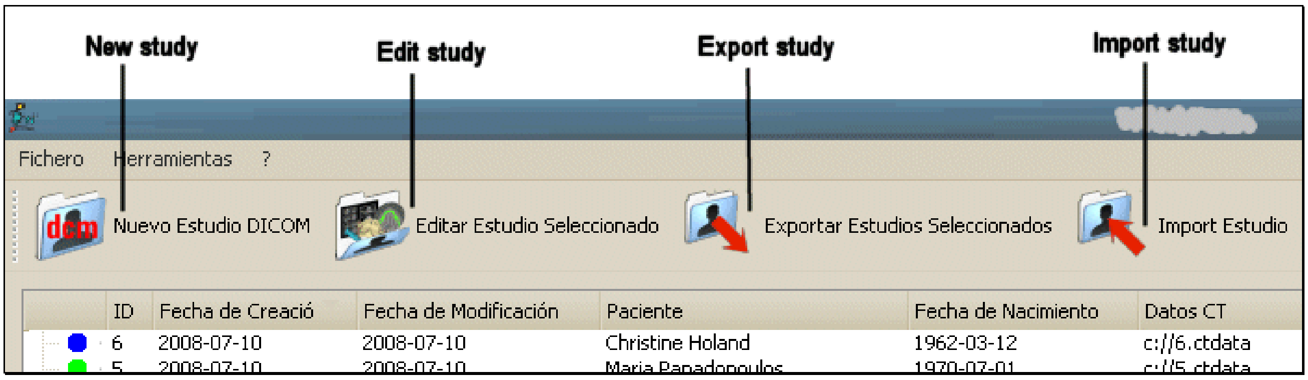


Figure 14 – Menus
Figure 14 – Les menus



	ID	Fecha de Creació	Fecha de Modificación	Paciente	Fecha de Nacimiento	Datos CT
🔵	6	2008-07-10	2008-07-10	Christine Holand	1962-03-12	c://6.ctdata

Figure 15 – Toolbars
Figure 15 – Les barres d’outils

The status of the different studies is to be changed by the doctors. Archived (but not deleted) studies are to be hidden (or not).

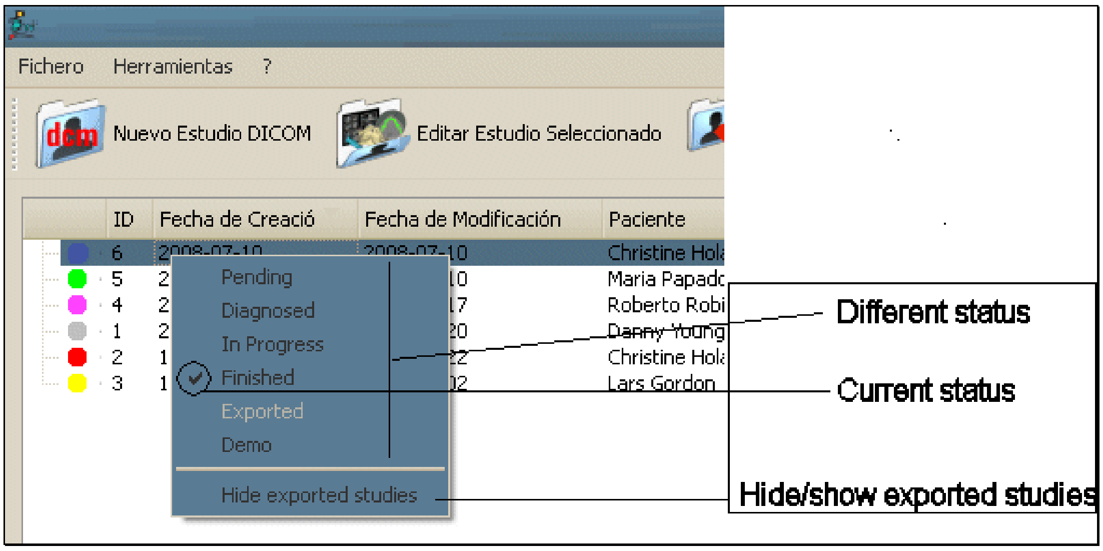


Figure 16 - Mouse right click menu
Figure 16 – Le menu click-droit de la souris

g) Study import and export

That part was particularly interesting because requirements only mentioned to use XML format. We made all the other choices. To parse the XML file, we chose SAX file access, because the file is read once, and an event reading is very appropriate.
Importing and exporting from an XML file is, actually, converting from one format to the other; as XML is a convenient structured description language. So the XML reflects the programming object structure, containing a study, a patient, a doctor, a clinic, an arcade and implants.

Here is a schema to explain the hierarchy used in the file:

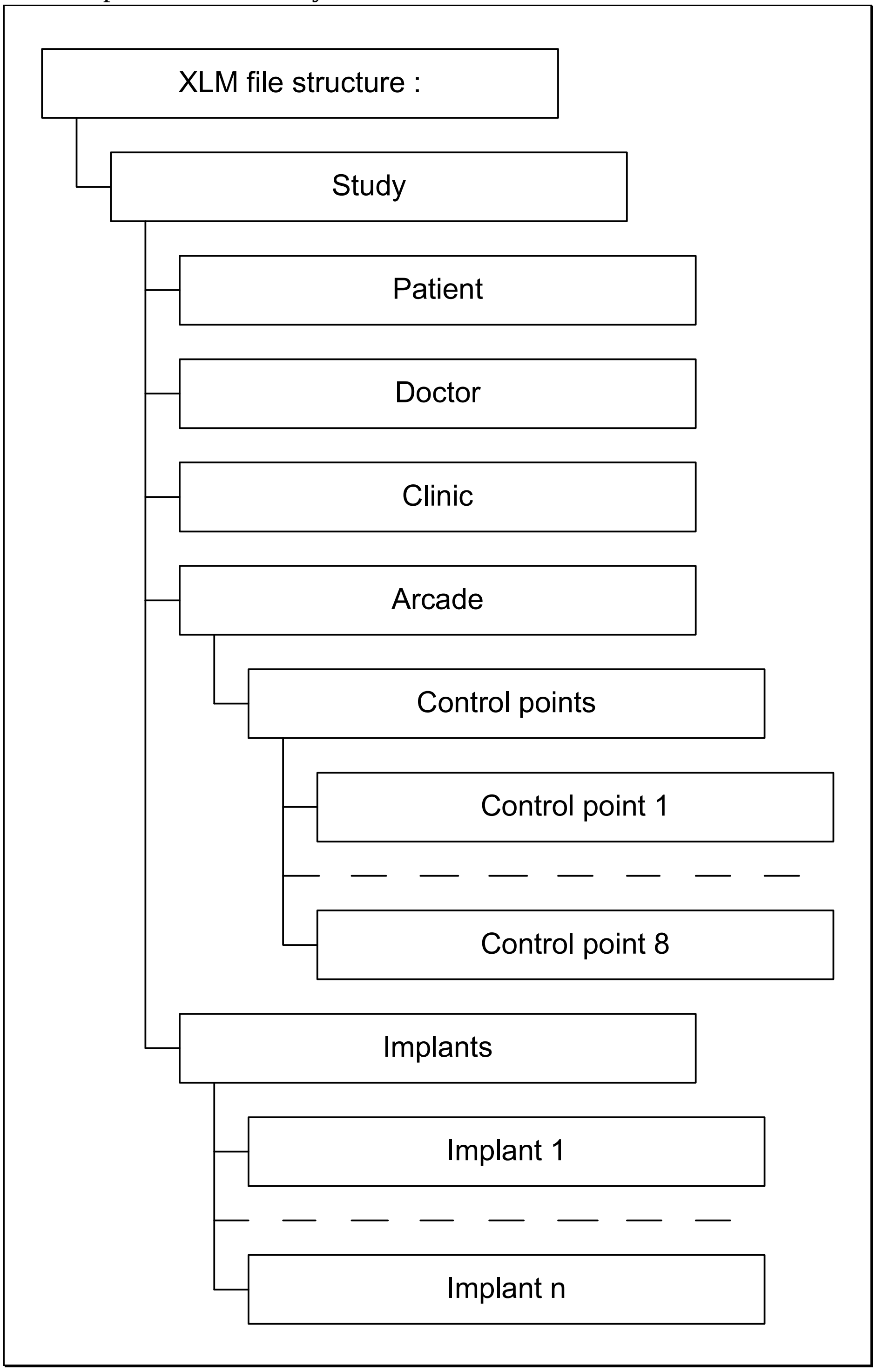


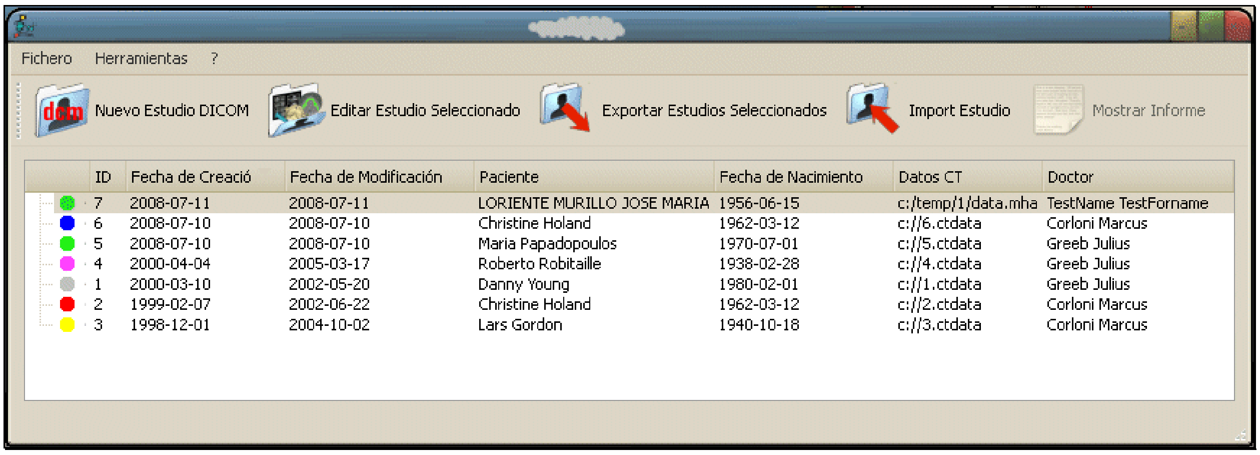
Figure 17 - XML schema structure
Figure 17 – Schéma de la structure XML

Once the study objects are made “serializable”, they all become capable and responsible of parsing XML themselves, and grab their own information.

Exporting a study, means to save all the database information to the XML file and moving its CTData (medical images) to a user-specified directory. The idea is simple: each object writes its data, in a special order and structure.

When we import a study, its components are created, one after the other, and populated from the XML file. After some validity and integrity checks, it is saved to the database.

h) Generally



	ID	Fecha de Creació	Fecha de Modificación	Paciente	Fecha de Nacimiento	Datos CT	Doctor
●	7	2008-07-11	2008-07-11	LORIENTE MURILLO JOSE MARIA	1956-06-15	c:/temp/1/data.mha	TestName TestForname
●	6	2008-07-10	2008-07-10	Christine Holand	1962-03-12	c://6.ctdata	Corloni Marcus
●	5	2008-07-10	2008-07-10	Maria Papadopoulos	1970-07-01	c://5.ctdata	Greeb Julius
●	4	2000-04-04	2005-03-17	Roberto Robitaille	1938-02-28	c://4.ctdata	Greeb Julius
●	1	2000-03-10	2002-05-20	Danny Young	1980-02-01	c://1.ctdata	Greeb Julius
●	2	1999-02-07	2002-06-22	Christine Holand	1962-03-12	c://2.ctdata	Corloni Marcus
●	3	1998-12-01	2004-10-02	Lars Gordon	1940-10-18	c://3.ctdata	Corloni Marcus

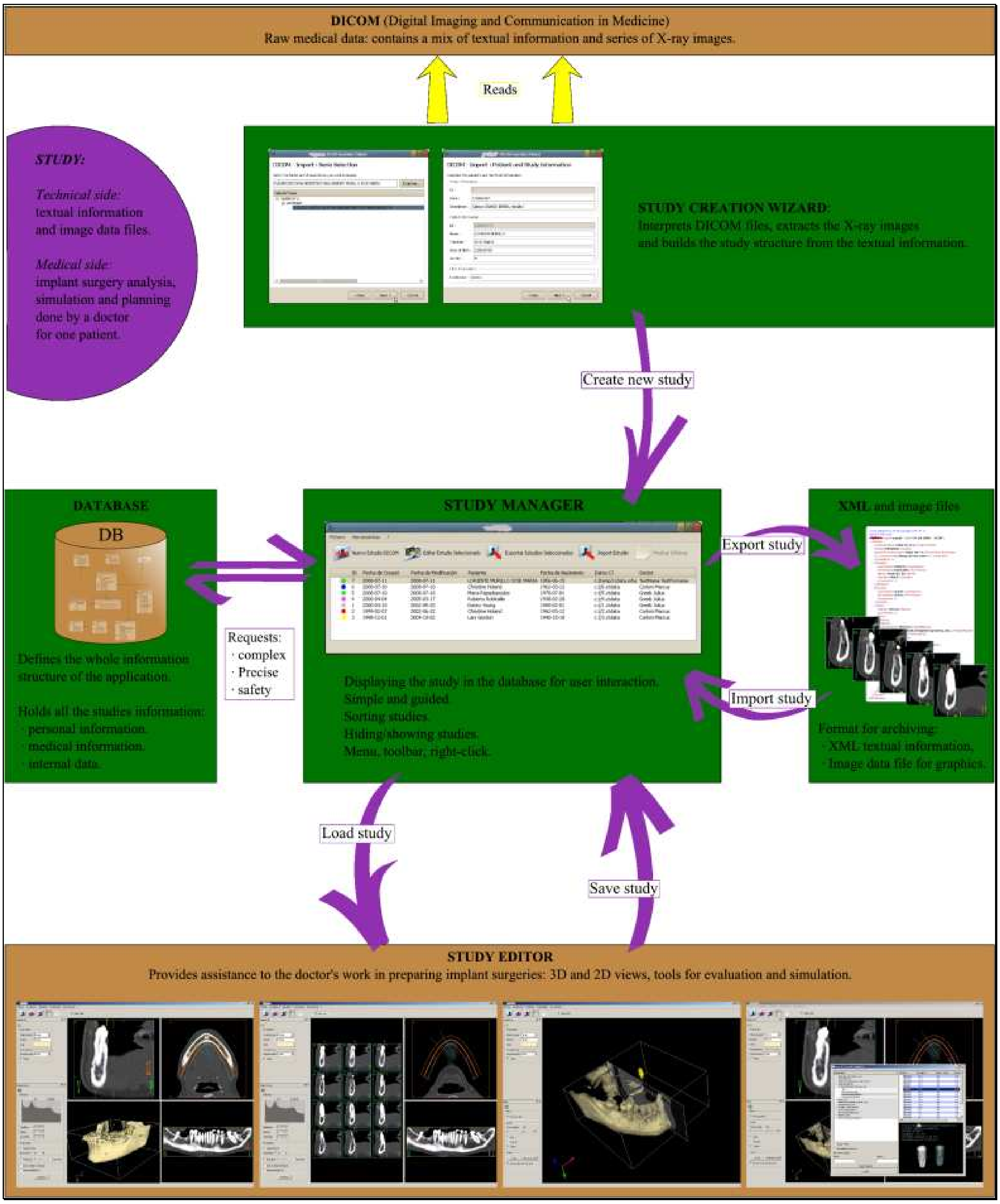
Figure 18 - General view
Figure 18 – Vue générale

This is what the prototype looks like, for the moment.

It shows up at the application start as the main application window. The user will choose one or more studies to perform an action. Although this isn’t the part of the application that required the most work, it does appear to be in the most important one.

Conclusion

In a nutshell



DICOM (Digital Imaging and Communication in Medicine)
Raw medical data: contains a mix of textual information and series of X-ray images.
Reads
STUDY:
Technical side: textual information and image data files.
Medical side: implant surgery analysis, simulation and planning done by a doctor for one patient.
STUDY CREATION WIZARD:
Interprets DICOM files, extracts the X-ray images and builds the study structure from the textual information.
Create new study
DATABASE
DB
Defines the whole information structure of the application.
Holds all the studies information:
· personal information.
· medical information.
· internal data.
Requests:
· complex
· Precise
· safety
STUDY MANAGER
Displaying the study in the database for user interaction.
Simple and guided.
Sorting studies.
Hiding/showing studies.
Menu, toolbar, right-click.
Export study
Import study
XML and image files
Format for archiving:
· XML textual information.
· Image data file for graphics.
Load study
Save study
STUDY EDITOR
Provides assistance to the doctor's work in preparing implant surgeries: 3D and 2D views, tools for evaluation and simulation.

Figure 19 – Synthesis
Figure 19 – Synthèse

In my internship, I participated in the development of medical application for dental surgery. I was responsible of designing the database, the main window for study management and the input/output modules. In fact, I worked on medical data structures, and implemented methods to convert one into the other. DICOM to objects, objects to database, database to objects, objects to XML, XML to database; creating, saving, loading, exporting, and importing studies.
Not only the structures but also their user interfaces, for a neat and user-friendly GUI.

The work was really appreciated by the team and by the clients. The goals are reached: the similarity to the old interface and major user-friendly improvements. (The team did think the old software wasn't convenient and nice, and moreover it had some bugs.) On the functional side, there are still a few missing elements that prevented the complete integration and functioning.

Bottom line

My internship was really among the best I could have imagined.
I had the opportunity to work in a young spirited company, working on innovative projects, using new technology. I loved the fact the relationships were friendly, providing a nice working environment. I also loved the teamwork: working on the same project, with each our responsibilities, and helping out each other, and taking decisions together in our weekly team meeting.
I really love this atmosphere, which sometimes brought us out to restaurant together after a department meeting, for example.

To be in a biomedical department was also a great experience, joining technical programming, with special data, to a very practical use. I think it is grateful to be able to visualise graphically the results of our work. It is a kind of thrilling and useful challenge to provide tools we know to be humanly good and helpful.

Annexe 1: Request for loading a study

```
// SQL Request :   ***************
      QString mySqlRequest = tr(
            "SELECT "
                  "Study.ID, "
                  "Study.State, "
                  "DATE(Study.DateOfCreation) AS DOC, "
                  "DATE(Study.DateOfCurrentState) AS DOCS, "
                  "Study.CTDataPath, "
                  "Study.Comments, "
                  "Patient.ID, "
                  "Patient.Lastname, "
                  "Patient.Firstname, "
                  "DATE(Patient.DateOfBirth) AS DOB, "
                  "Patient.Gender, "
                  "Patient.Comments, "
                  "Doctor.ID, "
                  "Doctor.LastName, "
                  "Doctor.Firstname, "
                  "Clinic.ID, "
                  "Clinic.Name"
            "FROM "
                  "Study, "
                  "Patient, "
                  "Doctor, "
                  "Clinic "
            "WHERE ("
                  "Study.PatientID=Patient.ID "
            "AND "
                  "Study.DoctorID=Doctor.ID "
            "AND "
                  "Study.ClinicID=Clinic.ID ) "
                  "%1"
            "ORDER BY "
                        "Patient.Lastname, "
                        "Patient.Firstname, "
                        "Study.ID;"
      )
        .arg(
               ( m_TreeWidget->GetHiddenExportedStudies() )?
                        "AND Study.State <> 'exported' "
                        :""
         );

// End of SQL request ***************
```

This request is used to get all the information at the same time, when loading an existing study. It's just an example, to show how a good request can save a lot of programming.

Annexe 2: Functional modules

This schema has been designed among one of our team meeting for building the plan of our documentation. Even if it is one of our first versions, it is interesting to see all the different modules and sub-modules that constitute internally our software. (Not everything has been implemented)

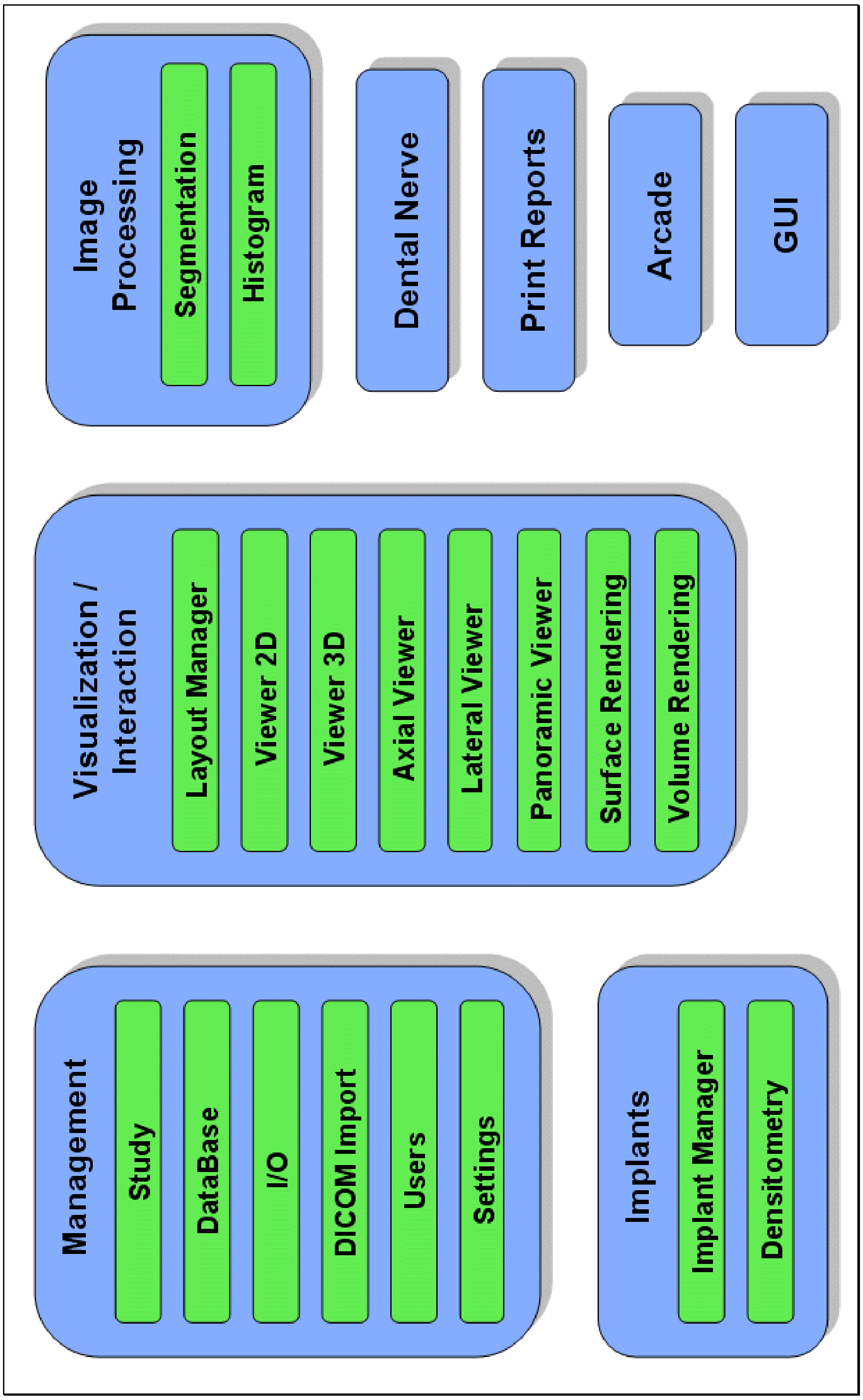


Figure 20 - Functional modules
Figure 20 – Modules fonctionnelles

Annexe 3: The study class description

```
/** \class Study
 *  \brief Stores information related to Studies.
 *
 * This class represents a dental clinic study.  It stores information such as CT-images, Patient, Clinic,
 * Doctor, or planned Implant objects, Arcade model and DentalNerve.
 *
 * Studies can be in 6 different states:
 * - Pending: means that the Study is created in the DataBase, but there is no current planning.
 * - Diagnosed: means that there is a planning going on.
 * - In Progress: means that the study is in an intermediate state.
 * - Finished: means that the planning and intervention finished and the Study can be exported.
 * - Exported: means that the study have been exported but not deleted.
 * - Demo: means that the study is used for demonstration.
 *
 * Study objects can be loaded/saved from/to the DataBase. Studies are stored in the DataBase when they
 * are first imported (from file or DICOM) and removed from the DataBase when they are exported to file.
 *
 * Study objects can also be read/written to file in an XML-based *** proprietary format. This can be used
 * as a way of interchange of studies from the radiologist to the dentist and also as a way of exporting
 * finished studies.
 *
 * Information related to the current Implant objects in the current Study is stored in an ImplantManager
 * contained in this object.
 *
 * DEVELOPER NOTES
 *
 * This class and objects contained in this class inherit from XMLStreamObject in order to implement its
 * serialization in XML format.
 *
 * DESCRIPTION OF OPERATIONS
 *
 * - Load/Save current study to the DataBase.
 * - Import/Export studies to/from external files in BTI XML-based proprietary format.
 * - Set/Get Clinic information.
 * - Set/Get Patient information.
 * - Set/Get Doctor information.
 * - Set/Get Study State. Available states are Pending, Diagnosed and Finished.
 * - Set/Get Study Date of Creation.
 * - Set/Get CT image data.
 * - Add/Delete Implant to/from Study.
 * - Set/Get Arcade model.
 * - Set/Get DentalNerve model.
 * - Delete study from the database.
 *
 * DATA ACCESS
 *
 * - Get Study Data from the Study Table of the DataBase or from external files.
 *
 * ERROR HANDLING
 *
 * - Error handling is solved by queries and standard output.
 *
 * SEE ALSO
 *
 * - Clinic
 * - Doctor
 * - Patient
 * - Arcade
 * - DentalNerve
 * - Implant
 * - ImplantManager
 *
 */
```

As I haven't spoken of coding standards, here I present an example of the class description format we used in the code, just above the class declaration (*.h files), for a developer side documentation. These comments are conveniently interpreted by Doxygen to generate our developers guide.