BIMS: Biomedical Information Management System

Oscar Mora Pérez

Projecte Final de Carrera

Director: Jesús Bisbal

Juliol 2009

Enginyeria Informàtica

Universitat Pompeu Fabra

A la meva família, pel seu suport incondicional

Abstract

This final year project presents the design principles and prototype implementation of BIMS (Biomedical Information Management System), a flexible software system which provides an infrastructure to manage all information required by biomedical research projects.

The BIMS project was initiated with the motivation to solve several limitations in medical data acquisition of some research projects, in which Universitat Pompeu Fabra takes part. These limitations, based on the lack of control mechanisms to constraint information submitted by clinicians, impact on the data quality, decreasing it.

BIMS can easily be adapted to manage information of a wide variety of clinical studies, not being limited to a given clinical specialty. The software can manage both, textual information, like clinical data (measurements, demographics, diagnostics, etc ...), as well as several kinds of medical images (magnetic resonance imaging, computed tomography, etc ...). Moreover, BIMS provides a web - based graphical user interface and is designed to be deployed in a distributed and multiuser environment. It is built on top of open source software products and frameworks.

Specifically, BIMS has been used to represent all clinical data being currently used within the CardioLab platform (an ongoing project managed by Universitat Pompeu Fabra), demonstrating that it is a solid software system, which could fulfill requirements of a real production environment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

BIMS: Biomedical Information Management System is a project developed with the goal of addressing several limitations suffered in the data acquisition phase of biomedical research projects, in which the University Pompeu Fabra takes part. In general, these problems are widely related to the lack of mechanisms to validate the content of the forms used to gather medical information about the patients. In the chapter 2 several examples are described, but in general, the typical problems are fields which contains numbers out of range, or expressions which does not match with a specific pattern. All these limitations impact in the quality of information, decreasing it, and making unusable the biomedical information being collected.

Summarizing, this project is motivated from the need to control the mechanism used to collect medical data in biomedical research projects. These improved control mechanisms in data acquisition impacts on its quality, increasing it, and making it more useful in research projects.

1.2 Project goal and scope

The main objective of the project is to design and develop a system to create, and manage clinical studies based on Case Report Forms (CRF). These CRF's are composed of fields, and the clinicians fill them with specific patient information. Moreover, in order to force the clinician to follow specific content field rules, like forcing the content of a field to match with a regular expression, the system provides a set of functionalities to constraint the information entered in the field. Therefore, the quality of the data gathered in the clinical study meets the appropriate requirements to use the clinical study, and all related information in research projects. Due to the fact that biomedical research projects manage a wide range of information types and structures, the project also includes the objective to flexibly to adapt to a wide range of biomedical research specialties.

1.3 Project outline

This report is organized as follows. Chapter 2 describes the origins and motivations of the project, and presents a formal requirements analysis.

The state of the art, and the technologies used for the developments, and implementations of this project are detailed in Chapter 3.

Chapter 4 describes the architecture of BIMS: Biomedical Information Management System. The system design and several related aspects are commented in this chapter.

Chapter 5 describes the development of an external tool to migrate all clinical information gathered under an ongoing research project (CDTEAM). This tool represents a realistic use case of BIMS.

Chapter 6 comments the general conclusions of the project.

Finally, the possibilities for the future work are discussed in chapter 7, where the most promising ideas for the BIMS future are commented in a briefly description.

1.4 Planning

This section describes the initial planning established for the global development of the project. It is based on the classical approach to software development engineering [27], also named waterfall model. In this model, the division between stages is clear, and well defined.

All task groups are the listed below:

- Requirements analysis
- Architectonic design
- Implementation of the system
- Testing
- Documentation

It is important to note that this planning is not flexible, because it defines an strict separation between the stages of the project. In addition, it is necessary to complete a phase in order to start the following next. For this reason, in several stages of the project life time, the planning has not been respected, and phases like *implementation* and *testing* have been overlapped temporally. This change has transformed the inflexible initial planning, close to classic software development, to an eXtreme Programing (XP) methodology, where phases are reduced, and agile software development is accomplished by short cycles [2].

WBS	Name	Start	Finish
1	PFC	Nov 11	Apr 13
1.1	Kick off	Nov 11	Nov 11
1.2	Requirements analysis	Nov 11	Dec 19
1.2.1	Collection of requirements	Nov 11	Dec 3
1.2.2	Deliverable document Project Requirements	Dec 3	Dec 3
1.2.3	Requirements analysis	Dec 4	Dec 15
1.2.4	Deliverarble document Requirement Analysis.	Dec 15	Dec 15
1.2.5	Elaboration of document Requirements Analysis	Dec 16	Dec 19
1.2.6	Chapter 2 of the lecture: Requirements Analysis	Dec 19	Dec 19
1.3	Architectonic design of the system	Dec 8	Feb 13
1.3.1	Study software technology	Dec 8	Jan 30
1.3.2	Design of the class diagram of the system	Dec 16	Dec 30
1.3.3	Deliverable document: Class model of the system	Dec 30	Dec 30
1.3.4	Design of the persistence layer	Dec 29	Jan 16
1.3.5	Deliverable document: persistence layer schema E/R.	Jan 16	Jan 16
1.3.6	Design of the presentation layer.	Jan 12	Jan 30
1.3.7	Deliverarble document Presentation layer schema	Jan 30	Jan 30
1.3.8	Elaboration of the document Architectonic design of the system	Feb 2	Feb 13
1.3.9	Chapter 3 of the lecture: System architcture.	Feb 13	Feb 13
1.4	Implementation of the system	Dec 22	Feb 20
1.4.1	System core implementation	Dec 22	Jan 9
1.4.2	Persistence layer implementation	Jan 2	Jan 19
1.4.3	Presentation layer implementation.	Jan 19	Feb 10
1.4.4	Beta version of the system	Feb 10	Feb 10
1.4.5	Correction of the system	Feb 10	Feb 20
1.4.6	Beta version2 of the system	Feb 20	Feb 20
1.5	Testing	Feb 20	Mar 20
1.5.1	Elaboration of the global test which system must pass	Feb 20	Feb 24
1.5.2	Deliverable document: Tests collection	Feb 24	Feb 24
1.5.3	Test execution	Feb 24	Feb 27
1.5.4	Deliverable document: All tests and their results.	Feb 27	Feb 27
1.5.5	Correction of the system	Feb 27	Mar 16
1.5.6	First stable version of the system	Mar 16	Mar 16
1.5.7	Elaboration of the document: Tests	Mar 16	Mar 20
1.5.8	Chapter 4 of the lecture: Tests	Mar 20	Mar 20
1.6	Documentation	Mar 16	Mar 25
1.6.1	Elaboration of user guide, and administration guide of the system	Mar 16	Mar 25
1.6.2	Deliverable document: User Guide	Mar 25	Mar 25
1.6.3	Delivearble document: Administration Guide	Mar 25	Mar 25
1.7	Lecture elaboration	Mar 25	Apr 13
1.7.1	Collection of all document chapters generated at the end of every phase	Mar 25	Mar 26
1.7.2	Mount and complete lecture of the PFC	Mar 26	Apr 13
1.7.3	Lecture of the PFC	Apr 13	Apr 13

Figure 1.1: Planning of the project

Chapter 2

Requirements analysis

In the present chapter the main idea of this project is exposed with the mission to introduce few primary concepts and its motivation, as well as the problems which will be addressed. Moreover, a complete and clear set of requirements is listed, divided into functional and non-functional requirements, with the goal to build a strong basis for the development of the software application.

2.1 Introduction

The main goal of the project is to develop a software application which improves the procedure used to collect, store, and manage medical data for several kinds of clinical studies. These clinical studies are done in several hospitals, and Pompeu Fabra University takes part in managing the generated information. The following section describes the procedure used to accomplish the tasks of collect, storage, and manage medical data, in order to expose its problems. At this point, the problems and limitations will be used to elaborate a clear list of requirements, necessary to initiate the development of a software application oriented towards perform successfully the same procedure (collect, store, and manage medical data of clinical studies).

2.2 Current procedure of collecting and storing clinical studies

Currently, the process of collecting, storing, and managing medical data is mainly concentrated around two specific clinical specialties. These clinical studies are:

1. **Cardiological studies**. Collection of medical data about heart.

2. **Angiological studies**. Collection of medical data about the vascular system.

A clinical study is a generic concept, used to identify an independent project. In this way, a clinical study encompasses several kinds of group studies, which are a collection of medical data about a specific feature. For example, a cardiological study is a project which has the mission to collect, and manage medical data about patients with several kinds of heart diseases (e.g.

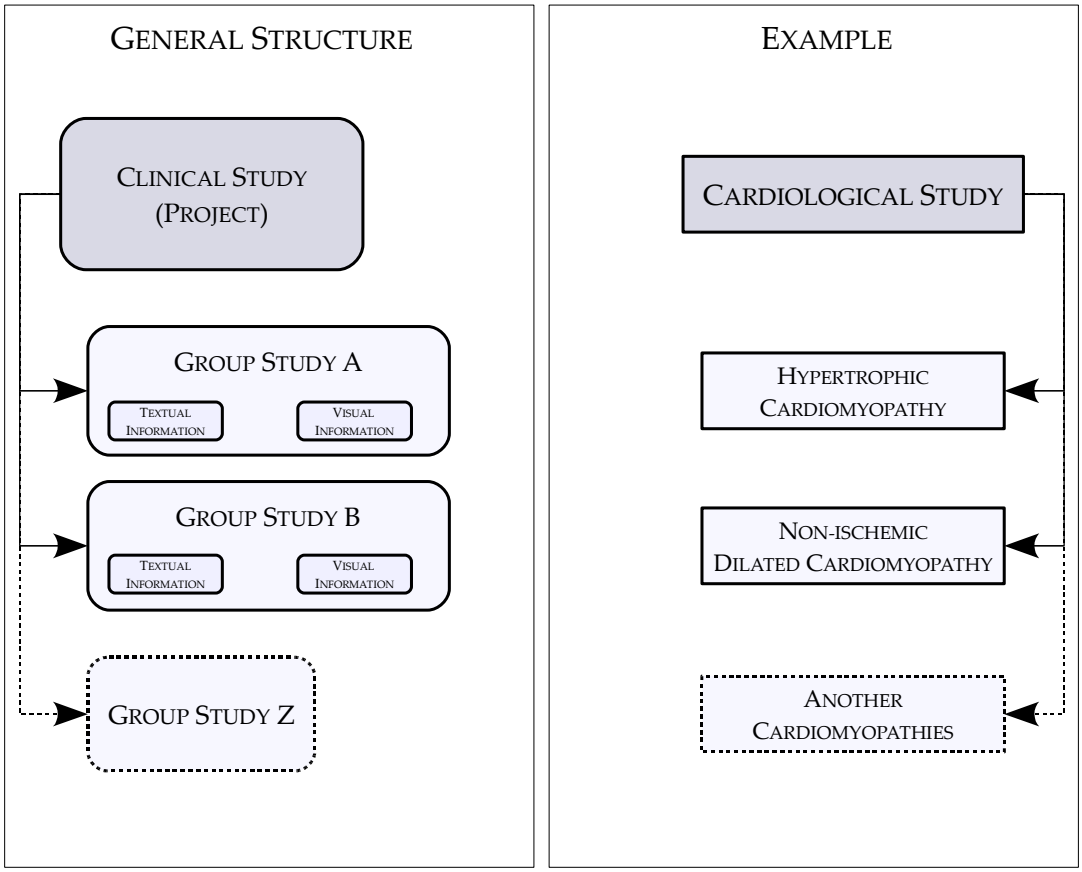


Figure 2.1: Relation between clinical study, and group study.

cardiomyopathies). In this context, a group study represents a set of patients with the same cardiopathy (figure 2.1). Due to this, different group studies are located under the umbrella of the same clinical study.

Every group study contains a collection of medical data of a patient. Medical data can be information as dates, measurements, treatments, or complex images of parts of the brain or of the heart. Hence, the information of medical data could be sorted in two groups, depending on its representation:

1. **Textual information.** Medical data represented by text, for example: dates, measurements, or written comments.

2. **Visual information.** Medical data represented by multi-modal images, for example, a magnetic resonance image (MRI), computed tomography (CT), two-dimensional ultrasound (2DUS), three-dimensional ultrasound (3DUS), single photon emission computed tomography (SPECT).

Figure 2.1 shows the general structure of the concepts: clinical study (project), and group study (specialization of a project). The cardiological study relation is showed too, as an example.

The current procedure of collecting, and storing medical data of a clinical study is executed in three steps.

1. First step: clinician records medical data of group study.
   (a) Clinician records textual information of an specific group study on a spreadsheet. The format of which has been agreed by all participant clinical centers.
   (b) Clinician records visual information of an specific group study on a Digital Video Disc (DVD).
2. Second step: medical data of group study is sent to Pompeu Fabra University from the hospital.
   (a) Clinician sends the spreadsheet, containing the textual information of an specific group study, to the Pompeu Fabra University by e-mail.
   (b) A data curator of research staff at Pompeu Fabra University receives the DVD which contains all visual information of an specific group study.
3. Third step: the Pompeu Fabra University data curator properly stores the information of the group study.
   (a) An script is used to copy all textual information contained in the spreadsheet to a DataBase Management System (DBMS).
   (b) Visual information is stored in a file system. This task is executed manually by the data curator of Pompeu Fabra University.
   (c) The DVD, which contains visual information about the group study, is stored physically in a internal library, in Pompeu Fabra University.

Figure 2.2 summarizes the process.

2.2.1 Problems with the current procedure

The current process of collecting and storing clinical studies, described in previous section, suffers from several limitations.

Firstly, in the first step, a spreadsheet is used to record all textual information of the clinical study. Sometimes, medical staff of a specific hospital do not respect the format and the structure of this spreadsheet, adding o deleting columns or parameters. This action has terrible consequences: the same kind of clinical study can take different structures, formats, and forms. As a consequence of this problem, a task of third step is affected, the use of the script to store properly all textual information in a persistent system in a automated way, does not work. The script has been developed to work with specific formated spreadsheets, and any derivation makes it impossible to use this script without modifications.

In the second step, medical data of clinical study are sent to Pompeu Fabra University. Textual information, stored in a spreadsheet is sent by electronic mail, and visual information is picked up by a member of research staff. In both cases, members of Pompeu Fabra research staff must spend a significant amount of time in order to get and process medical data of a clinical study.

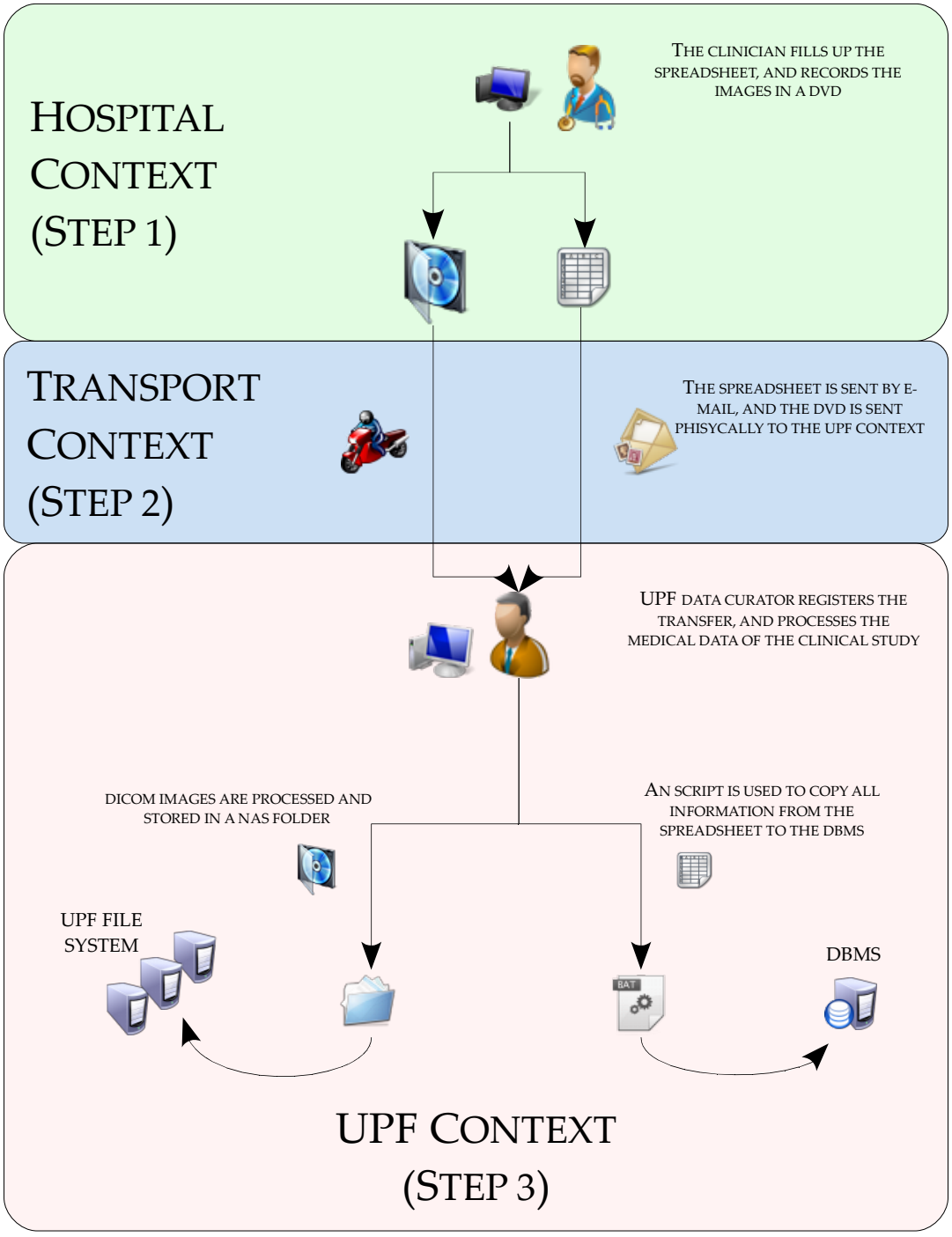


Figure 2.2: Current process of collecting, and storing clinical studies.

In the third step, a script is used to store textual information of clinical study in a proper persistence system, as a DBMS. The use of a script automates this task, however this method is too strict, and it is necessary to modify the script when a change in structure of the clinical study is produced. The other task of this step is to store the visual information in a plain file system. Currently these tasks are performed manually, and a significant amount of time is spent to accomplish this part of third step.

All these limitations are the motivation of a project to facilitate the process of collecting, storing and managing clinical studies.

2.3 Requirements

As it has been described along this chapter, the goal of this project is to develop a software application to improve the process of collecting, storing, and managing clinical studies. The next section provides a list of requirements is given aligned into two categories: functional requirements, and non-functional requirements.

2.3.1 Functional requirements

A functional requirement is a capability or statement of service the system should provide, and how the system should react to particular inputs and how the system should behave in particular situations [28]. In this subsection an ordered list of functional requirements is presented, in order to give the complete scope of the capabilities that should be provided by the software application.

1. **Accessibility**
   (a) The software application MUST provide a web interface to access to it.
2. **Security**
   (a) The software application MUST provide a security mechanism (based on user & password log-in method) in order to allow access to the medical staff which take part in the project, and Pompeu Fabra University research staff. Currently, there are several options to fulfill this requirement, like using a digital certificate combined with an smart card / e-token. In the context of this project, the logging based on a user / password credentials is the easiest option to implement and test.
   (b) The software application MUST provide a security mechanism, based on a deidentifying system, in order to preserve privacy of patient who takes part in clinical studies. This is done to avoid that medical records could expose patient identity.
3. **Core functionalities**
   (a) The software application functionalities MUST be categorized in two groups:
   Administrative tasks Tasks related with the configuration of the software application.
   Non-administrive tasks Tasks related with the main goal of the project: collect medical data of clinical studies (specifically study group).

(b) The users of the software application MUST be categorized in two roles depending on the tasks which they can perform. The roles are:

Administrator Users which can perform administrative tasks.

Normal user Users which can perform non-administrative tasks.

(c) The software application MUST provide a workspace environment (web site) which contains the only tasks available to the user role who has logged in. The tasks available depend on the role of the user.

i. Administrator role, the software application MUST provide a workspace environment (web site) containing only administrative tasks.

ii. Normal user role, the software application MUST provide a workspace environment (web site) containing only non-administrative tasks.

(d) The software application MUST offer the following administrative tasks.

i. The software application MUST provide a functionality to create new Administrator user. Required information:

A. Name and surname.

B. User name.

C. Password.

D. Electronic mail.

ii. The software application MUST provide a functionality to create new Normal users. Required information:

A. Name and surname.

B. User name.

C. Password.

D. Hospital.

E. Electronic mail.

iii. The software application MUST provide the functionality to assign specific study groups to a specific Normal user. By this functionality, a normal user can log in to software application to take part in a group study which belongs to a clinical study.

iv. The software application MUST provide the functionality to modify the information of the another Normal users.

v. The software application MUST provide the functionality to search specific study groups (cardiological studies or angiological studies) by different criteria:

A. Search by date.

B. Search by Hospital.

C. Search by study group.

(e) The software application MUST offer the following non-administrative tasks:

i. The software application MUST provide a infrastructure to receive, storage and manage the medical data about cardiological studies, which generally are divided in four sections:

Diagnosis Contains medical data about the general information of the patient and his clinical history.

Measurements Contains medical data about measures of specific parts of the heart.

Treatment Contains medical data about the treatments of the patient.

Follow-up Contains medical data about same patient six months later.

The medical data about cardiological studies is represented as:

A. Text information. Medical data represented by text characters.

Text comments. Text data (doctor comments, annotations, or dates).

Measurements. Numerical data (units, ratios, closed questions, length, volume, time, pressure, mass).

B. Visual Information. Medical data represented by images.

Images. DICOM[1] [23] format images of big size. Typically, the size of the image is around several G-Bytes.

ii. The software application SHOULD provide a system to save temporally an unfinished group study which belongs to a clinical study.

iii. The software application SHOULD provide a system to resume an unfinished clinical study, started in an other session of the user.

iv. The software application MUST provide a infrastructure to receive, storage and manage the medical data about angiological studies, which contains a wide range kind of medical data.

v. The software application MUST provide the possibility to search old clinical studies by the user who has log into the application. Thus, a Normal user can search, and view old clinical studies entered by himself.

2.3.2 Non-functional requirements

Non-functional requirements define a set of restrictions on the product being developed, the development process, and specify external constraints that the product must meet [28]. The following list contains all non-functional requirements:

1. **Accessibility**

   (a) The software application MUST be accessed only as a web site from any computer with Internet connection.

   (b) The web interface of the software application MUST follow the web presentation standards (XHTML 1.0 and CSS 2.0) described by World Wide Web Consortium[7] (W3C).

   (c) The web interface of the software application MUST be rendered successfully by the last version of the following web browsers: Microsoft Internet Explorer, Mozilla Firefox.

   (d) The web interface of the software application MUST NOT contain frames.

   (e) The web interface of the software application MUST have the following minimal web browser requirements in order to function correctly.

      i. It MUST be useable in a no - Java-script web browser.

      ii. It MUST be useable in web browser without cookies.

[1] Digital Imaging and Communications in Medicine

(f) The web interface of the software application MUST NOT require any plug-in to access to it correctly.

(g) The web interface of the software application SHOULD be rendered successfully by any web browser which respect XHTML 1.0, and CSS 2.0 standard described by World Wide Web Consortium (W3C).

1. **Development process**

   (a) The software application core MUST be developed using Spring framework 2.5, and Java 1.6.0.

   (b) The software application web interface MUST be developed with the following technologies:

   - Spring MVC
   - Spring Web Flow 2.0
   - XHTML 1.0
   - CSS 2.0

   (c) The software application persistence layer MUST be developed using Hibernate as Object Relational Mapping (ORM).

   (d) The software application MUST use MySQL 5.0 as DBMS[2].

   (e) The software application MUST deployed using a Tomcat Server 6.0.

   (f) The software application development MUST be managed using maven 2.0.8 as a project manager and comprehension tool.

   (g) The software application code, and documentation MUST be maintained, and managed, using a Subversion as a control version system.

   (h) The PACS[3] [14] server SHOULD be implemented using the dcm4che project [8].

2. **Product constrains**

   (a) The software application SHOULD manage only big sized DICOM format images. Typically, the size of DICOM format image is around two or three G-bytes. Types of images:

   i. Magnetic Resonance Image (MRI).
   ii. Computed Tomography (CT).
   iii. Two-Dimensional Ultrasound (2DUS).
   iv. Three-Dimensional Ultrasound (3DUS).
   v. Single Photon Emission Computed Tomography (SPECT).

   (b) The software application SHOULD storage visual information using *Picture Archiving Computer System* (PACS) technology.

In the following section, a requirements analysis is performed, in order to obtain a well-modeled requirements group, which provides a solid basis for the following phases of the development process.

[2]DataBase Management System
[3]Picture Archiving and Communication System

Primary actors	They are the domain elements who/which starts the process, and requests the functionality fulfilled by the system.
Secondary actors	They are the domain elements who/which receives any action by th system.
Context	A little description of the use case.
Preconditions	A group of circumstances to be accomplished before starting the use case.
Successful Postconditions	A group of circumstances which indicate that the use case has accomplished its objective.
Fail Postconditions	A group of circumstances which indicate that the use case has not accomplished its objective.
Successful scenario	A set of steps which describe the life cycle of the use case finished successfully.
Include	Another use case necessary as a previous stage.
Extend	Another use case which could be started after the last step of the successful scenario. It 's possible to link these use cases.

Table 2.1: Use case: information description about every field of the table.

2.4 Requirements analysis

In this section a rigorous requirements analysis is exposed in order to build a solid basis for the rest of the project, with the mission to minimize ambiguities derivated of the use of the natural language in the requirements collection. The content and structure of this section is widely based on a collection, analysis of a set of use cases [6].

Firstly, a use case diagram is presented as a reflex of the requirements flocked in section 2.3.1: Functional Requirements. The figure 2.3 shows all use cases fulfilled by the system which is going to be developed. Due to every use case has an associated information, e.g. a flux of events, or a group of preconditions, each use case diagramed in figure 2.3 has a associated table which contains several aspects to have in mind in development process.

Finally, in the second part of the section a first approach to the design of the system is presented. It is an informal diagram, but it so useful to start thinking about the several the parts which will compound the whole system, developed and built along the next phases of the project, and described in the following chapters.

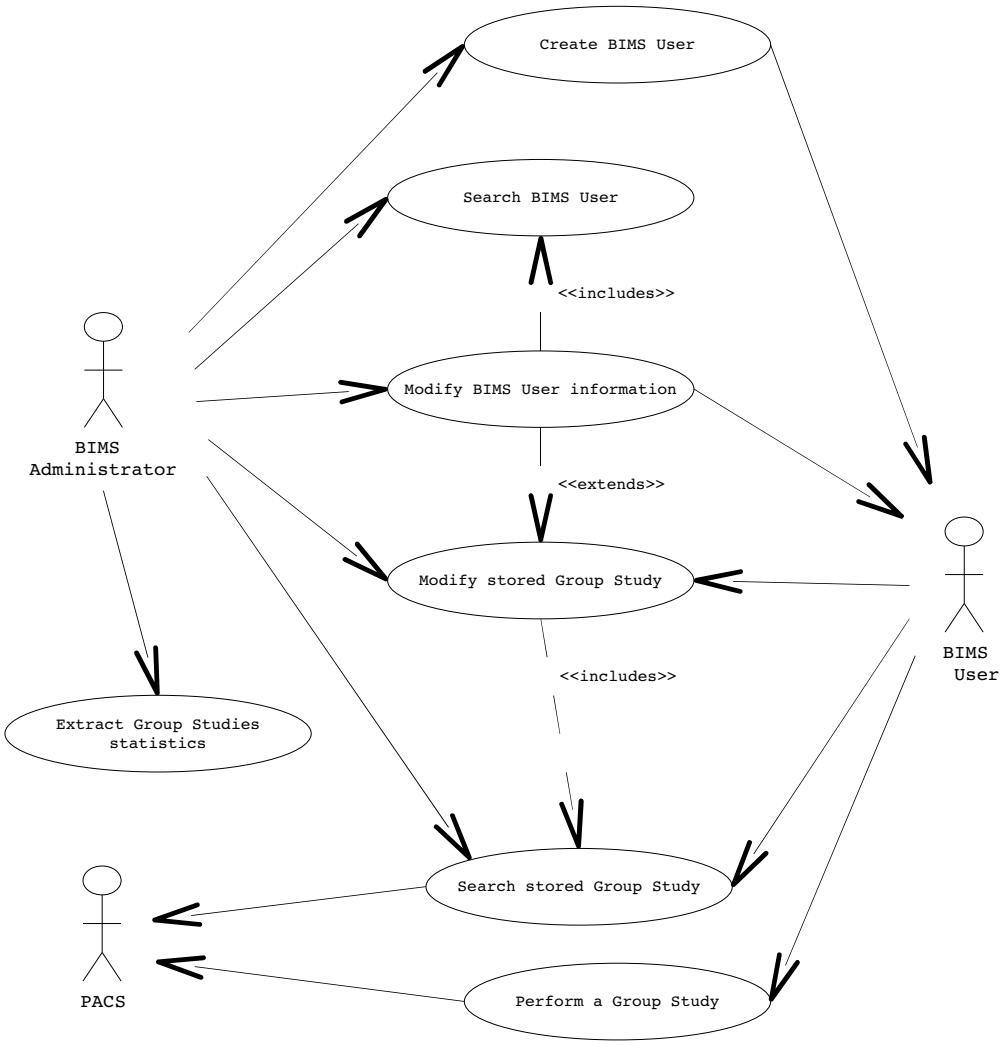


Figure 2.3: Use case diagram.

2.4.1 Create BIMS User

Primary actors	BIMS Administrator
Secondary actors	BIMS User
Context	The BIMS Administrator wants to provide access to the system to a specific medical staff member of specific Hospital.
Preconditions	• The BIMS Administrator is logged and validated in to the system as user with Administrator role.
Successful Postconditions	• The BIMS User information is stored in to the system. • The BIMS User receives user / password credentials in a e-mail. • The BIMS User logs in to the system successfully.
Fail Postconditions	• The BIMS User information is not stored in to the system.
Successful scenario	1. The BIMS Administrator access to the section of the BIMS web site which provides the functionality of adding new BIMS User to the system. 2. The BIMS Administrator introduces information about the new BIMS User: name, surname, hospital, e-mail, username, password. 3. The system displays a confirmation page, with the information introduced by the BIMS Administrator. 4. The BIMS Administrator confirms the process. 5. The system stores the information. If an error occurs, the system show the reason and provides a link to main page of the web site. 6. The system sends an electronic mail to the BIMS User specifying the credentials.
Include	-
Extend	-

Table 2.2: Use case: Create BIMS User

2.4.2 Search BIMS User

Primary actors	BIMS Administrator
Secondary actors	-
Context	The BIMS Administrator wants to search and view specific BIMS User information using a specific searching criteria (name, surname, user-name, hospital)
Preconditions	• The BIMS Administrator is logged and validated in to the system as user with Administrator role
Successful Postconditions	• The system displays to the BIMS Administrator all BIMS Users which match with de searching criteria introduced.
Fail Postconditions	• The system is not able to execute the query.
Successful scenario	1. The BIMS Administrator access to the section of the BIMS web site which provides the functionality of searching BIMS Users. 2. The BIMS Administrator introduces the criteria of the BIMS User search. 3. The system displays a list which contains all BIMS Users which match with the searching criteria introduced.
Include	-
Extend	-

Table 2.3: Use case: Search BIMS User

2.4.3 Modify BIMS User information

Primary actors	BIMS Administrator
Secondary actors	BIMS User
Context	The BIMS Administrator wants to delete or modify associated information of a specific BIMS User(General information, clinical studies assigned, group studies entered)
Preconditions	• The BIMS Administrator is logged and validated in to the system as user with Administrator role
Successful Postconditions	• All changes in BIMS User information are stored in the system. • The system sends an e-mail to the BIMS User in order to notify the performed action.
Fail Postconditions	• All changes in BIMS User information are not stored in the system.
Successful scenario	1. The BIMS Administrator selects a specific BIMS User. 2. The system displays all associated information to the selected BIMS User. 3. The BIMS Administrator modifies the desired information. 4. The system displays a confirmation before execute the changes. 5. The system modifies stored information. 6. The system sends an e-mail to the BIMS User to notify the actions performed.
Include	Search BIMS User
Extend	Modify stored Group Study

Table 2.4: Use case: Modify BIMS User information

2.4.4 Extract Group Studies Statistics

Primary actors	BIMS Administrator
Secondary actors	-
Context	BIMS Administrator wants to obtain specific statistics about over a set of Group Studies.
Preconditions	• The BIMS Administrator is logged and validated in to the system as user with Administrator role
Successful Postconditions	• The system displays the statistics to the BIMS Administrator.
Fail Postconditions	• The system is no able to execute the query.
Steps of successfully scenario	1. The BIMS Administrator access to the section of the BIMS web site which provides statistics operations over stored Group Studies. 2. The BIMS Administrator selects a set of Group Studies to extract specific statistics. 3. The system displays to the BIMS Administrator all available statistical operations available. 4. The BIMS Administrator selects the operation to perform over the selected Group Studies. 5. The system displays the results.
Include	-
Extend	-

Table 2.5: Use case: Extract Group Studies Statistics

2.4.5 Modify stored Group Study

Primary actors	BIMS Administrator, BIMS User
Secondary actors	-
Context	• The BIMS Administrator wants to modify any field of any Group Study stored in the system. • The BIMS User wants to modify any field of any Group Study entered in to the system by himself.
Preconditions	• The BIMS Administrator is logged and validated in to the system as user with Administrator role. • The BIMS User is logged and validated in to the system as user with Normal User role.
Successful Postconditions	• The system modifies the Group Study.
Fail Postconditions	• The system is no able to perform the modification over the Group Study.
Successful scenario	1. The BIMS Administrator / the BIMS User select a specific Group Study. 2. The system displays all associated information to the Group Study. 3. The BIMS Administrator / The BIMS User modifies the desired information. 4. The system displays a confirmation before execute the changes. 5. The system modifies stored information. 6. The system sends an e-mail to the BIMS User to notify the actions performed.
Include	Search stored Group Study
Extend	Modify BIMS User information

Table 2.6: Use case: Modify stored Group Study

2.4.6 Search stored Group Study

Primary actors	BIMS Administrator, BIMS User
Secondary actors	-
Context	The BIMS Administrator wants to search and view specific Group Study stored in the system. Or, the BIMS User wants to search and view specific Group Study stored in the system and entered by himself.
Preconditions	• The BIMS Administrator is logged and validated in to the system as user with Administrator role. • The BIMS User is logged and validated in to the system as user with Normal User role.
Successful Postconditions	• The system displays desired information.
Fail Postconditions	• The system is no able to perform the search.

Successful scenario	1. The BIMS Administrator / the BIMS User access to the section of the BIMS web site which provides the functionality of searching Group Studies.
	2. The BIMS Administrator / the BIMS User introduces the criteria of the BIMS User search.
	3. The system performs the search using the parameters introduced.
	  • In the BIMS Administrator case, the search is over all Group Studies stored in the system.
	  • If the BIMS User case, the search is over all Group Studies stored in the system, and entered by the same BIMS User.
	4. The system displays a list which contains all Group Studies which match with the searching criteria introduced. Every Group Study has a link to the images associated contained in PACS implementation.

Include	Search stored Group Study
Extend	Modify BIMS User information

Table 2.7: Use case: Search stored Group Study

2.4.7 Perform a Group Study

Primary actors	BIMS User
Secondary actors	-
Context	The BIMS User wants record medical data about a specific Group Study under a specific Clinical Study.
Preconditions	• The BIMS User is logged and validated in to the system as user with Normal User role. • The BIMS User has been assigned to the Clinical Study which belongs the Group Study to perform.
Successful Postconditions	• The information about Group Study entered by de BIMS User is stored in the system, and in the PACS implementation.
Fail Postconditions	• The system is no able to store the information of the Group Study.
Successful scenario	1. The BIMS User access to the section of the BIMS web site which provides the functionality of performing Study Groups. 2. The system displays the Clinical Studies assigned to the BIMS User. 3. The BIMS User selects Clinical Study and Group Study which wants to enter. 4. The BIMS User enters the required information. 5. The system stores information of the Group Study, and the images are sent to the PACS implementation.
Include	-
Extend	-

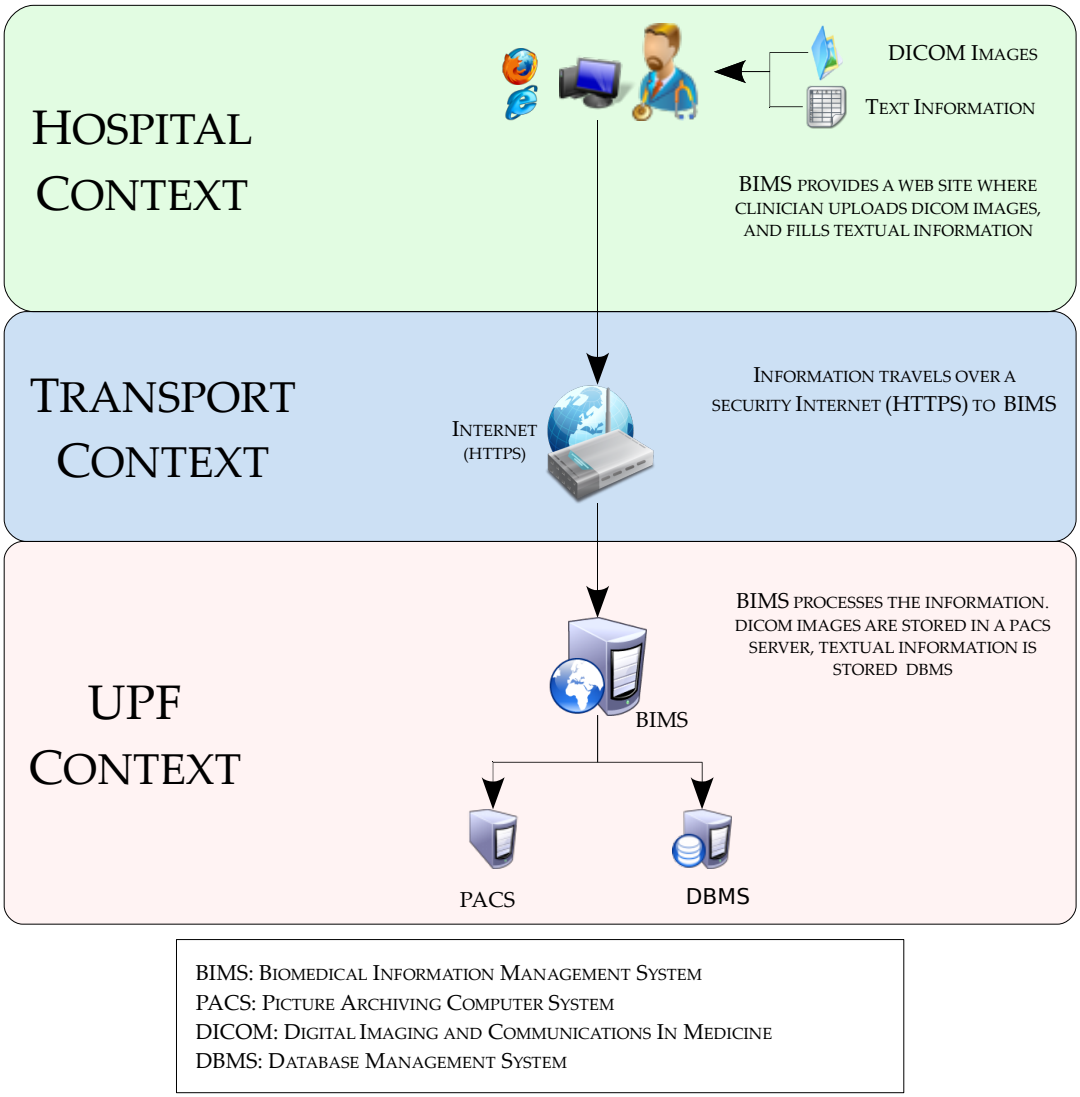
Table 2.8: Use case: Perform a Group Study

2.5 High level approach to the system architecture

Since all requirements have been collected, written, and expressed in a rigorous way, the following and natural step is to present the first approach to the system architecture. It is important to note that it is an informal approach, because the mission of this section is not give a deep design of the final system. Instead of this, the goal is to introduce the general aspects of the system developed to fulfill the list of use cases presented in the last section. Moreover, in order to see how the new system fits in the domain presented in section 2.2: Current procedure of collecting and storing clinical studies, a very similar diagram is used. Figure 2.4 shows the global idea behind the system which is going to be designed and built in the following phases of the project.

BIMS: BIOMEDICAL INFORMATION MANAGEMENT SYSTEM is the software system developed to fulfill the project requirements. The main objective of BIMS is to provide a global infrastructure to receive, store, and manage medical data grouped in Group Studies. Moreover, the BIMS offers a set of functionalities to access to stored medical data, and extract statistics and related information. The main idea is to assemble a web application accessible by a web explorer (Microsoft Internet Explorer, or Mozilla Firefox). Due to this, this system will be accessible from any computer of the world with Internet connection. This approach introduces two basic variations respect the procedure showed in figure 2.2. On the one hand, the visual information storage system based on a fairly file system disappears, and the BIMS will use a more proper system implemented with PACS[4]. On the other hand, and maybe the most important difference is that Pompeu Fabra University research staff have not to take part in procedure of collecting and storing clinical studies (picking up DVD or retyping spreadsheets), it will be an automated process. The tasks performed by UPF consists in the supervision of the BIMS daily operation.

[4]Picture Archiving Computer System



BIMS: BIOMEDICAL INFORMATION MANAGEMENT SYSTEM
PACS: PICTURE ARCHIVING COMPUTER SYSTEM
DICOM: DIGITAL IMAGING AND COMMUNICATIONS IN MEDICINE
DBMS: DATABASE MANAGEMENT SYSTEM

Figure 2.4: High level architecture of BIMS

Chapter 3

State of the art

3.1 Introduction

Nowadays, there are a lot of frameworks which provide a set of tools to build competitive applications easily. In this project, several frameworks has been studied and tested with the objective to develop a final software package to fulfill the requirements described in chapter 2, in a rigorous way close to an enterprise environment. In the following sections, a briefly description is presented about every technology which take part in the project. The aim of this chapter is to provide a smooth introduction the reader to the following items:

- **Spring Framework**. Platform to build and run enterprise Java applications, based plain old Java objects (POJO's).
- **Hibernate**. Object-relational mapping (ORM) tool to provide an object-oriented persistence and query service.
- **JavaServer Faces**. Server-side user interface component framework for Java technology-based web applications.
- **Maven**. Software project management tool, used to control project's build, reporting and documentation from a unique point.
- **OpenClinica**. Software platform used to collect, manage, and store data on clinical trials in a database.

The following sections contains a short introduction of these technologies, used in different parts of BIMS. In the bibliography the references are listed, and they can be used to get more information.

3.2 Spring Framework

Spring is an open source framework, created by Rod Johnson and described in his book *Expert One-on-One: J2EE Design and Development*. It was created to address the complexity of enterprise application development. Spring makes possible to use plain-vanilla JavaBeans (POJO's)

to achieve several kind of functionalities that were previously only possible with *Enterprise Java Beans* (EJB):

- *Java Naming and Directory Interface* (JNDI).
- *Java Management Extensions* (JMX).
- *Java Persistence API* (JPA).
- *Java Transaction API* (JTA).

However, Spring's usefulness is not limited to server-side development. Any Java application can benefit from Spring in terms of simplicity, testability, and loose coupling.

The core of the Spring Framework is based on the principle of Inversion of Control (IoC). Applications that follow the IoC principle use configuration that describes the dependencies between its components. It is then up to the IoC framework to satisfy the configured dependencies. The *inversion* means that the application does not control its structure; it is up to the IoC framework to do that. This injection of dependencies at runtime has sometimes led to IoC being given the much more descriptive name dependency injection (DI). Spring's DI implementation puts focus on loose coupling: the components of the application should assume as little as possible about other components.

Summarizing in a global description, Spring is a lightweight dependency injection and aspect-oriented container and framework. By steps:

- **Lightweight**. Spring is lightweight in terms of both size and overhead. The bulk of the Spring Framework can be distributed in a single JAR file that weighs in at just over 2.5 MB. Moreover the processing overhead required by Spring is negligible.
- **Dependency Injection**. Spring promotes loose coupling through dependency injection (DI). When DI is applied, objects are passively given their dependencies instead of creating or looking for dependent objects for themselves. You can think of DI as JNDI in reverse-instead of an object looking up dependencies from a container, the container gives the dependencies to the object at instantiation without waiting to be asked.
- **Aspect-oriented**. Spring comes with rich support for aspect-oriented programming (AOP) that enables cohesive development by separating application business logic from system services (such as auditing and transaction management).
- **Container**. Spring is a container in the sense that it contains and manages the lifecycle and configuration of application objects. In Spring, it is possible to declare how each of the application objects should be created, how they should be configured, and how they should be associated with each other.
- **Framework**. Spring makes it possible to configure and compose complex applications from simpler components. In Spring, application objects are composed declaratively, typically in an XML file. Spring also provides much infrastructure functionality (transaction management, persistence framework integration, etc.), leaving the development of application logic to the application architect.

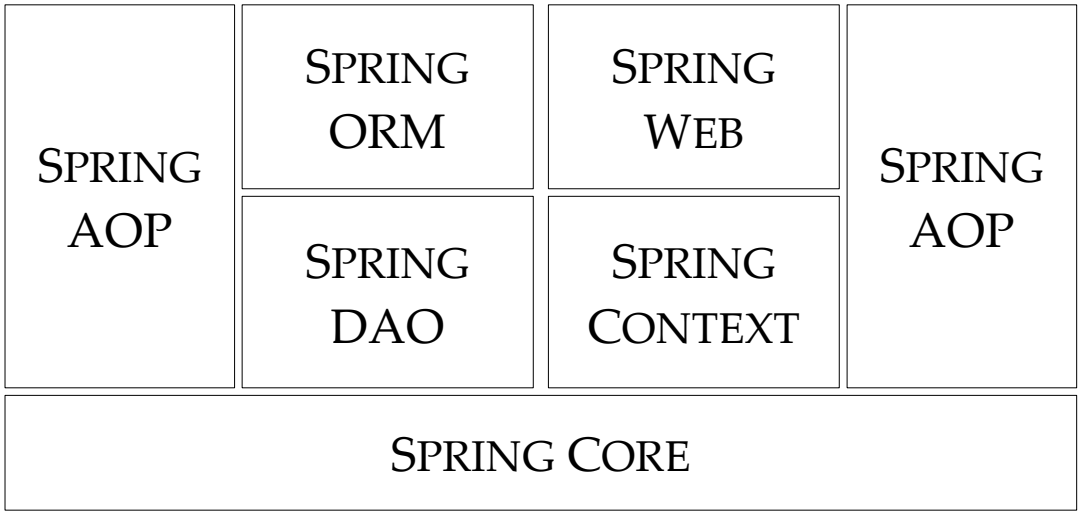


Figure 3.1: Spring framework architectural view

The Spring Framework is made up of several well-defined modules (see figure 3.1). When taken as a whole, these modules give to the architect everything he needs develop enterprise-ready applications. It is important to notice that the Spring framework provides the freedom to to choose the modules that suit the application needs, and look to other options when Spring does not fit the needs. In fact, Spring offers integration points with several other frameworks and libraries.

Each of the modules (or components) that comprise the Spring framework can stand on its own or be implemented jointly with one or more of the others. The functionality of each component is as follows:

- **Spring Core** The core container provides the essential functionality of the Spring framework. A primary component of the core container is the *BeanFactory*, an implementation of the Factory pattern. The BeanFactory applies the Inversion of Control (IOC) pattern to separate an application's configuration and dependency specification from the actual application code.

- **Spring Context** The Spring context is a configuration file that provides context information to the Spring framework. The Spring context includes enterprise services such as JNDI, EJB, e-mail, internalization, validation, and scheduling functionality.

- **Spring AOP** The Spring AOP module integrates aspect-oriented programming functionality directly into the Spring framework, through its configuration management feature.

- **Spring DAO** The Spring JDBC DAO abstraction layer offers a meaningful exception hierarchy for managing the exception handling and error messages thrown by different database vendors. The exception hierarchy simplifies error handling and greatly reduces the amount of exception code you need to write, such as opening and closing connections.

- **Spring ORM** The Spring framework plugs into several ORM frameworks to provide its Object Relational tool, including JDO, Hibernate, and iBatis SQL Maps. All of these comply to Spring's generic transaction and DAO exception hierarchies.
- **Spring Web module** The Web context module builds on top of the application context module, providing contexts for Web-based applications.
- **Spring MVC framework** The Model-View-Controller (MVC) framework is a full-featured MVC implementation for building Web applications. The MVC framework is highly configurable via strategy interfaces and accommodates numerous view technologies including JSP, Velocity, Tiles, iText.

Finally, it is important to notice that Spring framework aim is not *to reinvent the wheel*. Spring leans heavily on existing APIs and frameworks. For example, Spring does not implement its own persistence framework, instead, it fosters integration with several capable persistence frameworks, including simple JDBC, iBATIS, Hibernate, and JPA.

In order to get more information about Spring framework please consult bibliography chapter of this report.

3.3 Hibernate

Hibernate is an open source object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. One of the most important mission of Hibernate is to solve object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions. Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), also provides data query and retrieval facilities and can significantly reduce development time otherwise spent with manual data handling in SQL and JDBC.

The figure 3.2 diagrams a architecture example to positionates the Hibernate element in a application components stack.

Hibernate makes use of persistent objects (POJO) along with XML mapping documents for persisting objects to the database layer. Rather than utilize byte code processing or code generation, Hibernate uses runtime reflection to determine the persistent properties of a class. The objects to be persisted are defined in a mapping document, which serves to describe the persistent fields and associations, as well as any subclasses or proxies of the persistent object. The mapping documents are compiled at application startup time and provide the framework with necessary information for a class. Additionally, they are used in support operations, such as generating the database schema or creating stub Java source files.

3.4 JavaServer Faces

JavaServer Faces (JSF) is a open source Java framework for building Web applications. It was developed through the Java Community Process (JCP) and it will become a part of Java 2 Enterprise Edition (J2EE). It simplifies development by providing a component-centric approach to

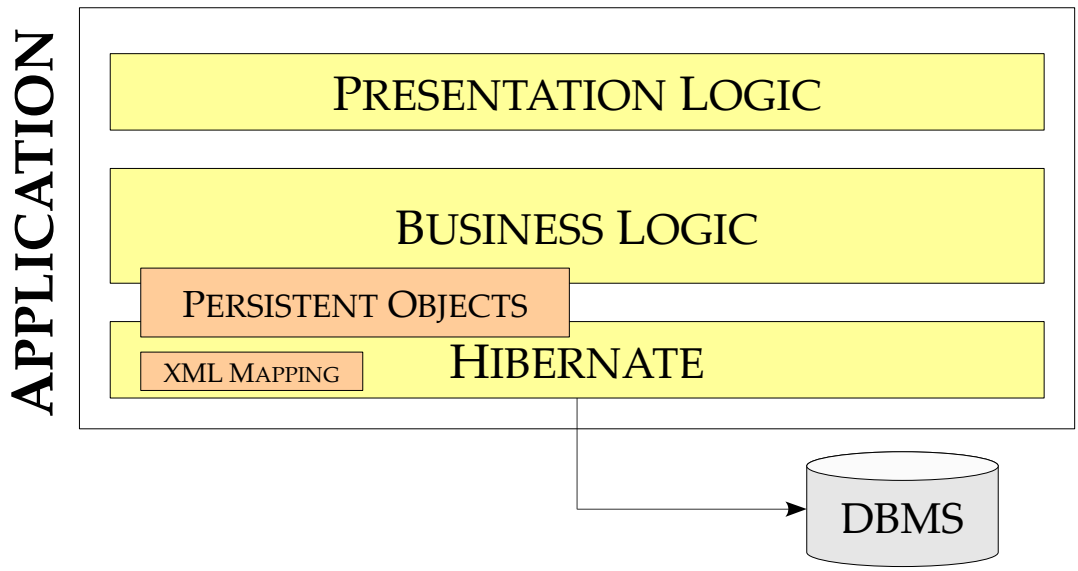


Figure 3.2: Hibernate position

developing Java Web user interfaces (UI). For this, one of the main goals of Java Server Faces is to bring the RAD[1] style of application development, made popular by tools like Microsoft Visual Basic and Borland Delphi, to the world of Java web applications. Moreover, JSF also ensures that applications are well designed with greater maintainability by integrating the well established Model View Controller (MVC) design pattern into it's architecture.

The most important aspect of JavaServer Faces lies in its user-interface component model, where applications are merely built from collections of components that can render themselves in diverse ways for multiple client types. Vaguely similar to other proprietary technologies such ASP.Net, JavaServer Faces' UI component model technology offers unprecedented productivity by allowing the developer to construct Web user interfaces using prefabricated user-interface (UI) components, as opposed to having to construct the user interface entirely from scratch. JavaServer Faces UI components come in many forms and can be as simple as an *outputLabel* which simply displays text, or as complex as a *dataTable* which can represent a tabular data from collections of data such as from a database table.

The Java Server Faces specification provides a set of base UI components in its Reference Implementation (RI). These are very useful on their own, and include two libraries of components which largely mirrors the standard HTML input elements which aids in common application development tasks such as internationalization, and validating/converting input data. In addition to providing a base library of UI components, the JavaServer Faces API offers the ability to create, and extend custom JavaServer Faces UI components providing additional functionality above and beyond the base components. Apache MyFaces (Tobago project) is an example of this possibility to create new JavaServer Faces UI components.

Finally, it is important to notice that Java Server Faces is considered a web application framework because it performs a lot of common development tasks, so that application architects can

[1]Rapid Application Development

Figure 3.3: Tobago calendar component.

focus on more interesting aspects like business logic. One of the key features is support of the Model / View / Controller design pattern, which enforces separation of presentation and business logic code.

3.4.1 Apache MyFaces - Tobago project

Tobago project, under Apache software Foundation[2], offers a set of tools to create web applications with a consistent look and feel by providing you with a collection of comfortable, high-level components. These components can be positioned with a layout manager. The goal is to approximate the appearance of classic desktop applications. The figure 3.3 shows an example of a calendar component of Tobago project.

Tobago is based on a strict separation of structure and design. It is possible to describe the structure of a page and which controls it contains; the representation of the page and its style are handled by Tobago. Furthermore, the design and the style are customizable elements through theme management. To achieve output independence, the views should be developed without any HTML, CSS, or JavaScript. A Tobago page normally contains only JSF and Tobago tags. Features, styling, and design via HTML, CSS, or JavaScript are handled externally by the theme. Currently, Tobago project contains only themes for HTML clients, but this effort to define a real separation between the information and the design makes feasible to develop to other clients.

The development of Tobago started in 2002. With the introduction of the JSF standard, it was decided to base Tobago on JSF. By 2005, Tobago was released as an open source project and became a subproject of Apache MyFaces in 2006.

[2]http://www.apache.org/

3.5 Maven

Maven is an open source project management framework which encompasses a project object model, a set of standards, a project life cycle, a dependency management system, and logic for executing plug-in goals at defined phases in a life cycle. In order to use Maven, it is necessary to describe the project using a well-defined project object model (POM), Maven can then apply cross-cutting logic from a set of shared (or custom) plug-ins.

Maven was borne of the practical desire to make several projects at the Apache Software Foundation work in the same, predictable way. Prior to Maven, every project at the Apache Software Foundation had a different approach to compilation, distribution, and Web site generation.

3.6 OpenClinica

OpenClinica is a free[3], open source clinical trial software platform for Electronic Data Capture (EDC) and medical data management in clinical research. OpenClinica provides a web-based interface to manage clinical trials for electronic data capture and clinical data management in clinical research. It facilitates some related tasks like, protocol configuration, design of case report forms, clinical data capture, and study/data management.

Following, some aspects of OpenClinica platform are described:

- Manage Study. Facilitates configuration and management of clinical trial protocols, sites, CRFs, users and study event definitions. You can define data elements, CRFs, and protocol events without any custom programming.
- Submit Data. Provides a user-friendly web-based interface for subject enrollment, electronic data submission, and data validation. The figure 3.4 shows an example of a CRF generated by OpenClinica.
- Extract Data. Enables definition, filtering, and extraction of study datasets.
- Administer System. Allows overall system oversight, auditing, configuration, user account management, and reporting by administrators.

OpenClinica is not a tool, or framework which takes part actively in the BIMS design and development. Indeed, they are projects with similar objectives. Although OpenClinica is located in a different context than BIMS (OpenClinica is a software package developed and supported by Akaza Research, a private company), it is useful to study the functionalities provided by OpenClinica as a reference. There are two main requirements in BIMS project, which are not fulfilled by OpenClinica software. Firstly, OpenClinica does not support anonymization operations of critical patient information (subsection 4.3.5 describes BIMS anonymization process). Secondly, OpenClinica does not consider the collection of DICOM files in any way, as well as their management using an external PACS server. These are compulsory needs to be fulfilled in BIMS project.

[3] http://www.openclinica.org

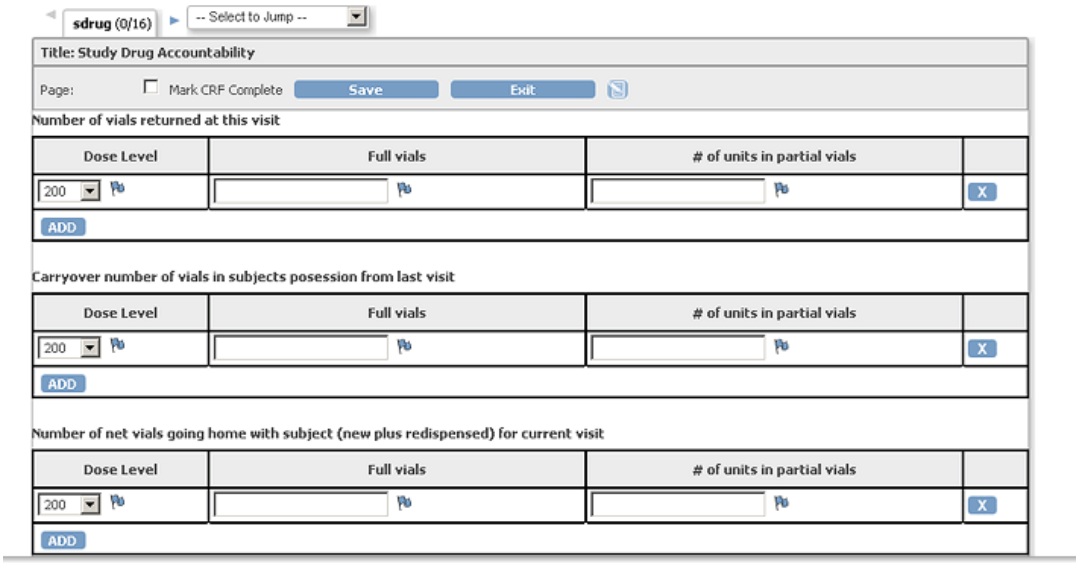


Figure 3.4: CRF example of OpenClinica

Chapter 4

BIMS: Biomedical Information Management System

This chapter describes the architecture of BIMS: Biomedical Information Management System. Firstly, the design showed in the figure 2.4 (section 2.5) is used to link and describe an architectural design of BIMS. It includes all elements involved in the whole process of creating and managing clinical studies.

BIMS is composed of two main components:

1. **BIMSCore**. It is the heart of BIMS.
2. **BIMSWebAccess**. It is a access provider to BIMSCore services.

4.1 Introduction

The goal of this section is to describe the BIMS design from a high level point of view, with the aim to provide an overview of the whole system. This section links and follows the simple design introduced in section 2.5.

The whole system structure is depicted in figure 4.1. The system is a web-based software application, with a security layer based on encrypted communication, and a checked access using a user/password credentials. Even though this is a high level design approach, it must be noted that two important requirements described in chapter 2 are fulfilled. On the one hand, accessibility is guaranteed since a web-based application is available from any computer with Internet connection in the world. On the other hand, an https connection, and a logging-access based on user/password credentials, provides a proper security level to guarantee confidentiality of data submitted by the clinician.

The textual information submitted by the clinician is stored in a database management system, and the visual information is stored in a Picture Archiving Communication System (PACS). The database schema is managed by BIMS system, hence, this is considered part of BIMS. In contrast, PACS server is considered to be administrated by an external entity, for this we do not consider it as a part of BIMS. It is an external service to manage DICOM objects.

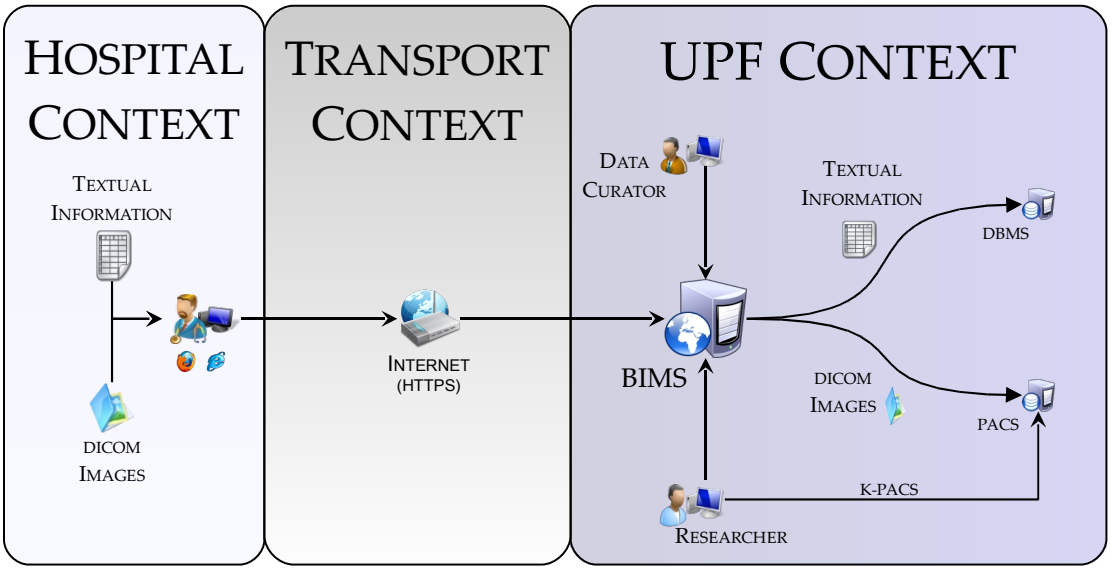


Figure 4.1: First approach to BIMS: Biomedical Information Management System.

The next section focuses on the BIMS design, as the basis of the process to manage medical information of clinical studies.

4.2 BIMS: Biomedical Information Management System

BIMS is a system composed by two independent software projects. The internal design of the system, as whole entity, is based on a modularization process which follows a simple principle: a clear separation between all concerns of the system. Historically, this is a widely discussed topic [24] since the early 70's.

The main idea behind the modularization is to build an architecture which is split in several layers. Every layer is responsible for providing a set of features to manage an specific concern. Hence, the modularization process success depends on the good work on initial stages of the project, where general motivations and requirements analysis are performed. An appropriate modularization process provides several benefits:

- Distributed, and parallel development. The ability to define a clear separation between concerns of a system can be used to build the whole system as the development of several independent projects.
- Improved usability. The ability to create a system composed of independent modules provides an improved usability which makes it easy to understand the whole system. A well implemented modularity reduces the learning curve, making it smoother and easier to navigate through the code of every module.
- Debugging becomes easier. The modularization process, applied over a global system, encapsulates the system concerns in different modules. Hence, if a bug is detected, it is easier

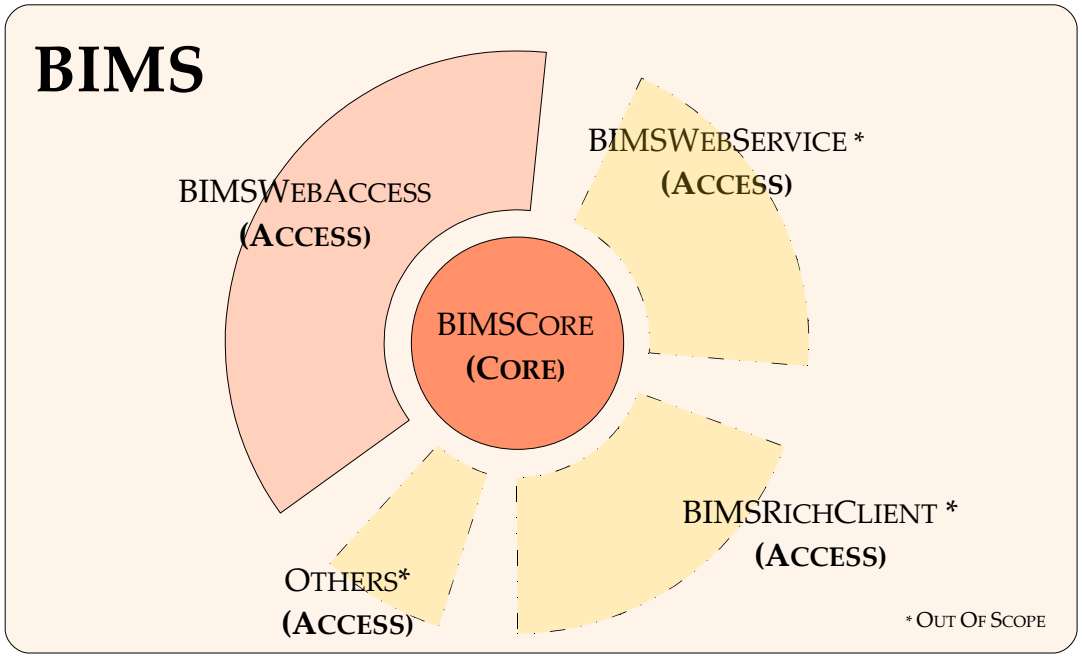


Figure 4.2: BIMS high level architectural view

to find and focus on the module where this error is located.

This chapter describes the design of BIMS in a incremental manner, from a high level architecture, to a lower level point of view. Figure 4.2 shows the high level architectural view of BIMS. The whole system is divided into two layers. Hence, it is possible to assign every functionality provided by the system to an specific layer. Figure 4.2 shows a first layer, named *core*, and a higher layer named *access*. The objective of this division is to establish a separation between the domain context functionalities, and the features to access to BIMS functionalities. The decision to define this separation allows creating to create two different and independent software projects:

1. **BIMSCore**. A software project to provide the application domain functionalities.
2. **BIMSWebAccess**. A software project to provide access to BIMSCore functionalities. .

To illustrate the relationship between the BIMSCore, and the BIMSWebAccess project, figure 4.3 shows an stack diagram which depicts BIMS as a whole system composed of the two projects. The BIMSWebAccess project provides accessibility features through a web browser, and the BIMSCore project provides a unified library which contains all domain functionalities BIMS.

On the one hand, BIMSCore project is a software library which gathers all domain context functionalities of the BIMS. On the other hand, BIMSWebAccess is a web-based software application which provides an structured access to BIMS functionalities through a web portal, using a web browser. BIMSWebAccess web-based application imports the BIMSCore project library as the engine to execute domain context functionalities. The modular design, and the separation between accessibility issues (managed in the access layer), and context domain functionalities

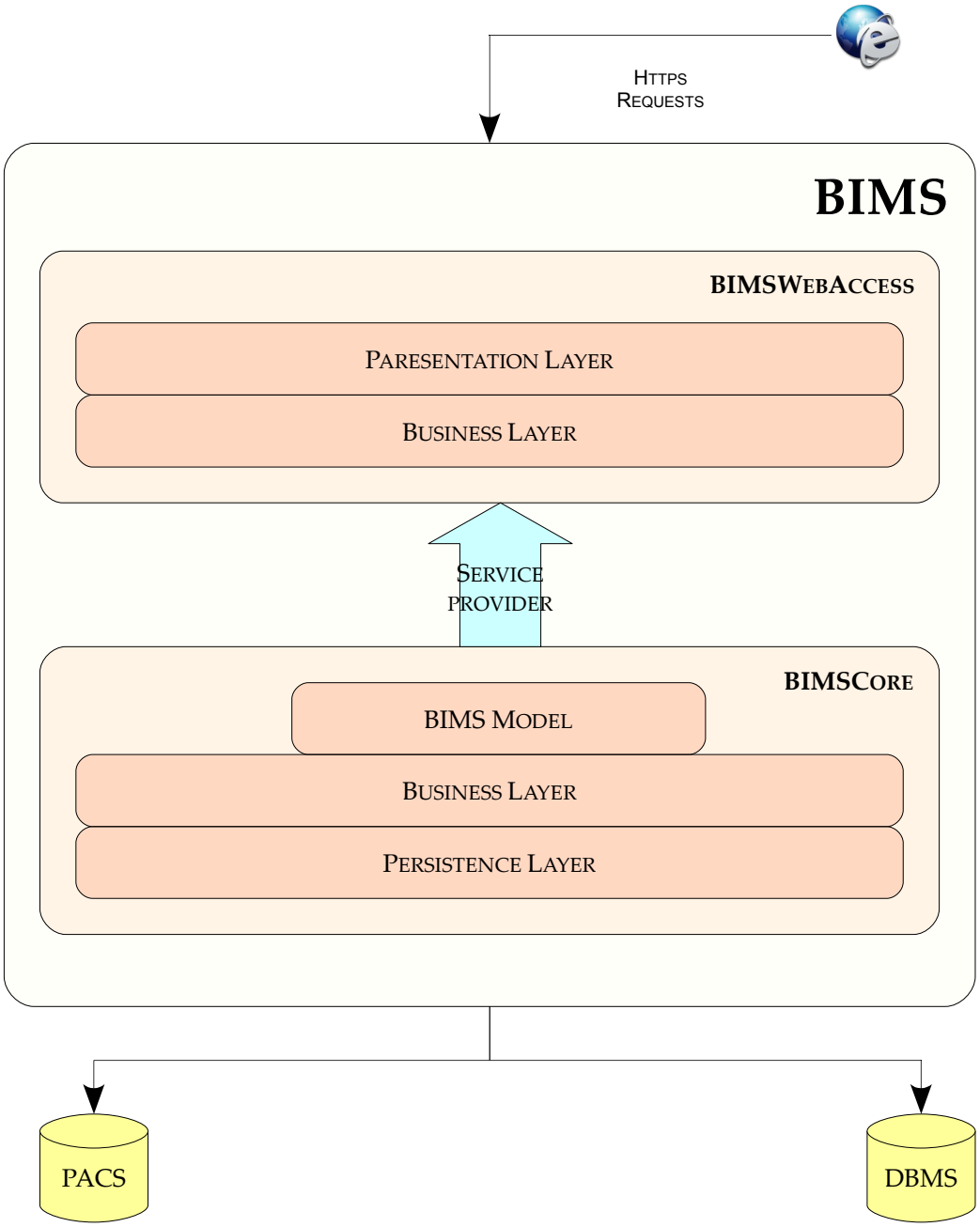


Figure 4.3: Application stack of BIMS

(managed in the core layer) establishes a consistent framework to develop other software project possibilities, to fulfill other requirements of other environments. Thus, it would be possible to develop a web service to provide access to BIMSCore (in the figure 4.2 is a hypothetical software project named BIMSWebService at access layer). Also, it would be possible to develop an hypothetical rich client (using Swing visual technology) to implement an auto installer application, to deploy in the user computer to use BIMS.

Summarizing, the modular design provides a flexible system based in two layers, which are implemented by independent projects. Hence, each of these projects has an specific mission, BIMSCore must provide a set of context domain functionalities through a public and unique entry point (a unique and public interface), and BIMSWebAccess has to provide a mechanism to access to BIMS using a web browser.

4.3 BIMSCore. The first layer of BIMS architecture

The first level of BIMS architecture is an independent project named BIMSCore, which is the heart of the system. The objective is to encapsulate in a self-sufficient project all the context domain capabilities of BIMS, without being concerned about presentation issues. Hence, the domain tasks, as management of the model domain elements of the system (which implies CRUD[1] operations) are tasks which are addressed by the project software located at the core of the system: BIMSCore.

The BIMSCore provides a software library which gathers all domain functionalities in a unified package. This library can be imported in other environments to use it. The presentation issues must be addressed by other software project. Accordingly, the BIMSCore library is accessed using a public interface which provides a centralized way to access to all methods. The BIMSCore project is divided in three parts, each one located in a different package. In the next section the architecture of BIMSCore project is presented in order to introduce the library design.

4.3.1 BIMSCore Architecture

The BIMSCore is a software project to built a library which gathers all domain context functionalities provided by BIMS. The project is divided in three parts, and everyone of these encapsulate a set of classes and interfaces as it follows:

1. **BIMSCore Model**. It contains a set of classes to represent the domain model environment of the BIMSCore project.
2. **BIMSCore Service**. It contains a set of interfaces and classes to provide service tasks over the classes of the BIMSCore Model package.
3. **BIMSCore DAO**. It contains a set of classes and interfaces to provide an encapsulation layer to manage all database operations.

Figure 4.4 shows a block diagram to illustrate the three parts of the BIMSCore software project. The BIMSCore Model package contains a set of classes which represent the elements of the domain environment. An example of this domain representation is the *User* element, which

[1] Create / Retrieve / Update / Delete

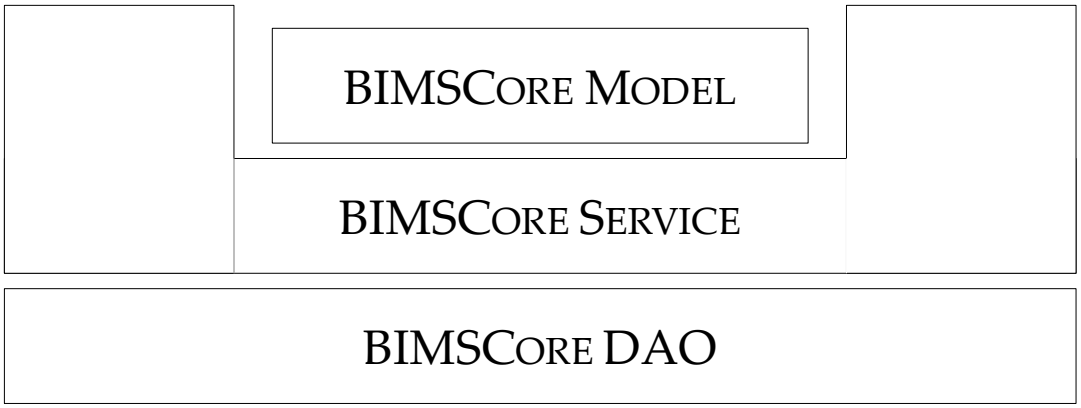


Figure 4.4: BIMSCore achitecture

represents a person who uses the system. It is a typical mapping between the domain environment, and the system being developed. The objective is to put all elements of the domain context, into BIMSCore Model package, without any extra-functionality, obtaining only a representation of the domain. Section 4.3.2 details the design of this package.

The BIMSCore Service package gathers all functionalities available to manage the classes of BIMSCore Model package. In order to clearly organize the package, every class of the BIMSCore Model package has a service provider class, which gathers all related functionalities. The following example illustrates the relationship between the two packages: the BIMSCore Model package contains a class named *User*, an object of the model domain to represent a person who uses the system, and all functionalities over the *User* object, like searches, deletes, modifications (typical CRUD operations), are provided by a class of the BIMSCore Service package called *UserManager*. All the service features of classes located in BIMSCore Model package, are implemented by the classes of BIMSCore Service package. The BIMSCore Service package is further explained in section 4.3.3.

The BIMSCore DAO package provides a high level encapsulation of database operations. The objective is to design a low level layer to hide the persistence issues, and avoid to operate using SQL statements, or against an specific database engine. The design structure of BIMSCore DAO is similar to BIMSCore Service package. Thereby, there is an specific DAO class in BIMSCore DAO, for every object in the context domain in the BIMSCore Model package. To make this description more illustrative, the same example can be used. In the BIMSCore Model package there is a model object named *User*, its objective is represent a real person who uses the system. Then in the BIMSCore Service package there is a class named *UserManager*, and its mission is to provide a set of functions over the User element. And finally, in order to encapsulate all database opertions with the element User, a class named *UserDAOManager* is located in BIMSCore DAO, in order to hide all persistence related operations, avoiding the use of SQL[2] queries. The BIMSCore DAO package is described in the section 4.3.4.

The figure 4.5 is an UML package diagram, where appears the three packages described above.

[2]Structured Query Language

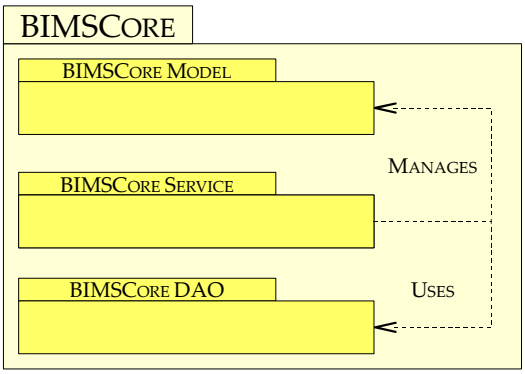


Figure 4.5: BIMSCore package level architecture

4.3.2 BIMSCore Model

The BIMSCore Model package is designed to represent the domain environment. To accomplish this, it is necessary to do a mapping process, to represent information of the context domain elements in the software system. Traditionally, the software engineering has based this mapping process on a method named *single level modeling* approach, which consists of creating a model object class for every element of the domain [13, 1, 3]. For this, during the development process, the software engineer identifies the elements of the domain environment which contains essential information, and creates an object class in the software system. This is a typical mapping procedure, commonly performed in systems currently in production. This approach, which is widely accepted in other domains, is not satisfactory in medical environments [12, 10]. The clinician domain is characterized by three main aspects [20]:

- Large. A well known ontolgy, SNOMED-CT, contains over 350.000 atomic concepts and 1.5 million relationships.
- Complex. Different views of information.
- Open-ended. There are numerous advances in clinical research, which constantly update clinical practice.

Using the single level modeling approach it is necessary to perform a direct relationship between every element of the context domain environment and an object class of the system. Due to the clinical context is a large environment, it is necessary to do a remarkable effort to represent all elements in a software system.

Two level modeling

Currently, there is another approach in order to model large, and complex domains, like clinical environments. It is named the *two level modeling* [13, 1] approach. The mission of this modeling method is to separate the information and the knowledge into two levels [1]. The principle of the two level modeling approach is to define a first level composed of a set of simple classes, which have an abstract meaning. This first level must be small in size, in order to be comprehensible,

and contain only non-volatile concepts in order to be maintainable. The behavior of these simple classes are constrained by a set of rules in a higher level, which represents the knowledge, and where concepts of the domain can be expressed. It is a promising approach to model complex domains resulting in flexible systems. The two level modeling has been used in some parts of BIMSCore design.

It is important to notice that the goal of this package is to provide a representation of the domain environment elements. Therefore, the classes of BIMSCore Model package do no contain any functionality, they are simple classes with *accessors* methods. All features, or functionalities which imply management of BIMSCore Model classes, are encapsulated in the BIMSCore Service package.

Figure 4.8 shows the UML class diagram of BIMSCore Model package. It contains the result of mapping all elements of the context domain to the BIMS. Some parts of the UML class diagram are designed following the one level modeling approach, while other parts are designed with the two level modeling methodology to provided a high level flexibility.

With the aim provide the possibility to manage a wide range a clinical projects types, some parts of the architecture has been developed with an high abstract meaning, using the two level modeling approach. The data type mapping of the fields are represented by the classes *Medical-Data*, and *Value* (see figure 4.6). *MedicalData* is a representation of the information associated to the field, like name, description, and other informational aspects (units, ...), and *Value* is a class to content the value filled by the clinician. The following example illustrates the mapping process performed: if the height of the patient is a field needed to be included in a group study, thus, the *MedicalData* class is instantiated to represent the concept of patient height (and informational aspects like units), and *Value* is an abstract class, instantiated as a *DoubleValue* at runtime. The flexibility of this design is based on the fact that a field only can contain a reduced set of data types (Boolean, String, Text, Date, Double), and *MedicalData* contains a set of directives to constrain the *Value* instantiation.

Figure 4.7 shows the flexible design developed to manage a wide range of structures based on a recursive inclusion of sections, which groups several *MedicalData* objects. The mapping process to represent the recursive groupping of fields results in a simple class named *Module*. With this design, the Module class is defined as a container of *Module* classes (recursive inclussion), and *MedicalData* classes. The last example exposed (height of the patient) could be used to illustrate this design decision, a *MedicalData* which represents the height of the patient could located in a *Module* object which represents the *body information* of a group study. Due to the this design, based on recursive inclusion, the *body information Module* could be included in a higher level *Module*.

The design exposed above is based on a set of highly, and simple classes, which a can adopt several instantiation forms. In order to constraint the form adopted, a set of rules could be defined. These rules, which represents the second level of the two level modeling approach, are injected at runtime into BIMS. Thus, it is possible to control the instantiation at runtime. Summarizing, this design allows to define different clinical studies, associated to very different clinical specialties, without stopping BIMS to modify its code.

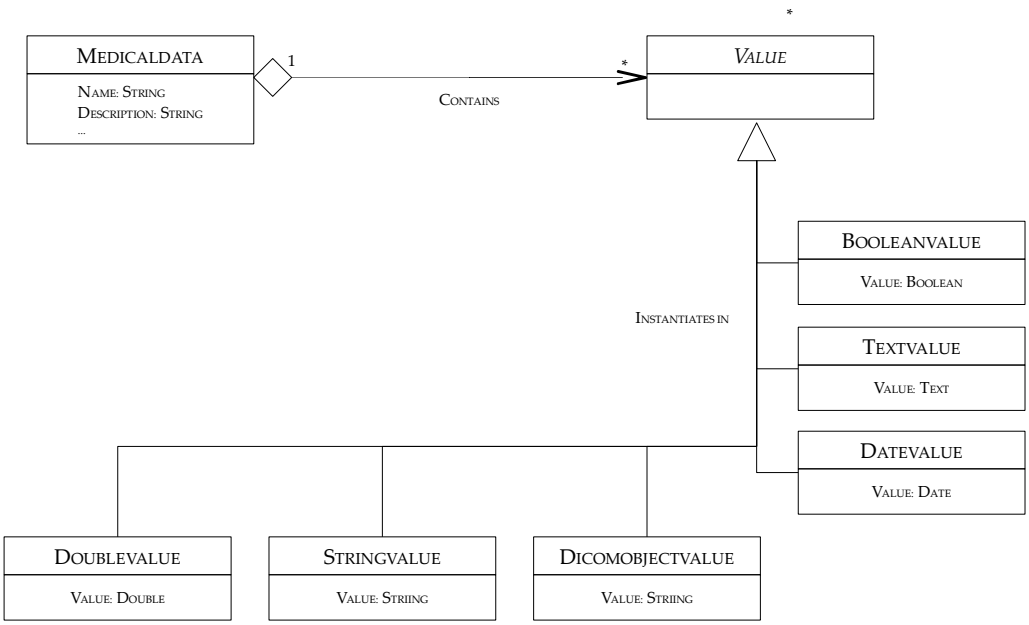


Figure 4.6: Relationship between MedicalData and Value

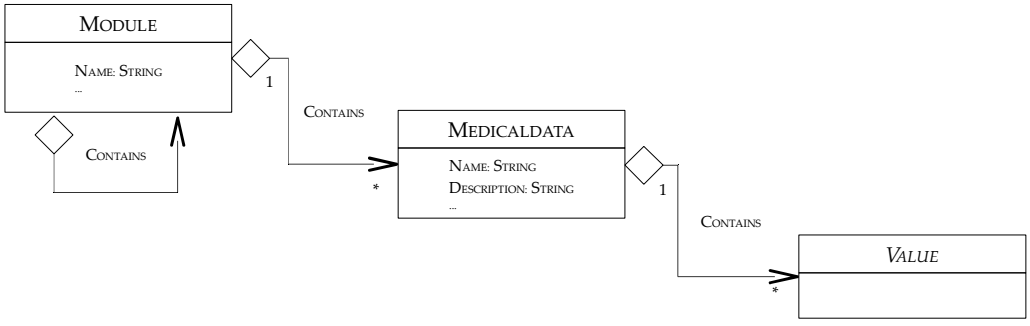


Figure 4.7: Relationship between Module and MedicalData

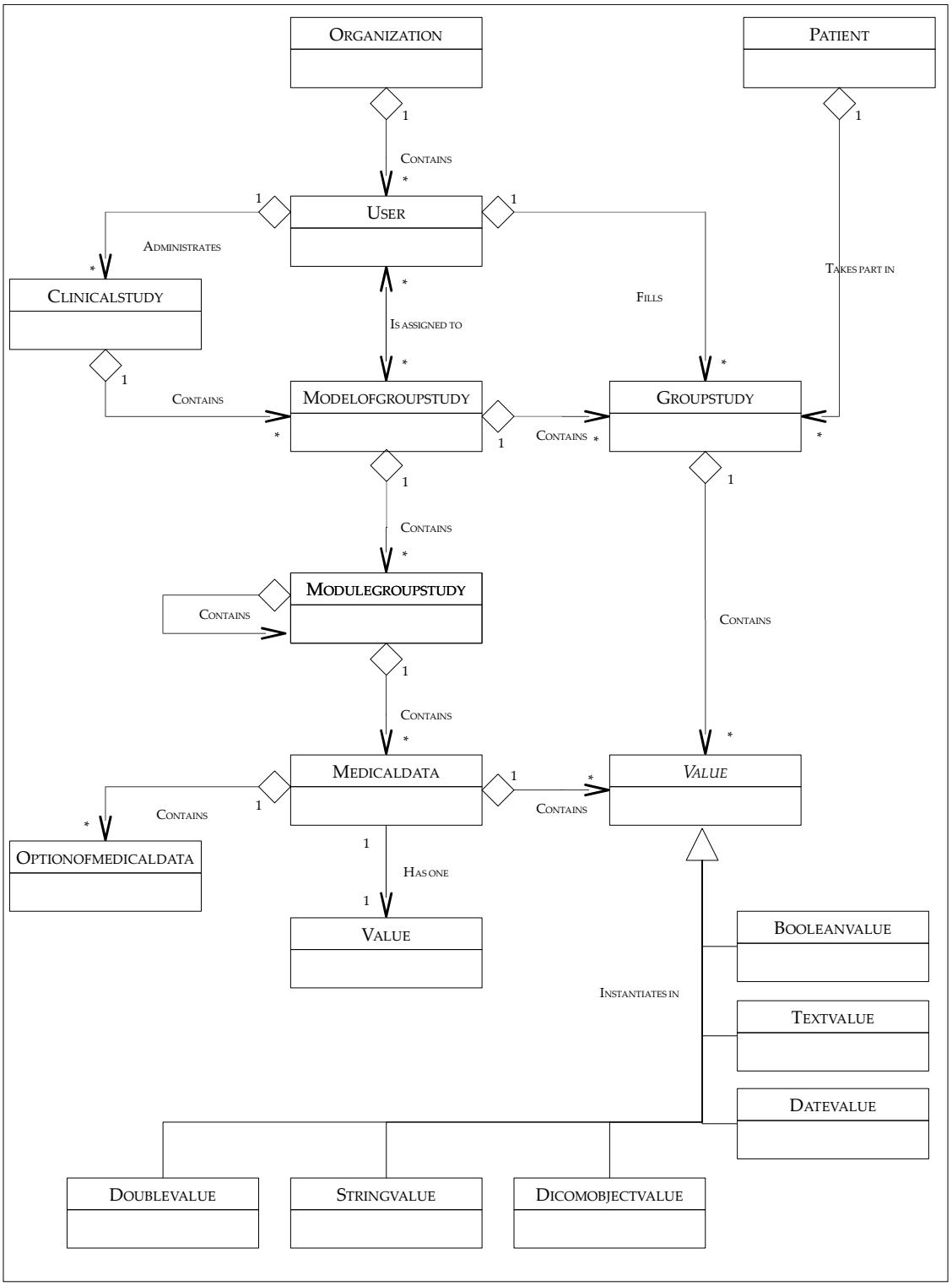


Figure 4.8: UML Class diagram of BIMSCore Model Package

4.3.3 BIMSCore Service

The BIMSCore Service package contains a set of interfaces and classes to provide management functionalities over BIMSCore Model package classes. These functionalities include the typical CRUD operations, and related management transactions. The division is based on the objective to make a real separation between the classes which represent elements of the domain, and the classes which provide functionalities, and management operations over the elements of the domain. Hence, classes and interfaces of the BIMSCore Service package provides management functionalities over BIMSCore Model classes.

With the aim to develop an understandable system, scalable, and easy to study, every model class of the BIMSCore Model package is managed by an specific class located in the BIMSCore Service package. In this case, interface - based programming has been applied, and for this, in the BIMSCore Service package a set of interfaces defines the available tasks which provides the package globally. Hence, there is a direct relationship between every class of BIMSCore Model package and the set of interfaces and classes of BIMSCore Service package. The following example illustrate the relationship between BIMSCore Model and BIMSCore Service package: in the BIMSCore Model there is a class named *User* which is a formal representation of a person who uses the system. All related management operations, like create a *User*, modify its information, delete it, are defined in a interface named *UserManager*, and this is implemented in a class called *UserManagerImplementation*. There is a naming convention used to facilitate the relationship between the content of BIMSCore Model package, and BIMSCore Service package. This naming convention is formally expressed in 4.2 for interface names, and in 4.1 for the class names:

$$InterfaceName_{BIMSCoreService} := ClassName_{BIMSCoreModel} + `Manager` \tag{4.1}$$

$$ClassName_{BIMSCoreService} := ClassName_{BIMSCoreModel} + `ManagerImplementation` \tag{4.2}$$

Following the expression 4.2 and 4.1, it is possible to build table 4.1 which represents the complete relationship between BIMSCore Model package and the interfaces, as well as the classes which define and implement the service tasks in BIMSCore Service.

With this approach, every model object of BIMSCore Model package has a related a service interface/class associated in BIMSCore Service package. This strict mechanism forces to the integration and the definition of all related functionalities of an specific class of BIMSCore Model package, in the associated interface/class of BIMSCore Service package. For example, all features related with the management of *User* model object, are defined in *UserManager*, and implemented in *UserManagerImplementation* (see table 4.1). In order to unify all functionalities under a unique service framework, an special interface/class is defined: *BIMSCoreEngine*, *BIMSCoreEngineImplementation*. The *BIMSCoreEngine* is an interface which defines all functionalities provided in the BIMSCore Service package, and the *BIMSCoreEngineImplementation* is its implementation. Using this approach, this package provides a unique entry point to access to all methods which provide service functionalities.

The figure 4.9 depicts the UML class diagram to the BIMSCore Service package.

BIMSCore Model	BIMSCore Service Interface	BIMSCore Service Implementation
Organization	OrganizationManager	OrganizationManagerImplementation
User	UserManager	UserManagerImplementation
Patient	PatientManager	PatientManagerImplementation
ClinicalStudy	ClinicalStudyManager	ClinicalStudyManagerImplementation
ModelOfGroupStudy	ModelOfGroupStudyManager	ModelOfGroupStudyManagerImplementation
ModuleGroupStudy	ModuleGroupStudyManager	ModuleGroupStudyManagerImplementation
MedicalData	MedicalDataManager	MedicalDataManagerImplementation
OptionOfMedicalData	OptionOfMedicalDataManager	OptionOfMedicalDataManagerImplementation
DoubleValue	ValueManager	ValueManagerImplementation
DateValue	ValueManager	ValueManagerImplementation
DICOMObjectValue	ValueManager	ValueManagerImplementation
TextValue	ValueManager	ValueManagerImplementation
StringValue	ValueManager	ValueManagerImplementation
BooleanValue	ValueManager	ValueManagerImplementation

Table 4.1: Relationship between BIMSCore Model and BIMSCore Service

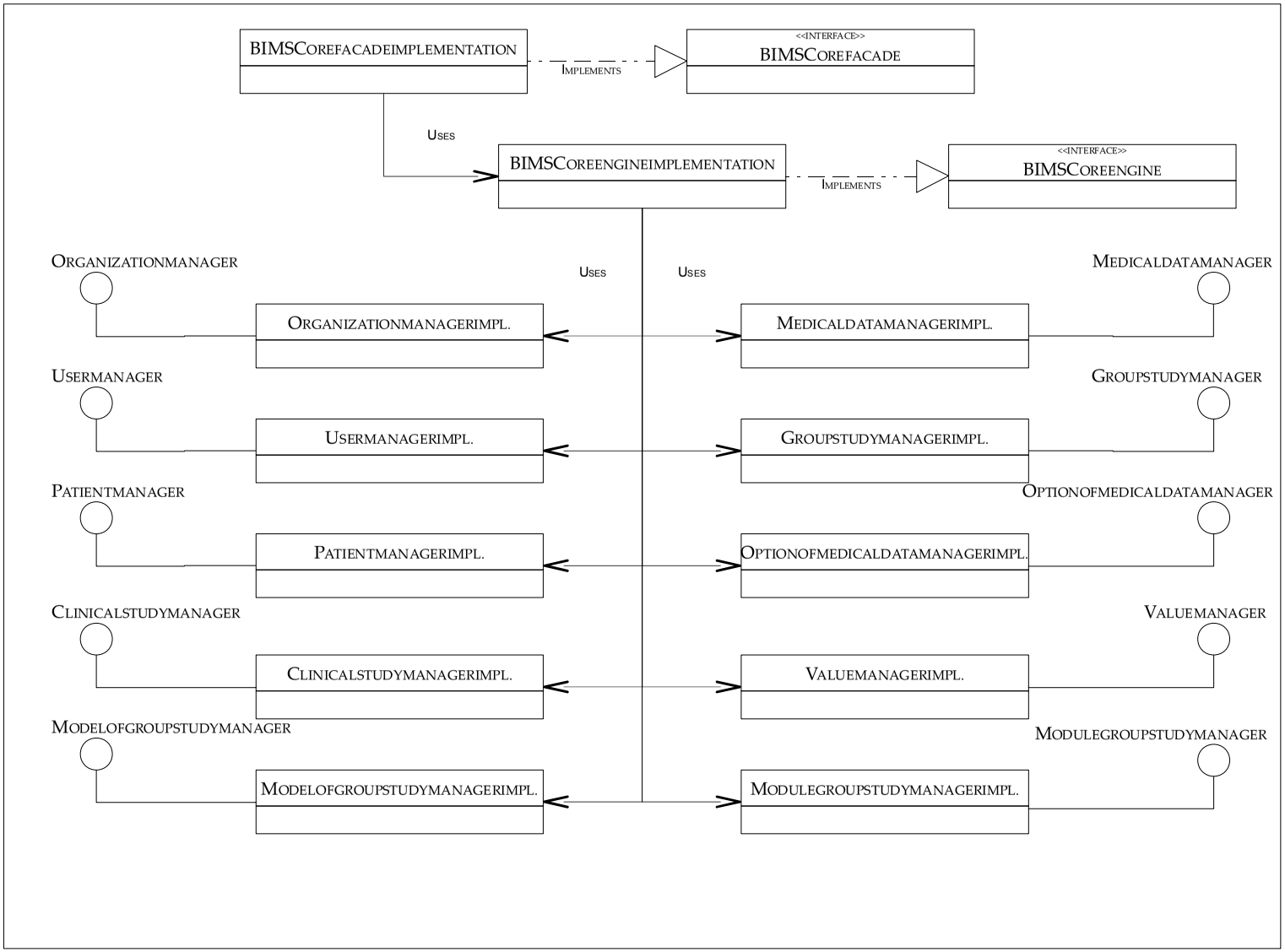


Figure 4.9: BIMSCore Service package UML Class diagram

4.3.4 BIMSCore DAO

The BIMSCore DAO package contains a set of interfaces and classes to encapsulate database operations. The main mission is to avoid the extensive use of SQL statements, and perform database operations oriented to model objects. For this, and following the same reasoning of BIMSCore Service package, an interface/class pair is defined in BIMSCore DAO package, for every class of BIMSCore Model. Following the same example used along this chapter, the *User* class of BIMSCore Model package has an interface called *UserManager*, and an implementation named *UserManagerImplementation* in BIMSCore Service package to provide management operations. Similarly, in the BIMSCore DAO package there is another pair interface/class, named *UserDAO-Hibernate*, *UserDAOHibernateImplementation* respectively, to provide a object level encapsulation to manage database operations. With this methodology, the whole BIMSCore software project does not contain any SQL statement, because all database operations are object-oriented transactions.

Figure 4.10 shows the UML diagram class of the interfaces and classes of BIMSCore DAO package. It is a simple diagram because there are not complex relationships between the classes and interfaces, and it is mainly composed by interface/class pairs, all of these are associated to the classes of BIMSCore Model package. The expression of line equation 4.3 shows the naming convention for interfaces of BIMSCore DAO package. It is based on the name of BIMSCore Model class, and the suffix 'DAOHibernate'. The naming convention of the classes of BIMSCore DAO package is expressed in 4.4, and it is based on the suffix 'DAOHibernateImplementation'.

$$InterfaceName_{BIMSCoreDAO} := ClassName_{BIMSCoreModel} + `DAOHibernate` \quad (4.3)$$

$$ClassName_{BIMSCoreDAO} := ClassName_{BIMSCoreModel} + `DAOHibernateImplementation` \quad (4.4)$$

The table 4.2 shows the complete relationship between BIMSCore Model and BIMSCore DAO package expressed in 4.3 and in 4.4.

4.3.5 Anonymisation mechanism

The data managed by BIMSCore contains crucial information, which can be used to identify a patient. This information is named PHI[3]. The HIPAA[4] defines Protected Health Information as *individually identifiable health information that is transmitted or maintained in any form or medium by a covered entity* [9]. Therefore, it is necessary to manage carefully all data (in DICOM headers, and clinical studies) to anonymize all critical information which could fall under the PHI umbrella.

Currently, there are several approaches to anonymize PHI based on public/private key encryption mechanisms [15]. This approach has two objectives. Firstly, delete all PHI of clinical study, and secondly, provide a method to follow up an patient who has participated in several studies, because the pseudonym identifier assigned in all clinical studies is the same.

[3]Protected Health Information

[4]Health Insurance Portability and Accountability Act

BIMSCore Model	BIMSCore DAO Interface	BIMSCore DAO Implementation
Organization	OrganizationDAOHibernate	OrganizationDAOHibernateImplementation
User	UserDAOHibernate	UserDAOHibernateImplementation
Patient	PatientDAOHibernate	PatientDAOHibernateImplementation
ClinicalStudy	ClinicalStudyDAOHibernate	ClinicalStudyDAOHibernateImplementation
ModelOfGroupStudy	ModelOfGroupStudyDAOHibernate	ModelOfGroupStudyDAOHibernateImplementation
ModuleGroupStudy	ModuleGroupStudyDAOHibernate	ModuleGroupStudyDAOHibernateImplementation
MedicalData	MedicalDataDAOHibernate	MedicalDataDAOHibernateImplementation
OptionOfMedicalData	OptionOfMedicalDataDAOHibernate	OptionOfMedicalDataDAOHibernateImplementation
DoubleValue	ValueDAOHibernate	ValueDAOHibernateImplementation
DateValue	ValueDAOHibernate	ValueDAOHibernateImplementation
DICOMObjectValue	ValueDAOHibernate	ValueDAOHibernateImplementation
TextValue	ValueDAOHibernate	ValueDAOHibernateImplementation
StringValue	ValueDAOHibernate	ValueDAOHibernateImplementation
BooleanValue	ValueDAOHibernate	ValueDAOHibernateImplementation

Table 4.2: Relationship between BIMSCore Model and BIMSCore DAO

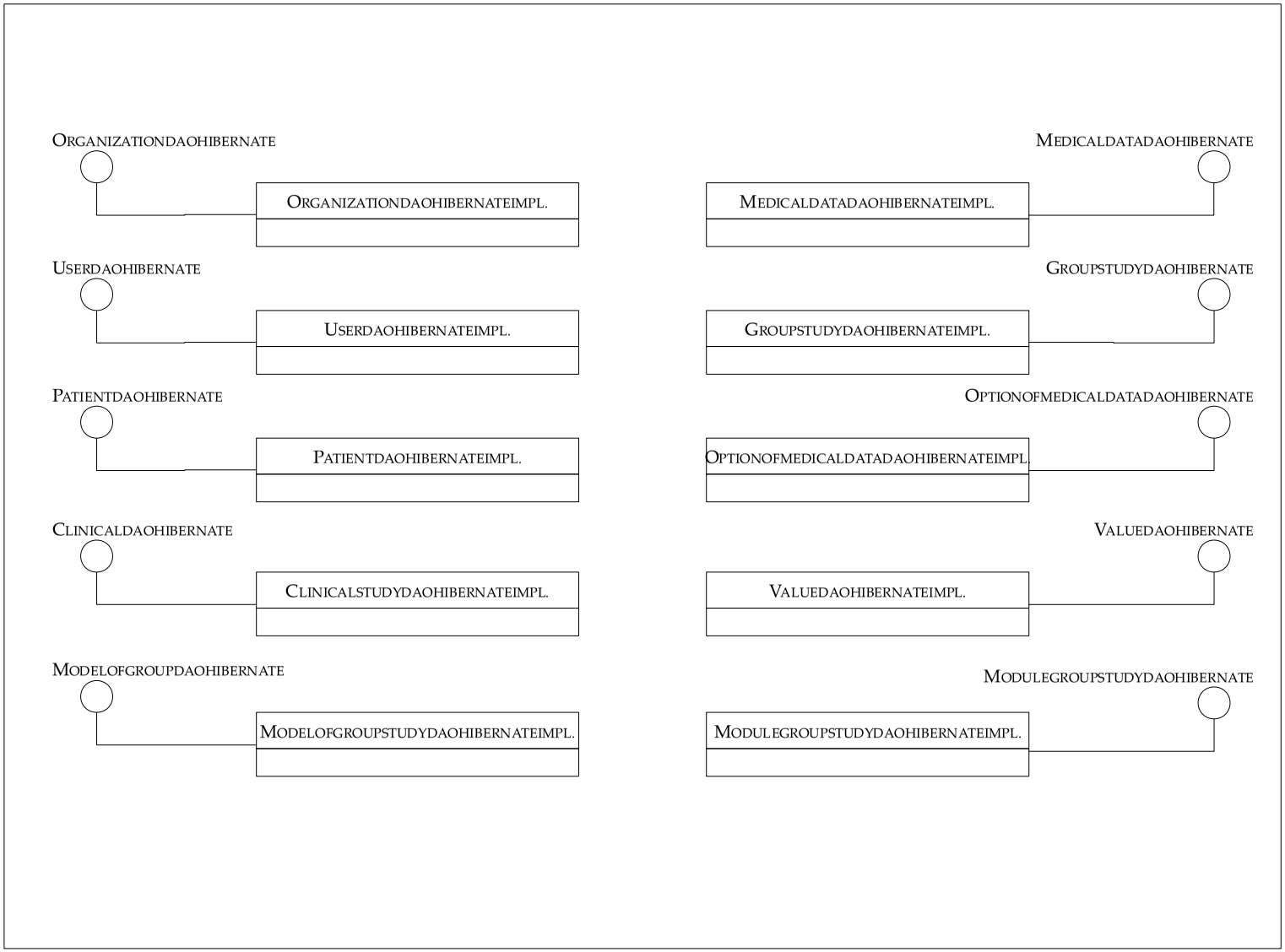


Figure 4.10: BIMSCore DAO package UML Class diagram

In BIMS environment the anonymisation mechanism is based on a look-up table, where the clinician records a tuple for every patient who takes part in a clinical study. This tuple is composed by a well-known identifier which acts as a seed (the BIMS encourage to use national patient identifier), and a MD5 hash pseudonym generated by BIMS from the seed. This mechanism has been proposed in similar web-based environments [4], and generates the same pseudonym for a patient who takes part in several clinical studies since the same seed would be used. Hence, BIMS only stores a pseudonym generated from a MD5 hash, and the clinician has the responsibility to maintain the look-up table, in order know the relationship between the patient pseudonym and the patient identity.

One of the disadvantage of the BIMS anonymizer mechanism is the possibility of losing the look-up table maintained by the clinician. If the clinician loses the table that keeps the relationship between the pseudonym, generated by the BIMS, and the national patient identifier, it will not be possible to follow-up on a patient using its pseudonym.

4.4 BIMSWebAccess. The second layer of BIMS architecure

The second level of BIMS architecture is a project named BIMSWebAccess, which defines a structured access to the BIMS. The objective is to isolate in a self-sufficient project all the presentation issues of BIMS, without being concerned about internal features of the system. In this case, and with the objective to fulfill accessibility requirements described in chapter 2, BIMSWebAccess is a web - based application which provides access to BIMS through an standard web browser.

BIMSWebAccess project is composed of two layers:

1. **Presentation layer**. It is responsible for managing all view related aspects.

2. **Business layer**. It establishes a communication way between the user actions in the web interface, and the features provided by BIMSCore project.

4.4.1 Presentation layer

The presentation layer manages all aspects related with the visualization. The objective is to build a simple, intuitive and usable web interface to interact with the user. The organization of the interface elements like menus, lists, buttons, and accessibility items, is described in this section.

The visual structure of the presentation layer is defined using a global template. This technique allows to specify the visual layout of the interface in a unique point of the project. Thus, all pages of the project are visually structured following the same pattern, coded in the template of the BIMSWebAccess project.

Figure 4.11 shows the visual structure used in whole system. The structure is simple, with the objective to facilitate the user navigation. It is composed of four parts:

- **Title**. This part contains the title of the system: *BIMS: Biomedical Information Management System*, and a couple of support icons, one to identify the user logged, and the other to logout. Figure 4.12 shows an snapshot of this part.

- **General menu**. This part contains a set of buttons to execute the available operations. Figure 4.13 shows an snapshot of this part, which contains four buttons:

    1. *Home*. To access to the welcome page of the system.
    2. *My Studies*. To access to the global list of the user logged studies
    3. *Start*. To start a new study.
    4. *About*. General information about BIMS.

- **Lists, tables**. This part is reserved to contain tables, or list of items. Figure 4.14 shows an snapshot of this section.

- **Working area**. This is part is used to place information requested by the user, like information about a clinical study. Figure 4.15 shows an snapshot of this part, and figure 4.16 shows the same part containing a group study.

The motivation behind of using a global template to represent the visual organization is to encapsulate the visual structure details in one point of the BIMSWebAccess project. This provides two main advantages:

1. **Avoids code repetition**. The code used to define the visual organization is only located in the template.
2. **Facilitates maintenance/modification operations**. If the visual organization information of BIMSWebAccess is only located in a unique point of the project, tasks like modifications, are performed easily.

These advantages facilitate the development of the visual structures to build a intuitive, and usable web interface to interact with the system.

Implementation

The presentation layer of BIMSWebAccess has been implemented using a set of JSP[5] with Tobago technology (see section 3.4.1). Figure 4.17 shows an snapshot of the system, where the interface structure showed in the figure 4.11 can be appreciated.

4.4.2 Business layer

The business layer of BIMSWebAccess project provides an interaction way to link user actions in web interface (presentation layer) and functionalities provided by BIMSCore project (see section 4.3). The mission of business layer is to define a bidirectional communication mechanism to transport the information between the person, who is using the system, and the BIMSCore project, the service provider of BIMS.

The business layer design is simple and it is composed of a set of isolated objects which receive the user actions, through the web interface, and call the methods provided by BIMSCore library.

[5]Java Server Pages

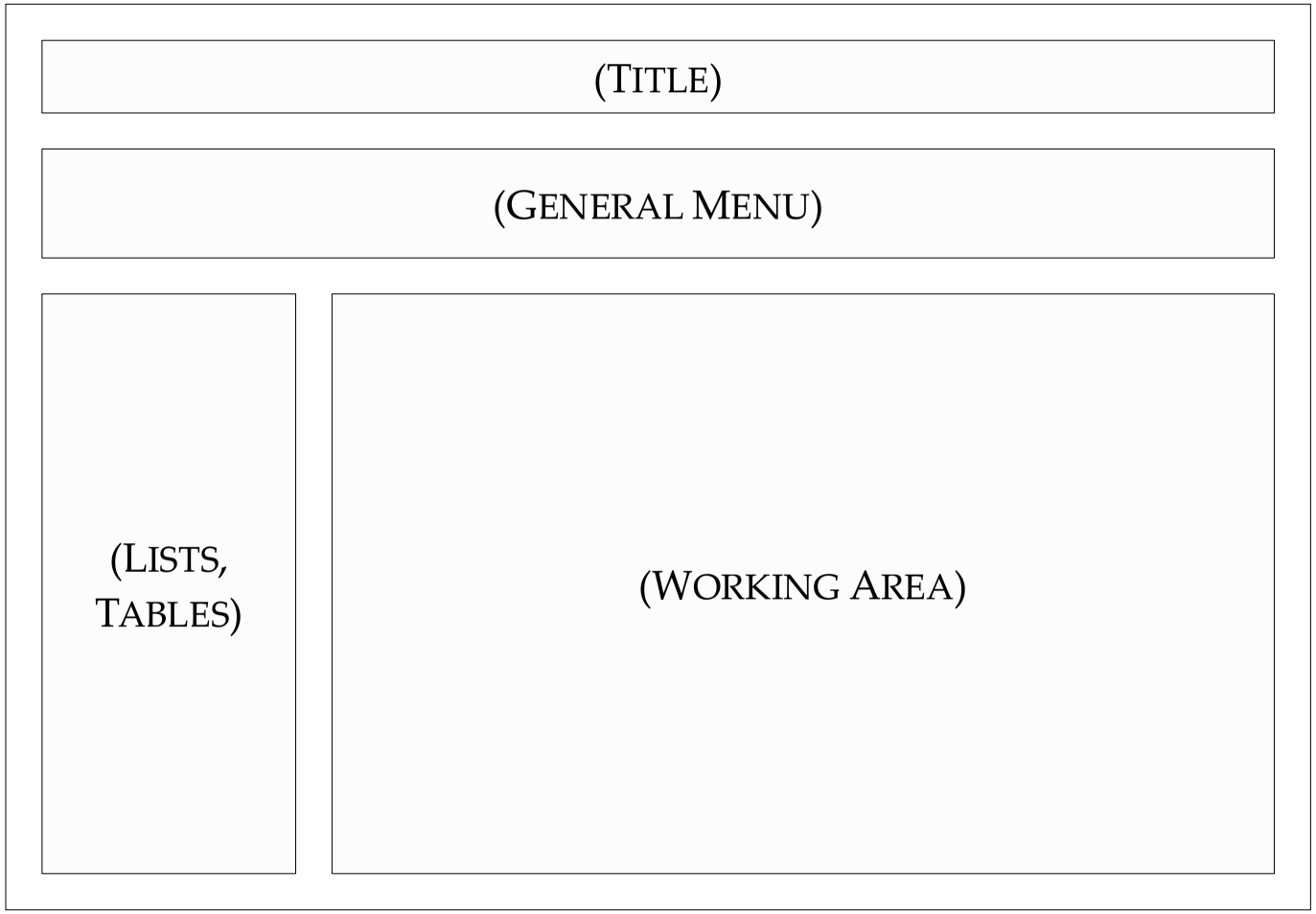


Figure 4.11: Organization of the presentation layer

Figure 4.12: Snapshot of the Title

Figure 4.13: Snapshot of the General Menu

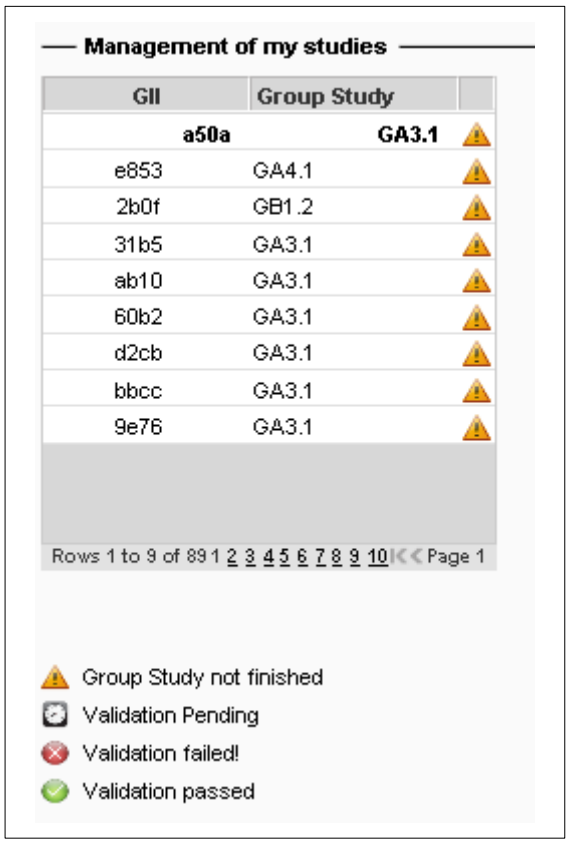


Figure 4.14: Snapshot of the list of studies of the logged user

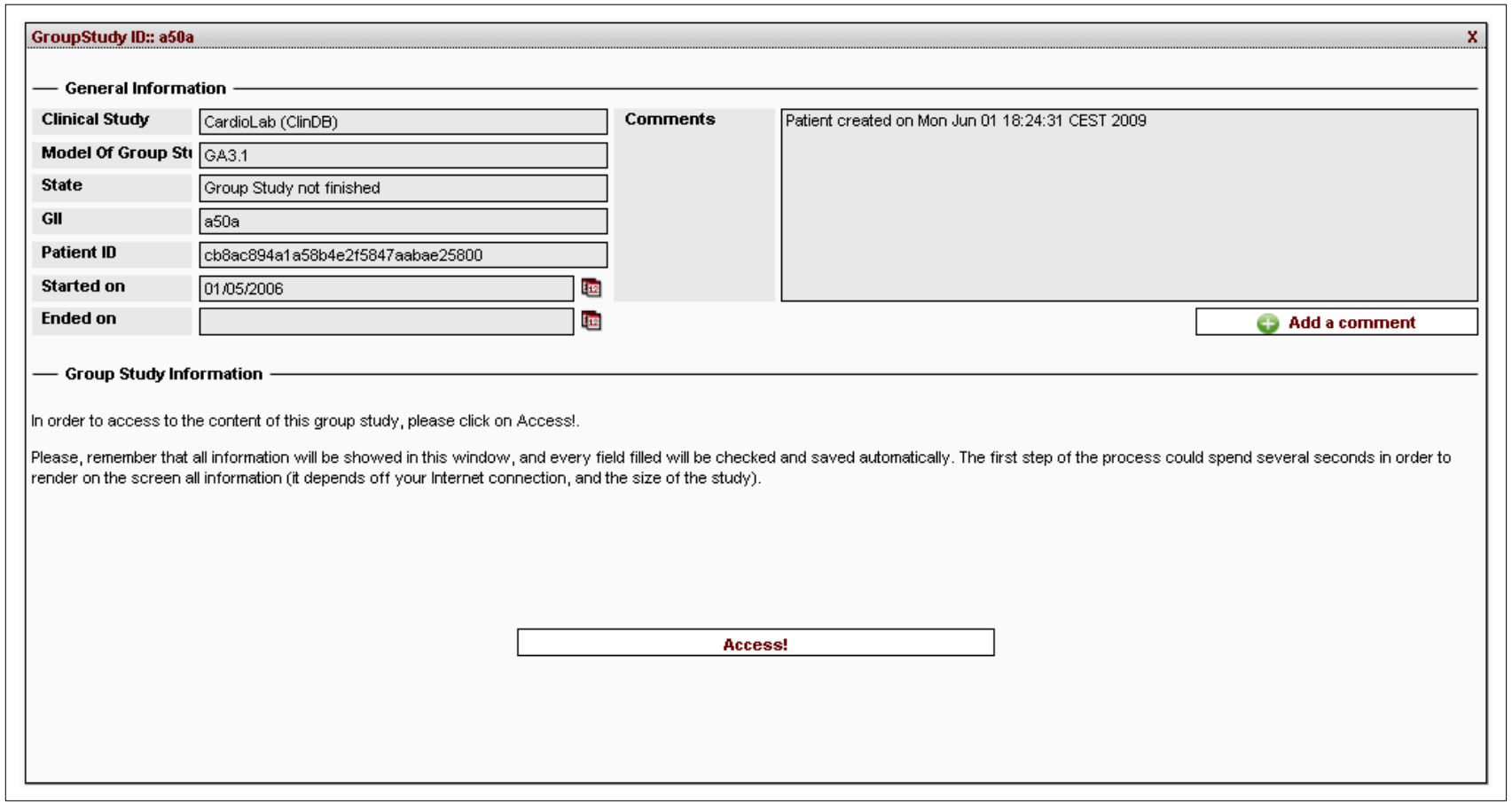


Figure 4.15: Snapshot of the working area

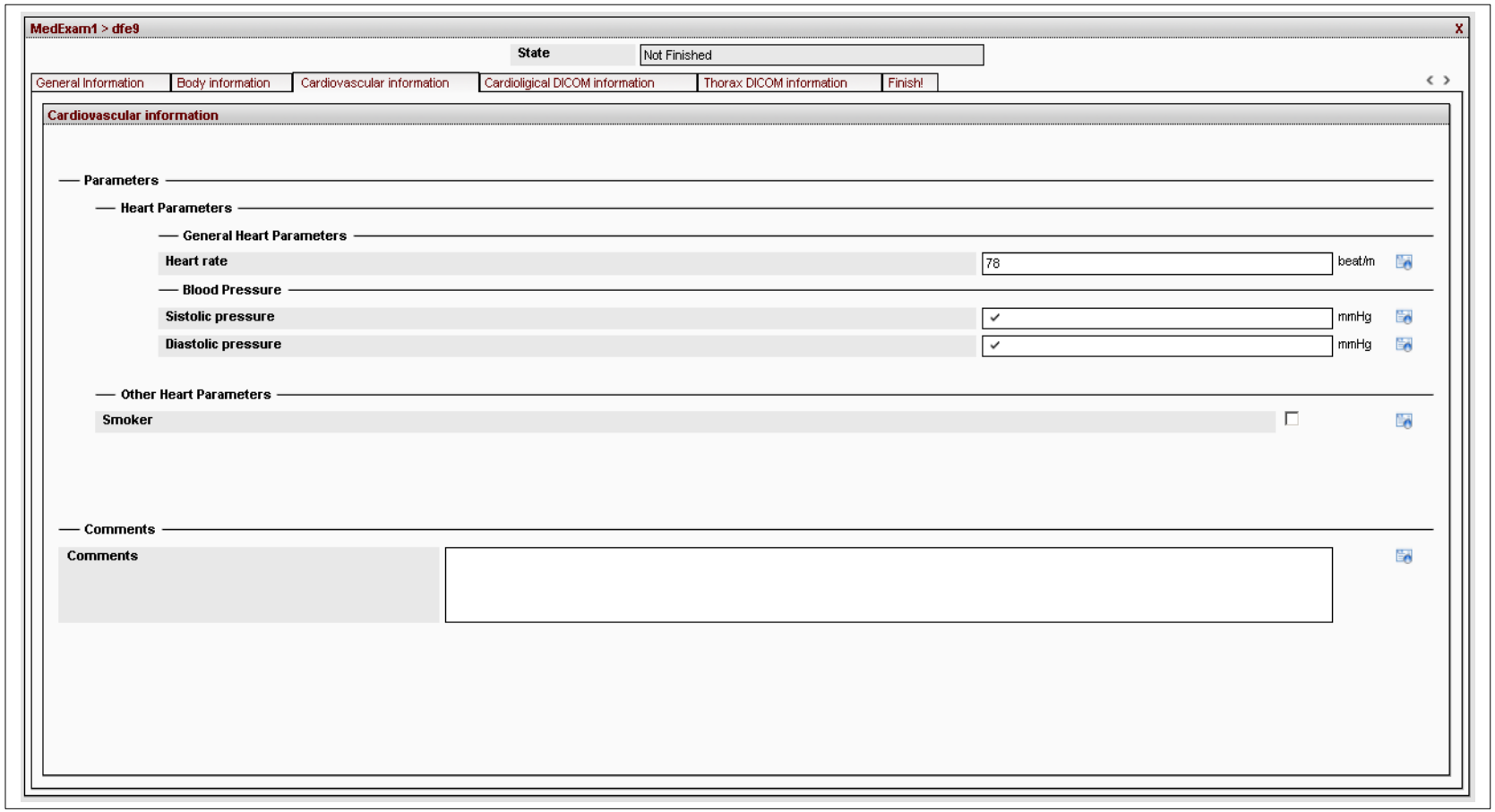


Figure 4.16: Snapshot of the working area containing a group study

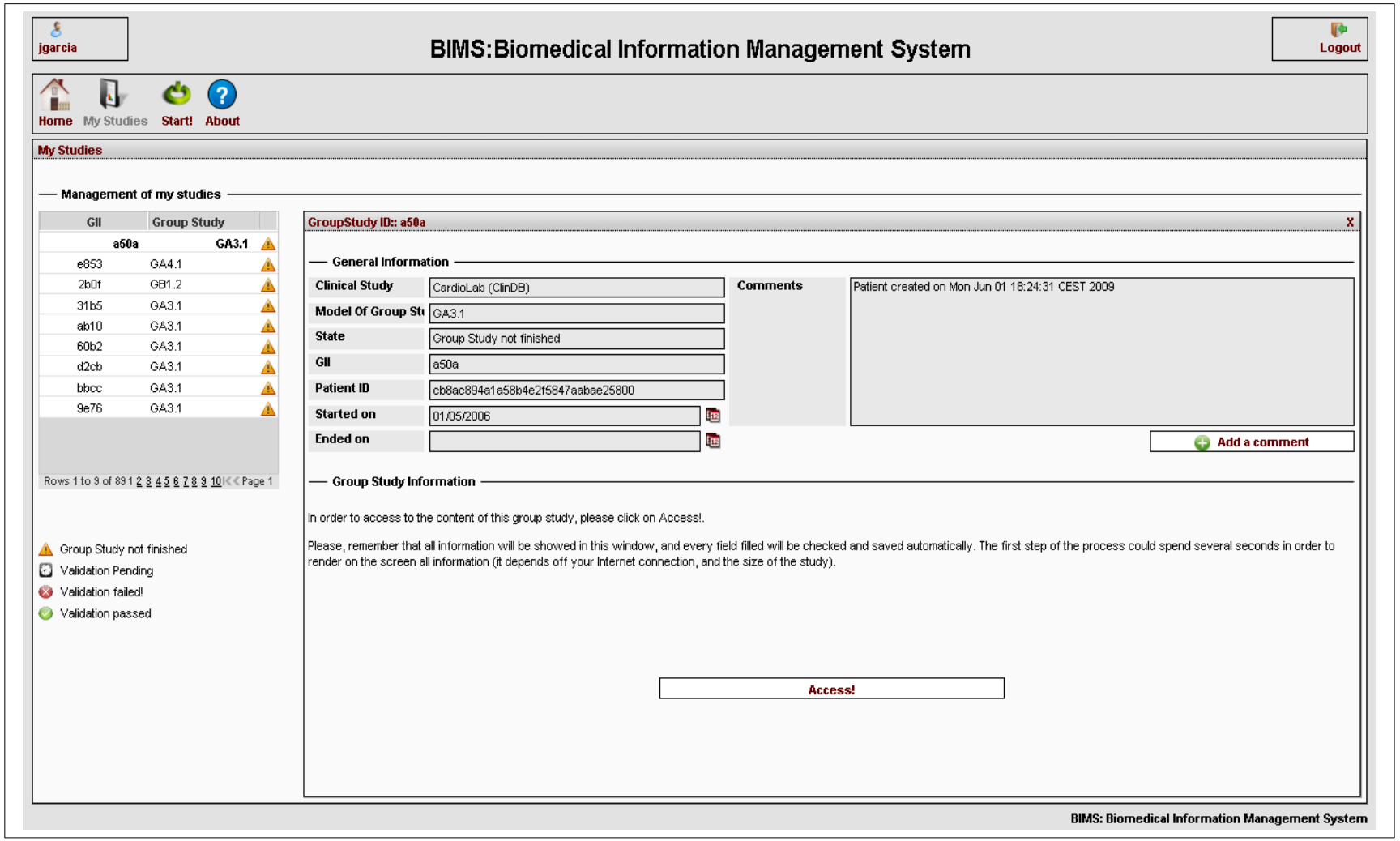


Figure 4.17: Snapshot of BIMS

Similarly to a bridge which communicates two sides of a river. Even though it is possible to group all functions of BIMSWebAccess business layer in a single object, the modularity principle applied along the whole project is applied again, in order to separate, and concentrate similar functionalities in the same object of the business layer.

The following list contains all objects of business layer. Moreover, a briefly description included.

- **BIMSWebAccessControllerClinicalStudyManager**. Groups all calls to BIMSCore related with the management of clinical studies.
- **BIMSWebAccessControllerMyStudiesManager**. Groups all calls to BIMSCore related with the management of group studies entered by an specific clinician.
- **BIMSWebAccessControllerOrganizationAdministrationManager**. Groups all calls to BIMSCore related with the management of organizations included BIMS research projects.
- **BIMSWebAccessControllerStartStudyManager**. Groups all calls to BIMSCore related with the requesting management operations of a group study.
- **BIMSWebAccessControllerUserAdministrationManager**. Groups all calls to BIMSCore related with the management of the BIMS users.
- **BIMSWebAccessUserManager**. Groups all calls to BIMSCore related with the profile information of the BIMS users.

Summarizing, the main mission of the business layer of BIMSWebAccess project is to provide a link between actions of the user in the web interface, and the respective associated calls to BIMSCore project library.

Chapter 5

ClinDB2BIMS: CardioLab Migration

ClinDB2BIMS is a project which was created in order to perform a thorough and realistic testing on BIMS. The mission of the project is to migrate all CardioLab information into BIMS. The CardioLab platform is an ongoing initiative whose main goal is the development of tools for image - based computational analysis, modeling and simulation of the cardiac function, under the CDTEAM[1] project. For this reasoning, a good test to demonstrate the strength of BIMS is to migrate all information maintained by CardioLab into BIMS.

5.1 Introduction

Figure 5.1 illustrates the migration process to copy all CardioLab platform information, stored in a relational database called ClinDB, into BIMS. CardioLab platform uses MySQL[2] technology as DBMS.

Normally, a migration process must be designed specifically for the environments which take part in the operation. Accordingly, it is necessary to define an strict mechanism to link, in a one - to - one relationship, the structures of the source and the destination information repositories. In addition, and due to the fact that the information stored in ClinDB is of poor quality, it is necessary to improve upon it, before dumping it into BIMS. Otherwise, the migration process will not be useful because poor data quality of ClinDB will be propagated to BIMS, referring to the principle known in the database world as *garbage - in, garbage - out*.

5.2 Methodology

ClinDB2BIMS is a project based on a command line application, which connects to ClinDB database, extracts the medical information, transforms it, and loads it into BIMS. This is a well known methodology, named Extract - Transform - Load (ETL) [29], widely used in migration processes in the data warehousing applications context.

The first step of the migration process is to study and model the source information reposi-

[1] Consorcio para el Desarrollo de Tecnologías Avanzadas para la Medicina - http://www.suinsa.com/cdteam/
[2] http://www.mysql.com

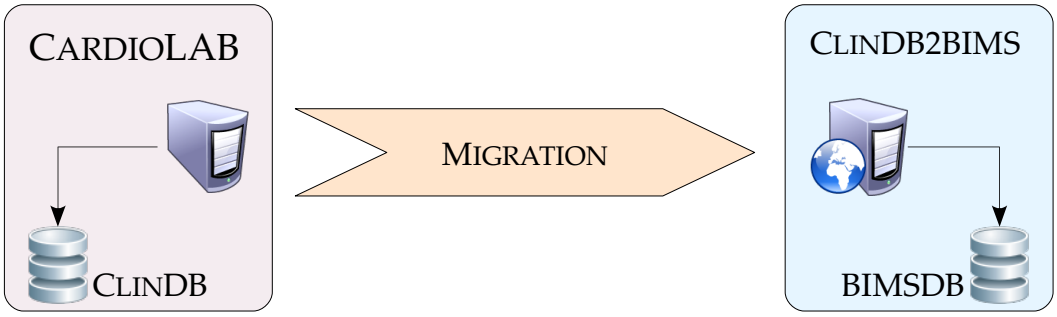


Figure 5.1: Migration process

tory: ClinDB database. Therefore, it is necessary to analyze the database schema of ClinDB. The ClinDB schema is simple, and stores all fields of an specific group study in the same row, where every column represents a field of the group study data collected. This is a flat form to storage information because the structure of the CRF is not saved. For these reasons, it is feasible to define an statement to retrieve all fields in a single query execution. The execution of this query, used to retrieve all fields of a group study, is the first stage of ETL approach: *Extract*.

The second step of the migration process is to define an predefined mechanism to link every one of the columns, retrieved with the extraction query, with a unique *MedicalData* object of BIMS environment. For example, if the extraction query retrieves a field from ClinDB which represents the height of the patient, it is necessary to find the *MedicalData* object of the study which represents the height in BIMS environment. Typically this approach is performed writing complex lists of nested if statements, to establish a rudimentary transformation from the source to the destination information repository. ClinDB2BIMS proposes a new approach to perform this step in a easier way, avoiding the endless lists of concatenated if statements. The method is based on following the same naming convention in ClinDB database and BIMS. Thus, if a column of ClinDB is named *height*, it is necessary to name the *MedicalData* object in BIMS, in the same form (property defined by a rule in the XML file which describes the second level of the modeling paradigm followed in BIMS, see section 4.3.2). This methodology, based on use the same name, facilitates the link creation to copy information from ClinDB into BIMS.

After the relationship between source and destination is established, it is possible to define a set of features to improve the data quality of the source information repository (ClinDB database). ClinDB2BIMS defines a set of methods to transform all ClinDB information, stored as a String type, in a more specific type in BIMS. For example, boolean fields, which are stored in ClinDB as a String (*yes* / *no*), are transformed to a boolean data type in BIMS environment. This step is the *Transform* stage in ETL approach.

The last part of the migration is to load data in the destination repository information. To this end, the functionalities provided by BIMSCore project are used (see section 4.3). This is *Load* stage in ETL approach.

Summarizing, the ClinDB2BIMS project proposes a methodology to migrate medical information contained in CardioLab platform into BIMS environment. This methodology is a semi-automated process, which avoids writing endless conditional statements to control all migration

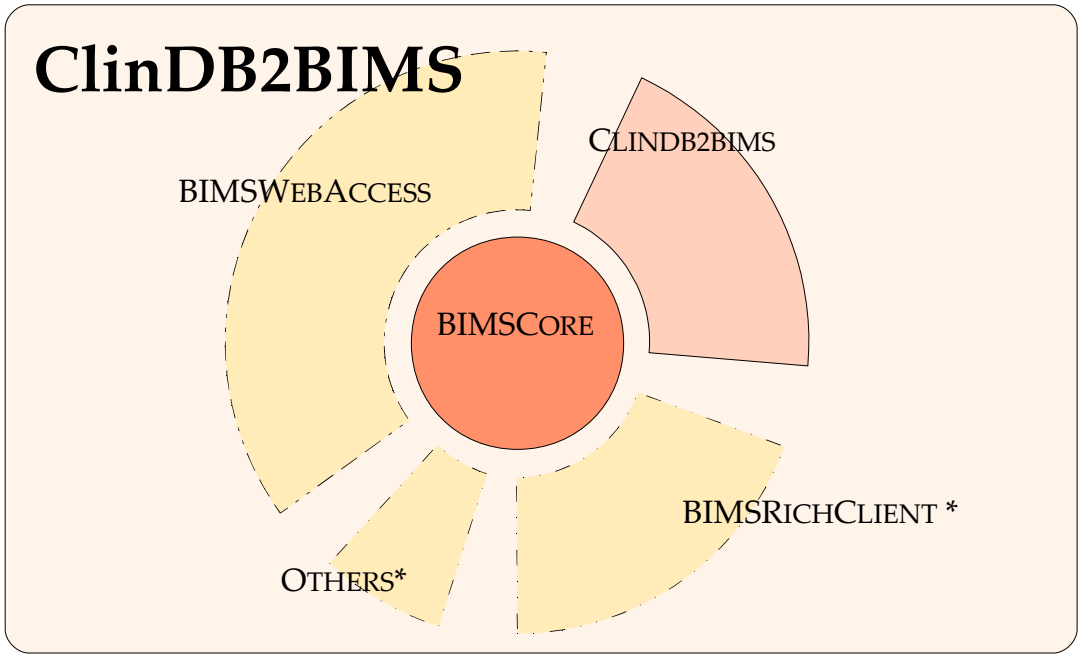


Figure 5.2: ClinDB2BIMS high level architectural view

details. This methodology is flexible, and it can be used to migrate other information repositories into BIMS.

5.3 Design

ClinDB2BIMS project design is an extension of BIMS project. The modularity of BIMS design, which divides the system in two layers, provides a strict encapsulation of the functions in one of the two levels. Using this approach, ClinDB2BIMS has been designed as a software application to be located at the second layer, using the functionalities provided by BIMSCore project.

Figure 5.2 illustrates the position of ClinDB2BIMS in BIMS design. The software project is located at the access layer, and uses all functions provided by the unified library of the BIMSCore project. Following this design, the responsibility of ClinDB2BIMS is reduced to manage the first and the second step of the migration methodology described in 5.2:

1. **Extract**. In the first step, ClinDB2BIMS must be able to query the system to retrieve an specific list of group studies to copy into BIMS.

2. **Transform**. In the second step, ClinDB2BIMS has to perform some actions to improve information data quality of ClinDB.

The last step of the process is to load all information into BIMS environment. To do this, ClinDB2BIMS uses the functionalities provided by BIMSCore (see section 4.3).

Figure 5.3 shows the UML class diagram of ClinDB2BIMS project. It is a simple design composed of a main object (*ClinDB2BIMS*), and a set of objects which gather all migration features. With the aim to define an clear separation, and facilitate the detailed understanding of this independent project, an specific class is responsible for migrating all the sections that built up an specific group study in CardioLab. Thereby, because a group study in CardioLab platform is composed of eleven sections, there are eleven objects to provide migration functions in ClinDB2BIMS. The name convention used to link the class with the part of the study to migrate, is expressed in 5.1.

$$ClassName_{ClinDB2BIMS} := SectionName_{CardioLab} + `Migrator` \tag{5.1}$$

Therefore, the class named F0DIAGNOSISMIGRATOR is responsible for providing a set of functions to migrate a part of the group study named *F0Diagnosis*. Even though this separation is not structly necessary, facilitates the study of this project in order to extend it to another environments.

5.4 Results and conclusions

ClinDB2BIMS has migrated all medical data of ClinDB database into BIMS successfully. Moreover, a set of checks have been applied during the migration, in order to improve the data quality of the CardioLab platform information.

Summarizing, this project demonstrates the strength of BIMS, improves the information data quality, and facilitates the access to the CardioLab platform using BIMS. Moreover a flexible methodology, based on the ETL approach, has been developed to perform migration tasks, without developing software applications based on never-ending nested conditional statements.

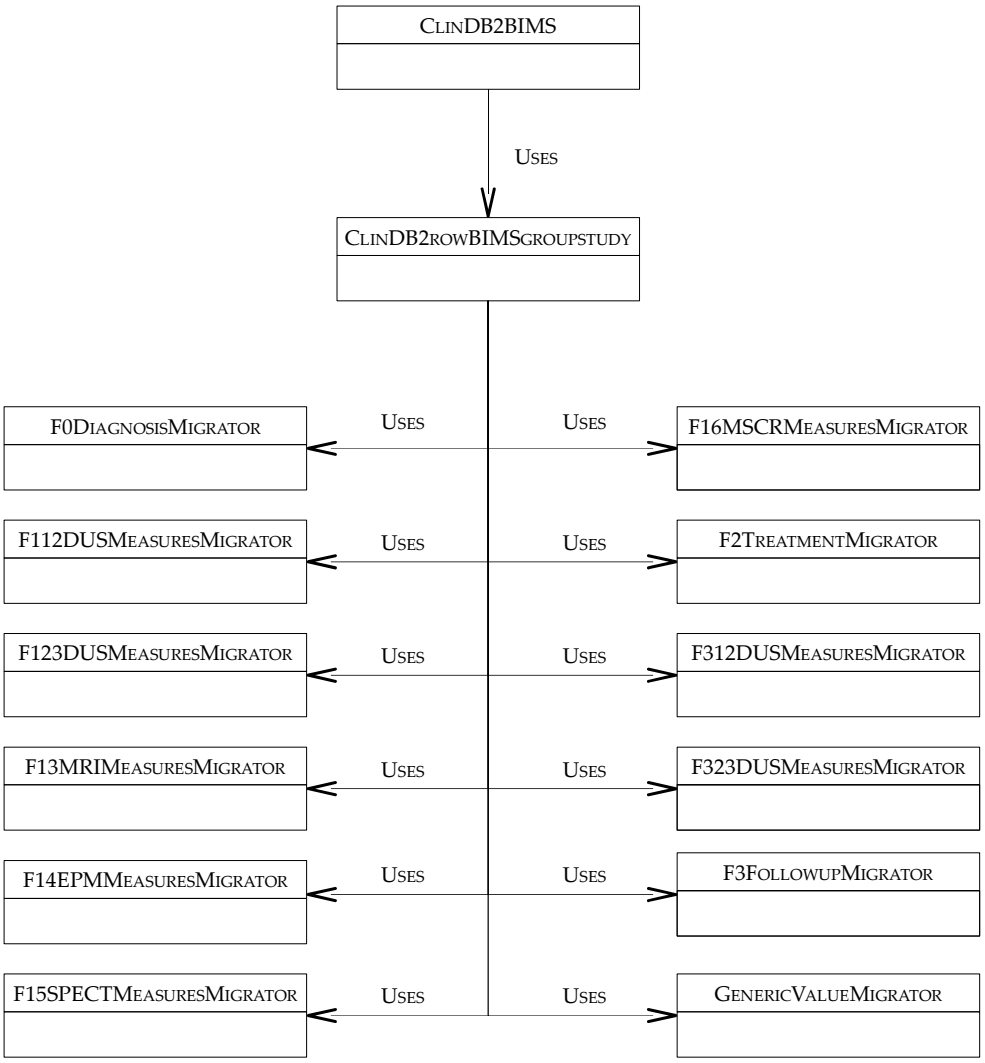


Figure 5.3: UML Class diagram of ClinDB2BIMS

Chapter 6

Conclusions

This project was motivated by the need to address data acquisition problems commonly encountered in biomedical research projects. The final result has been BIMS: Biomedical Information Management System.

BIMS is a software system which provides an infrastructure to gather, and manage all information required by biomedical research projects. BIMS can manage both, textual information, like clinical data, as well as images in DICOM format. This data management flexibility is the most important feature of BIMS, because it provides control on the information about a wide range of data types and structures. Therefore, BIMS can easily be adapted to store information of a extensive variety of clinical studies, not being limited to any given clinical specialty.

BIMS architecture is based on a two layer design, in order to define an strict separation between context domain functionalities (create a clinical study, create a user, etc ...), and accessibility functionalities (get access to BIMS through a web browser). Thereby, a self-countered software project is responsible for every layer of the BIMS architecture: the core layer and the access layer. On the one hand, BIMSCore is a software project that provides a library which meets all context domain functionalities. On the other hand, BIMSWebAccess is a software project which provides a web - based application in order to access BIMS using a web browser, with all security concerns.

The flexibility managing a wide range of data types and structures is achieved using an specific mapping process in the design of BIMSCore project. The BIMSCore mapping process is based on a well known approach named *two level modeling*, which consists on building a mapping scenario composed of two levels. The first level is composed of a set of highly abstract and simple objects, which can be combined in many different ways to construct more complex domain concepts. The second level is a set of specifications to define the properties and the combinations of the first level objects. Hence, the second level specifies a group of constraint rules which are applied to the first level objects at runt-time. Following this methodology, it is possible to define a wide range of data types, and structures in clinical studies managed by BIMS, without stopping the system to redesign, and re-adapt it to fulfill new requirements.

The use of flexible data acquisition systems to gather biomedical data, impacts positively in the information data quality of the research study, and improves its results. The data quality of a study must be managed carefully from the first stage of the work flow, where the information is entered into the study. To this end, several kind of strategies can be used to constraint the information submitted as a part of the study. BIMS provides a set of mechanisms to built a first level validation

of data submitted by the clinician. These mechanisms, based on enforcing the data submitted to match with a regular expression, or with a specific numerical range, ensure and raise the data quality of the clinical study. In addition, the medical research context is large, complex, and open-ended, and for these reasons, it is necessary to design and develop very flexible management systems. The mapping process of BIMSCore based on the *two level modeling* approach, provides an infrastructure adaptable to the specific characteristics of the biomedical research environments.

In conclusion, BIMS meets two important objectives in the management of biomedical research projects. On the one hand, BIMS provides flexibility to manage a wide range of data types and structures. On the other hand, BIMS provides a set of mechanisms to ensure a high level of data quality in the biomedical research studies.

Chapter 7

Future work

The results of this project provide a strong foundation for the implementation of further ideas related to BIMS. This chapter describes some of the most interesting lines of work which could be pursued as a continuation of this project.

7.1 BIMS Query & Export Module

The information managed by BIMS is commonly used by biomedical researchers, who test experimental algorithms over medical images. Thus, a typical task of a researcher, who needs access to BIMS, is to log into the system, search desired information (textual, and images), and use them in his experiments. The BIMS system provides a web browser navigation, as a simple way to search information. For this reason, it would be necessary to develop a new module to make specific queries to BIMS, and extract all information. For example, a query which a researcher could submit is to extract information (textual, and images) about smoker patients, with systolic blood pressure between 12 and 15, into a local directory. The BIMS Query & Export Module would be responsible for executing the query and pack all generated information in a folder defined by the researcher.

7.2 DICOM Object upload

The upload mechanism through a web - based application (HTTPS) is very critical, and it is not recommended for files of 40 Mb or more. It would be necessary to redevelop this transfer mechanism to upload massive files to BIMS. A potential alternative would be to use Secure File Transfer Protocol (SFTP), which provides a safe system to upload files, with all the necessary security mechanisms.

7.3 MedicalData relationships

The mapping process performed in BIMSCore Model package provides a very flexible mechanism to represent a wide range of data types to compose study CRF. This flexible representation lacks

a mechanism to define relationships between different MedicalData objects of BIMSCore Model package, and due to this the BIMS is not able link different fields of a form. A typical example is the following couple of questions:

1. The patient is a smoker?
2. How many cigarettes smokes per day?

It is obvious that the second question depends on the answer of the first question. For this, it is necessary to establish a relationship which disables the second question if the answer of the first is false. In order to maintain the high flexibility of the BIMSCore Model package, it is necessary that the relationship mechanism would be adaptable to a wide range of scenarios.

7.4 Visualization of fields based on their optionality

The flexibility of BIMS allows for the specification of whether a field of a group study is compulsory, or whether it is optional. If a field is optional, the system will allow to leave it empty, otherwise it will show a message to indicate that the field must be filled in.

An interesting functionality to add would be to design a feature to view only compulsory fields. For example, if a clinician does not want to spent so much time in a group study, and he desires to work only with mandatory fields, the activation of this functionality will produce that the system only renders the compulsory fields of the group study.

7.5 Log of changes

The clinical studies managed by BIMS could be modified, to solve mistakes or errors detected. For example, lack spelling, or misunderstood medical concepts can be modified at runtime. In order to gather all modifications, it could be interesting to maintain a historical log, where changes were recorded and assigned to an specific user of BIMS.

7.6 Standarized Model

The BIMSCore Model package is composed by a set of classes to represent the model domain elements in a flexible way. In the BIMS design stage, a simple and proprietary architecture was produced, and even though it is a model adaptable to a wide range of possibilities, it is not based on standardized model. Currently, there are organizations like Clinical Data Interchange Standards Consortium (CDISC) and European Committee for Standardization (CEN), which define specifications to develop and support global, platform-independent data standards that enable information system interoperability to improve medical research and related areas of health care [5, 12]. In a long term project related with the objective to create and manage clinical studies, it is necessary to adapt the model package to standard specifications.

Bibliography

[1] T. Beale. Archetypes: Constraint-based domain models for future-proof information systems. *Workshop Behavioral Semantics*, 2002.

[2] Kent Beck and Martin Fowler. *Planning Extreme Programming*. Addison-Wesley Longman Publishing Co., 2000.

[3] J. Bisbal and D. Berry. Archetype alignment: A two-level driven semantic matching approach to interoperability in the clinical domain. *HEALTHINF 2009*, pages 216–221, 2009.

[4] Peyton H. Bland, Gary E. Laderach, and Charles R. Meyer. A web-based interface for communication of data between the clinical and research environments without revealing identifying information. *Academic Radiology*, 2007.

[5] CDISC. Clinical data interchange standards consortium. http://www.cdisc.org/.

[6] Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley Professional, 2000.

[7] World Wide Web Consortium. W3c. http://www.w3c.org.

[8] DCM4CHE. Open source clinical image management. http://www.dcm4che.org.

[9] D. Fetzer and O. West. Protecting personal health information in research: Understanding the hipaa privacy rule. *Academic radiology*, 2007.

[10] CEN: European Committee for Standardization. EHR Communication Standard - ISO 13606. http://www.cen.eu.

[11] Apache Software Foundation. Apache myfaces. http://myfaces.apache.org/.

[12] S. Garde and E. Hovenga. Expressing clinical data sets with openehr archetypes: A solid basis for ubiquitous computing. *International Journal of Medical Informatics*, 2007.

[13] Jane Grimson, William Grimson, Damon Berry, Gaye Stephens, Eoghan Felton, Dipak Kalra, Pieter Toussaint, and Onno W. Weier. A CORBA-based integration of distributed electronic healthcare records using the Synapses approach. *IEEE Trans. Inf. Tech. in Biomedicine*, 1998.

[14] H. K. Huang. *PACS and Imaging Informatics: Basic Principles and Applications*. John Wiley, 2004.

[15] Luigi Lo Iacono. Multi-centric universal pseudonymisation for secondary use of the EHR. *Proceedings of the 2007 HealthGrid Conference*, 2007.

[16] Rod Johnson. *Expert One-on-One J2EE Design and Development*. Wrox, 2002.

[17] Sun M. JavaServer Faces Technology. http://java.sun.com/javaee/javaserverfaces/.

[18] Jan Machacek, Jessica Ditt, Aleksa Vukotic, and Anirvan Chakraborty. *Pro Spring 2.5*. 2008, Apress.

[19] Kito Mann. *JavaServer Faces in Action*. 2004, anning Publication Co.

[20] Catalina Martínez-Costa, Marcos Menárguez-Tortosa, Jesualdo Tomás Fernández-Breis, and José Alberto Maldonado. A model-driven approach for representing clinical archetypes for semantic web environments. *J. of Biomedical Informatics*, 42(1):150–164, 2009.

[21] Vicent Massol, Jason van Zyl, Brett Porter, John Casey, and Carlos Sanchez. *Better Builds with Maven*. Exist Global - Library Press, 2008.

[22] Dave Minter and Jeff Linwood. *Pro Hibernate 3*. 2005, Apress.

[23] NEMA. Digital imaging and communications in medicine. http://medical.nema.org/.

[24] D.L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15:1053–1058, 1972.

[25] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. Mc. Graw - Hill, 2006.

[26] Akaza Research. Openclinica. http://www.openclinica.org.

[27] W. W. Royce. Managing the development of large software systems: concepts and techniques. *Proceedings of the 9th international conference on Software Engineering*, 1987.

[28] Ian Sommerville. *Software Engineering*. Addison-Wesley, 2000.

[29] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. Conceptual modeling for ETL processes. *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, 2002.

[30] Zubin Wadia, Martin Marinschek, Hazem Saleh, and Dennis Byrne. *The Definitive Guide to Apache MyFaces and Facelets*. Apress, 2008.

[31] Craig Walls and Ryan Breidenbach. *Spring in Action, Second Edition*. Manning Publication Co., 2007.