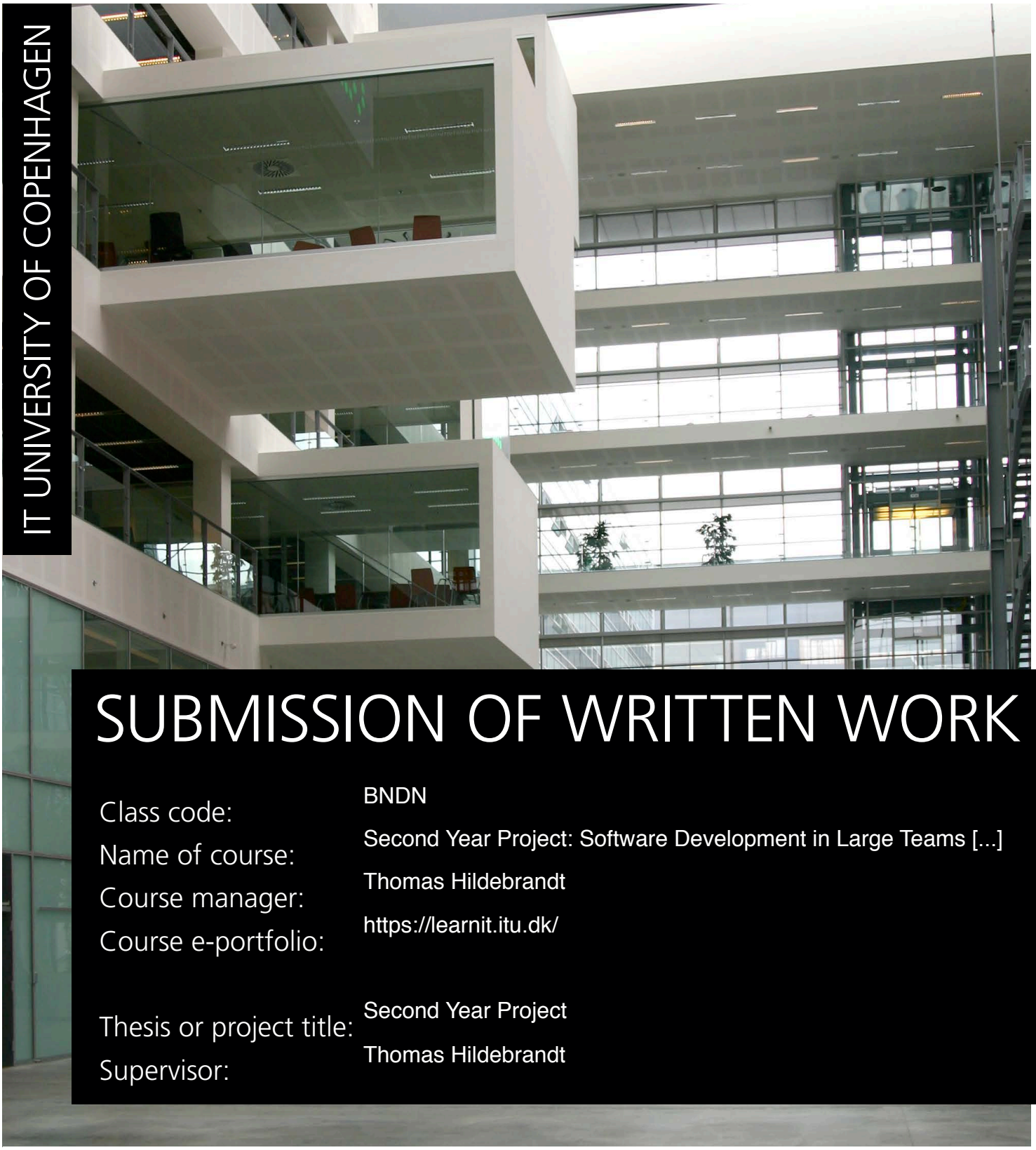
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code:	BNDN
Name of course:	Second Year Project: Software Development in Large Teams [...]
Course manager:	Thomas Hildebrandt
Course e-portfolio:	https://learnit.itu.dk/
Thesis or project title:	Second Year Project
Supervisor:	Thomas Hildebrandt

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Adam William Engsig	01/05-1993	adae@itu.dk
2.	Anders Fischer-Nielsen	06/05-1993	afin@itu.dk
3.	Anders Wind Steffensen	10/02-1993	awis@itu.dk
4.	Cecilie Strunge Jensen	28/09-1992	csje@itu.dk
5.	Mikael Lindemann Jepsen	17/02-1992	mlin@itu.dk
6.	Morten Albertsen	01/08-1988	moalb@itu.dk
7.			@itu.dk

Second Year Project:
Software Development in Large Teams with International Collaboration

IT-University of Copenhagen
Spring term 2015

Authors:

Adam William Engsig (adae)
Anders Wind Steffensen (awia)
Anders Fischer-Nielsen (afin)
Cecilie Strunge Jensen (csje)
Mikael Lindemann-Jepsen (mlin)

Morten Albertsen (moalb)

Contents

1 Introduction **3**
1.1 Conventions . . . 3

2 Requirements **3**
2.1 Scope . . . 3
2.2 Non scope . . . 4
2.3 Hospital Workflow . . . 4
2.4 Initial Plan of Division of Work . . . 5

3 Workflows, DCR graphs, and REST **5**
3.1 Workflows . . . 6
3.2 DCR graphs . . . 6
3.2.1 Events . . . 6
3.2.2 Relations . . . 8
3.3 REST . . . 8
3.3.1 HTTP Methods for CRUD Operations . . . 8
3.3.2 Be Stateless . . . 8
3.3.3 Resource Directory Structure . . . 8
3.3.4 Transfer XML or JSON . . . 8

4 Workflow elaboration **8**
4.1 Initial Workflow Design . . . 8
4.2 Feedback from External Partner . . . 8
4.3 Final Workflow . . . 8
4.4 Workflow Given by Lecturers . . . 8

5 Usermanual **8**
5.1 Compiling the Project . . . 8
5.2 Running the Client . . . 8
5.2.1 The Main Window . . . 8
5.2.2 History . . . 8
5.2.3 The Settings File . . . 8
5.3 The DCR Graph Parser . . . 8
5.4 Running Locally . . . 8

6 System Implementation **8**
6.1 Overview . . . 8
6.2 Overall Architecture . . . 8
6.2.1 Client . . . 8
6.2.2 Server . . . 8
6.2.3 EventAPI . . . 8
6.3 Interface-based Programming . . . 9
6.4 Dependency Injection . . . 9
6.5 Design of Client . . . 9
6.5.1 MVVM-principles in Client . . . 9
6.6 Design of Server and EventAPI . . . 9
6.6.1 Multi-layered Design . . . 9
6.6.2 Execution of an Event . . . 9
6.7 Distribution . . . 9
6.7.1 Location Transparency . . . 9
6.7.2 Global Identification of Events . . . 9
6.7.3 System Deployment . . . 9

6.8 RESTful APIs . . . 9
6.8.1 Restfulness of the System . . . 9
6.8.2 UnRESTful Parts of the System . . . 9
6.9 Minimal Logic Handling in Controllers . . . 9
6.9.1 Exception and Response Handling . . . 9
6.10 Persistence . . . 10
6.10.1 The Relational Database . . . 10
6.10.2 Data Distribution . . . 10
6.11 Role-Based Access Control . . . 10
6.11.1 Motivation . . . 10
6.11.2 Implemented Solution . . . 10
6.11.3 Discussion of the Implemented Solution . . . 10
6.12 Concurrency Control . . . 10
6.12.1 Motivation . . . 10
6.12.2 Implemented Solution . . . 10
6.12.3 Discussion of Implemented Solution . . . 10

7 Testing **10**
7.1 Unit Testing . . . 10
7.2 Integration Testing . . . 10
7.3 System Testing . . . 10
7.4 Acceptance Testing . . . 10
7.5 Discussion of Testing Approach . . . 10

8 Reflection on Project **10**
8.1 Thoughts on the Delivered Project . . . 10
8.2 Thoughts on Group Processes and Tools . . . 10

9 Conclusion **10**

A Healthcare Process From Brazil **11**

1 Introduction

The following report describes the system developed in the course "*Second Year Project: Software Development in Large Teams with International Collaboration*" taught by Thomas Hildebrandt and Rasmus Nielsen at the fourth semester of the Bachelor of Science in Software Development at the IT University of Copenhagen.

The report gives an introduction to Dynamic Condition Response Graphs (*DCR graphs*), an overview of the software architecture of the delivered system, how the system has been tested, and why the team believe it is functioning. Both the construction of the system and the report was completed in the Spring term 2015.

The purpose of the system is to support the functionality of DCR graphs, and to test the system on a workflow description of the processes in a Brazilian healthcare system. The processes are described by an external partner in Brazil, Eduardo A. P. Santos.

All DCR graphs presented throughout this report is generated through the online tool available at DCRGraphs.net.

1.1 Conventions

The following conventions is used throughout the report:

- `monospaced` text for code and classes
- "The system" refers to the actual product.
- "The solution" refers to the implemented solution to a problem.
- "The project" refers to the process of developing the system.
- If project or solution is prefixed with VS it refers to Visual Studio projects and solutions.
- Specific roles in workflows will be written with an *italic* font.
- Specific events in workflows will be written with a **bold** font

2 Requirements

This section states the given requirements to the system and how they were interpreted.

From the project description:

> "*The goal is to develop a functioning and correct web-service-based distributed workflow system that can support distributed coordination of workflows provided by the external and possibly international customers, and reconfigured if the workflow changes.*"

2.1 Scope

The system should fulfill the following requirements:

- The system must be able to handle generic DCR graphs and the logic they follow. This is to be tested on a DCR graph produced from a workflow of a Brazilian hospital and a DCR graph given by our lecturers.
- The system must be able to create, reconfigure, and delete workflows.

- The architecture of the system must resemble a peer-to-peer distribution.
- The implementation should be based on REST services.
- The system should provide a graphical user interface (UI) to the user.
- The UI should be able to present an event log or history, that describes which operations have been executed, aborted or changed in a given workflow. Because "log" and "lock" sounds the same the team chose to call the "logging" feature "History".
- The system must persist data such that it can be restarted and handle isolated occurrences of crashes.
- The system must implement concurrency control that ensures that an illegal state cannot be reached.
- The system must implement role based access control (RBAC) that ensures that a user is only able to make changes that the given user has permission to make.
- The system must allow reconfiguration of already existing events and their relations. This can be achieved by deleting and then recreating the individual events.

2.2 Non scope

The following was intentionally left out during the design and implementation of the system:

- The team realized that some workflows might require storing of data such as a patient record. The team has chosen not to support this feature.
- Multiple instances of the same workflow. The system only supports a single instance of each workflow, as the team did not see this as being the cornerstone of the given assignment - however the system supports creation of identical workflows.
- The system does not provide a UI for creation of DCR graphs. The team recommends the use of `http://www.DCRGraphs.net`
- Furthermore it is not in the scope of the project to support nested events on a workflow as is possible in the full specification of DCR graphs.
- Being able to handle crashes of services is not part of the scope of this project, no handling for this has been implemented.

2.3 Hospital Workflow

This section introduces the workflow, that was used as a sample workflow throughout the project. The workflow was based on a textual description, provided by our external partner. The workflow concerns medical care and treatment at a hospital and is seen from the point of view of the hospital, i.e. all assignments in the workflow are handled by the employees of the hospital.

The hospital workflow is designed in multiple iterations: the initial draft and the final workflow. The initial draft was the interpretation of the textual description. The initial draft was then reviewed by the external partner. The feedback was taken into consideration in the development of the final workflow.
The team believes that the final workflow supports the tasks the hospital needs.

Figure 1: reflection

2.4 Initial Plan of Division of Work

At the beginning of the project the group had an introductory meeting, discussing the members' expectations of the project, and outlining a group constitution.
To cope with different expectations of the members, everyone had a chance to express what part of the project they wanted to contribute to the most. The team had an agreement that no member should be assigned to work on a single part of the system. The decision was made to make sure that everyone felt ownership of the entire project and thereby making everyone able to test and work on any part of the project when necessary.

Having everyone involved equally in the project was declared as a goal for the team, however it was decided not to have everyone contribute equally in team management and implementation of the system. Four members would mainly be in charge of implementation and the remaining two would mainly handle team management. To make sure that everyone were focused on the entire project, a minimum of 30
The implementation of the system was focused on first documenting the proposed system architecture. Subsequently the assigned team members decided on the implementation details, and the entire team discussed the possible solutions in unison.

The team wanted tests to be a high priority during development.

To ensure that the report would reflect the project as well as securing against adding new bugs with new implementation, both a feature freeze and a code freeze were planned. These were planned two and one weeks respectively prior to handin.

3 Workflows, DCR graphs, and REST

This section will give readers a theoretical introduction to workflows, DCR graphs, and REST-based web services. These concepts will become relevant as they are used in the delivered solution and will be discussed later in this report.

Figure 2: Link to table

3.1 Workflows

A workflow is a standardized work description of a process that is executed repeatedly.
An instance of a workflow describes an ongoing process with several stages and describes in which order these states must be reached. A workflow contains several tasks that must be executed for the workflow to either finish in a successful or unsuccessful state. The workflow must also guarantee that the process finishes.
Certain types of workflow system notations exist. Today, most workflow systems are based on the flow-oriented process notation[1]. These, unlike those described with DCR graphs, do not support that the design of the workflow can change dynamically. Furthermore the user cannot deviate from the plan even if it made sense to do so business wise. One of the strengths of DCR graph-based workflows is that any route can be taken to complete the workflow as long as the rules of the workflow are followed.

3.2 DCR graphs

This section will give a brief introduction to DCR graphs. DCR graphs present an alternative notation to the notation of standard flow-oriented processes.
A DCR graph represents a workflow. A workflow could, for instance, be the yearly reporting of gas consumption in a household, as seen in Figure 3. A DCR graph is made up of a number of the following two components: *events* and *relations*. The term DCR *graph* is due to events representing nodes, and relations representing edges between nodes.

3.2.1 Events

Events represent activities within a workflow. In the example, Figure 3, there are five events, among others **Read Gasmeter** and **Bill Customer**. An event may have a role associated with it e.g. the event **Read Gasmeter** has a *Customer*, and the event **Bill Customer** an *Inspector*. The role determines who are allowed to execute the event.

[1] Concurrency and Asynchrony, *Søren Debois, Thomas Hildebrandt, and Tijs Slaats*, IT University of Copenhagen

Figure 3: Sample Pay Yearly Gas Bill

	Options	Default value
Pending	false | true	false
Included	false | true	true
Executed	false | true	false

A state for each event is needed for the functionality of a DCR graph to be implemented. The state of an event is comprised of the following information, see Table XYZ:

3.2.2 Relations

3.3 REST

3.3.1 HTTP Methods for CRUD Operations

3.3.2 Be Stateless

3.3.3 Resource Directory Structure

3.3.4 Transfer XML or JSON

4 Workflow elaboration

4.1 Initial Workflow Design

4.2 Feedback from External Partner

4.3 Final Workflow

4.4 Workflow Given by Lecturers

5 Usermanual

5.1 Compiling the Project

5.2 Running the Client

5.2.1 The Main Window

5.2.1.1 State of Events

5.2.1.2 The Status Bar

5.2.1.3 Reset Workflow

5.2.2 History

5.2.3 The Settings File

5.3 The DCR Graph Parser

5.4 Running Locally

6 System Implementation

6.1 Overview

6.2 Overall Architecture

6.2.1 Client

6.2.2 Server

6.2.3 EventAPI

d

6.3 Interface-based Programming

6.4 Dependency Injection

6.5 Design of Client

6.5.1 MVVM-principles in Client

6.6 Design of Server and EventAPI

6.6.1 Multi-layered Design

6.6.2 Execution of an Event

6.7 Distribution

6.7.1 Location Transparency

6.7.2 Global Identification of Events

6.7.3 System Deployment

6.8 RESTful APIs

6.8.1 Restfulness of the System

6.8.2 UnRESTful Parts of the System

6.9 Minimal Logic Handling in Controllers

6.9.1 Exception and Response Handling

6.9.1.1 Exception is Handled Immediately

6.9.1.2 Exception is Propagated Upwards

6.10 Persistence
6.10.1 The Relational Database
6.10.2 Data Distribution
6.11 Role-Based Access Control
6.11.1 Motivation
6.11.2 Implemented Solution
6.11.3 Discussion of the Implemented Solution
6.12 Concurrency Control
6.12.1 Motivation
6.12.2 Implemented Solution
6.12.3 Discussion of Implemented Solution
7 Testing
7.1 Unit Testing
7.2 Integration Testing
7.3 System Testing
7.4 Acceptance Testing
7.5 Discussion of Testing Approach
8 Reflection on Project
8.1 Thoughts on the Delivered Project
8.2 Thoughts on Group Processes and Tools
9 Conclusion

A Healthcare Process From Brazil

Project: Build the AS-IS model of the medical care of a hospital located in Curitiba, Brazil. This private hospital provides services (appointments, exams, surgeries) to the population. Brazilian government pays every provided service by the hospital.

General description:

1) Patient goes first to a medical care unit called UBS∗ (Healthcare Basic Unit). Each region in Curitiba has at least an UBS. The objective of an UBS is to provide the first care to the population. In UBS, a non-specialist physician initially treats the patient. The physician may recommend the patient an appointment with a specialist or request some exams. In both cases, UBS is responsible to schedule appointments and exams. Thus, at the end of the first appointment, UBS provides where and when the patient must go (to an appointment with a specialist or to do exams). Our interest here relies on appointments and exams scheduled in a specific hospital;

∗ UBS is a unit managed by the government of Parana State (Curitiba is the capital of Parana). So, its process is out of the scope of the present project.

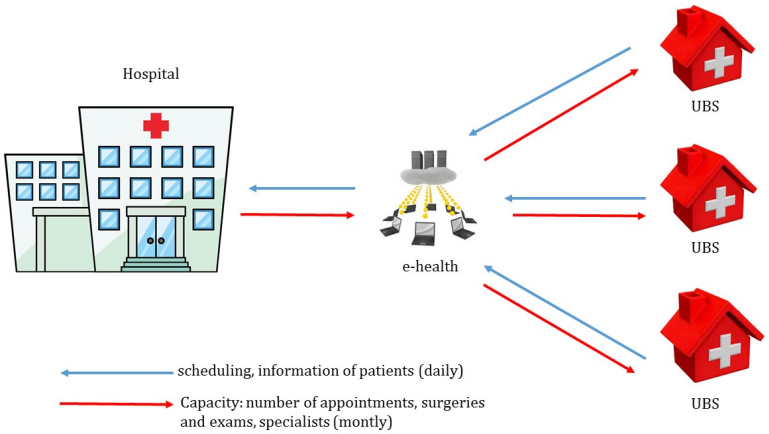
2) The hospital inform monthly to all UBS the amount of appointments and exams available. Thus, each UBS schedules an appointment or exams according to such availability;

3) There is an information system (called e-health) that manages the scheduling of patients. The e-health supports the healthcare public system of Paraná State. Private hospitals cannot modify or make any changes in e-health. The only thing that they can do (using e-health) is to inform the availability of appointments and exams (as described in 2);

4) In a daily basis, the hospital received (from e-health) the scheduling of patients. Thus, the first task that employers have to do is to download the file that contains a list of patients for that day (appointments and exams). This list includes private information of each patient, the time of appointment or exams, and the specialist for that appointment;

5) When the patient arrives in some of these clinics, the first reception has to check his/her scheduled time (according to information from e-health) and register some information about him/her. Then the patient is referred to a second reception where he/she has to provide an attendance record issued by the first reception. In such record is informed which specialist the patient should go or which exams are requested. The second

reception is responsible to organize the service queue of both appointments and exams;



6) In the case of an appointment, the specialist calls each patient according to the queue organized by the second reception. The medical examination is executed and the specialist creates a medical report. The specialist may request some exams, recommend an appointment with another specialist, or both, or may recommend some medications and ask the patient to return, or may recommend some surgery;

7) In the case of exams, the patient is first prepared. Then the exam is performed. A medical examination report must be created;

8) Depending of the specialist recommendation, after appointment or exams the patient has to go to the first reception in order to schedule new appointment or exams or surgery or return. The first reception has to provide information about date, time, addresses of appointments and exams. This step is called 'checkout';

9) It is possible that a specialist recommend for a specific patient a maximal priority over other patients concerning exams, appointments (with other specialist) or surgery. In this case, the patient is referred to other sector called ROTA. In this sector an auditor (a specialist working for Brazilian Government) will confirm the urgency of each case;

10)The hospital will charge the Brazilian government the services provide to all patients. Thus, is necessary that the all reports store the whole set of relevant information.

What the operations managers of the hospital want:

i) To build a pervasive healthcare process model for the hospital;

ii) To represent the exception alternatives in the healthcare process (cancellations and re-scheduling of appointments and exams);

iii) To reduce the total time of patients in clinic. A certain level of automation has been suggested to reach this objective. For example, the managers want to give permissions to the specialists to schedule another appointment or exams. Currently this activity is under responsibility of first reception;

iv) The current process does not consider the emergency cases. It is necessary to create a process to treat this situation;

v) To restrain the permissions of the specialists. The aim is each specialist can only schedule exams related to your specialty;

vi) To propose performance indicators to monitor the quality of healthcare process.