FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Encounter Management

João Pedro Martins dos Santos de Carvalho Aradas

Report of Project
Master in Informatics and Computing Engineering

Supervisor: Eurico Manuel Elias Morais Carrapatoso (Professor Auxiliar)

1st July, 2009

Encounter Management

João Pedro Martins dos Santos de Carvalho Aradas

Report of Project
Master in Informatics and Computing Engineering

16st July, 2008

Abstract

Historically the hospitals have made many efforts to make the correct management of their patients and the services provided to them. This management was not always an easy task due to a variety of factors. The most important is that every decision in a healthcare facility ultimately deals with human lives. Other factors include the large amount of data to be processed, organized and properly stored. These problems are worsened by the fact that the patient's data is considered sensible information. The technology streamlined these processes facilitating this management.

The Encounter Management application is divided in two components. The component of infrastructure (backend) and another for interaction with the user (frontend) in which the user can perform activities for administratively admit, discharge transfer and also the management of encounter/visits between the patients and the healthcare facilities with the use of web forms. The backend is a database for storage of local information, HL7 communication with external entities, and other interactions with external entities, such as sources of identification and authentication. The frontend should provide an interaction with the user, rich content, providing useful information, being able to cover the requirements for the management of a encounter of a patient, both at the arrival of the patient, as at the time of his discharge.

One objective of this project was to develop this software using the most advanced management concepts of encounters and also the most advanced technologies in the market. Another objective was to develop this software in a modular, dynamic and adaptable way to all streams of work related to hospital management encounters.

To achieve the objectives, a combination of agile methodologies and a rigid process of documentation was used.

The results obtained by applying this methodology resulted in a systematized modelling of the concept of encounters and all the medical terms associated (episodes of care, episodes and encounters). This modelling led to the development of the Encounter Management software for outpatient visits. This was properly modularized in frontend and backend.

Subsequently, it was concluded that to develop new modules, would only be necessary to add more workflows and the according user interfaces to the backend and frontend. The choice of a workflow engine to the software development has proved to be a right choice because it allows all new hospital workflows to be modelled correctly.

Resumo

Desde sempre os hospitais se preocuparam em fazer uma correcta gestão dos seus pacientes e dos serviços prestados a estes. Esta gestão nem sempre foi uma tarefa fácil devido a grande quantidade factores. O factor mais é importante deve-se ao facto de toda e qualquer decisão numa unidade de saúde pode potencialmente afectar vidas humanas. Outros factores incluem a grande quantidade de dados a serem tratados, devidamente organizados e guardados. Estes problemas são agravados pelo facto de todos os dados dos pacientes serem sensíveis. A tecnologia agilizou estes processos facilitando esta gestão.

A aplicação de gestão de visitas será constituída por uma componente de infra-estrutura (*backend*) e outra de interacção com o utilizador (*frontend*), na qual será possível realizar actividades de admissão, transferência e alta bem como uma gestão integrada de encontros entre pacientes e as unidades de saúde com o recurso a formulários web. O *backend* é constituído por uma base de dados para armazenamento da informação local, serviços de comunicação HL7 com entidades externas, e outras interacções com entidades externas, como sendo fontes de identificação e autenticação. O frontend deverá disponibilizar uma interacção com o utilizador, rica de conteúdos, disponibilizando informação útil, capaz de cobrir os requisitos de gestão de uma visita de um paciente, tanto no momento da chegada deste, como no momento da sua alta.

Um dos objectivos deste projecto foi desenvolver este software utilizando os mais avançados conceitos de gestão de visitas tal como as mais avançadas tecnologias existentes no mercado. Outro objectivo foi desenvolver o presente software de uma forma modular, dinâmica e adaptável a todos os fluxos de trabalho hospitalares relacionados com a gestão de visitas.

Para atingir os objectivos foi utilizada uma combinação entre metodologias ágeis e um processo de documentação rígido.

Os resultados obtidos através da aplicação desta metodologia culminaram numa correcta modelação do conceito de gestão de visitas e todos os termos médicos associados (episódios de cuidados, episódios e encontros). Esta correta modelação levou ao desenvolvimento do software de gestão de visitas do módulo de consultas externas. Esta foi correctamente modularizada em *frontend* e *backend*.

Posteriormente, concluiu-se que para realizar o módulo de medicina interna e outros relacionados com fluxos de trabalho hospitalares, apenas seria necessário adicionar mais fluxos de dados ao *backend* e respectivas interfaces ao *frontend*. A escolha de um motor de fluxos de dados para o desenvolvimento do software revelou-se acertada porque rapidamente qualquer nova situação hospitalar poderá ser modelada correctamente.

In loving memory of my Grandparents
Horácio Mariano And José Aradas

Acknowledgements

First and foremost i would like to thank everyone at Siemens that make this internship possible. Special thanks goes to António Martins and Daniel Ramos for guiding me continuously throughout the project. I would also like to thank the other internees at Siemens, Tiago Caçador for the endless discussions on Silverlight and WPF, Nuno Silva for the help in the implementation and Pedro Gomes for the help in all HL7 issues. Also from Siemens I would like to thank Joaquim Matos and Ricardo Gomes for all the help and comments on my work.

From FEUP I would like to thank the professor Eurico Carrapatoso for his guidance and advices that helped throughout the project.

I would also like to thank all my friends for all the good moments, the support and their great laughter.

The most important, I would like to thank my Parents Daniel Aradas and Delfina Aradas and the best sister in the world Carolina Aradas for their never ending love and support.

Last, but not least, I would like to thank Patrícia for the great moments.

João Aradas

Contents

List of Figures

List of Tables

Listings

Glossary

This project was conducted in the area of Healthcare. Because of that, some of the terms that are going to be introduced in the present thesis aren't of general knowledge. These terms that are presented in the following table are used throughout the document. For the full understanding of the project and this thesis it is necessary to acknowledge a few terms. In the following table it's described a term in English, the correspondent translation to Portuguese and finally a brief description of the term.

English	**Portuguese**	**Description**
Administrative discharge	Alta administrativa	The hospital have to ensure that the patient is ready to go home, maybe he still need care, not from the hospital but from family, friends, or any one else. Social assistance will be involved if necessary.
Administrative encounter	Encontro Administrativo	The personal contact between the patient and a health-care professional in which no patient care is rendered, e.g., scheduling an appointment.
Admission	Admissão	The process by which a patient becomes an inpatient in a healthcare facility. It consists of bed assignment, resource allocation for an inpatient stay, processing paperwork (clinical, administrative, and financial), and performing tasks required prior to the patient's arrival and/or execution of the inpatient services. Includes clinical assessment of the patient and initiation of the plan of care.
Ambulatory	Ambulatório	Ambulatory care services normally include: visit, encounter, consultation, treatment, and intervention using advanced technology, equipment and procedures. The patient's stay at the facility, from the time of registration to discharge, occurs on the same calendar day.
Ambulatory surgery	Cirurgia de ambulatório	This is a type of surgery that can be done to patient as outpatients, because it does not require that the patient stays in the hospital for more than 24 hours
Appointment	Consulta	Commitment of accomplishment a heath care service that has been previous scheduled.
Clinical discharge	Alta clínica	When a doctor says that patient is medical treated to go home.

Day-hospital	Hospital de dia	Medical care including diagnosis, observation, treatment and rehabilitation that is provided on an outpatient basis. Ambulatory care is given to persons who are able to ambulate or walk about. A well-baby visit is considered ambulatory care even though the baby may not yet be walking.
Encounter	Encontro	An encounter is the atomic activity that happens inside a healthcare facility.
Episode	Episódio	An episode occurs over time, consists of one or more encounters, and covers health services to address one or more medical problems. It ends when no appointments are made to address the medical problem. For example, chronic illness can give reason for uninterrupted care - a lifelong episode. A series of events which is distinct in itself. For example, a period of fever, which disappears, may be an episode of fever within a continuous process, such as a chronic illness.
Episode of care	Episódio clínico	A continuous course of care administered to a patient by an healthcare provider for a specific medical problem, condition, or reason. From an application's point of view, an episode of care encompasses one or more related Encounters.
Hospital discharge	Alta hospitalar	Similar to Administrative discharge.
Inpatient	Internamento	A patient whose care requires a stay in a hospital. As opposed to an *outpatient*. An inpatient on the other hand is ”admitted” to the hospital and stays overnight or for an indeterminate time, usually several days or weeks.
Lab	Análise	Study or determine the nature and relationship of the parts of by analysis; usually : to examine by chemical analysis.
Medical Appointment	Consulta médica	This defines the act of assistance performed by a doctor to an individual. It can consist in clinical observation, diagnostic, therapeutic prescription, counseling or checking the state of evolution of the patient's health state.
Operation	Operação	A procedure performed on a living body usually with instruments for the repair of damage or the restoration of health and especially one that involves incision, excision, or suturing.
Outpatient	Consulta externa	Is a patient who is not hospitalized overnight but who visits a hospital, clinic, or associated facility for diagnosis or treatment. Treatment provided in this fashion is called ambulatory care.
Outpatient appointment	Consulta externa	Term related to a Schedule visit to a doctor in the hospital.
Over schedule	Extra agenda	An appointment that is out of schedule.

Patient	Doente	A person who requires medical care.
Person	Pessoa	Each kind of person, like visitor, health professional, physician, desk clerk, patient.
Primary care	Cuidados primários	Basic or general health care focused on the point at which a patient ideally first seeks assistance from the medical care system.
Primary Health Care Center	Centro de saúde	Basic level of health care that includes programs directed at the promotion of health, early diagnosis of disease or disability, and prevention of disease.
Referral	Referenciação	The concept that defines, if a person will need Primary care or Secondary care.
Small Surgery	Pequena cirurgia	Surgical operation with a value of K inferior to 50.
Sonho	Sonho	System used in most of the Portuguese public hospitals. This system is used for doing admissions, some clinical support and billing.
Subsequence appointment	Consulta subsequente	Next appointment for the same reason of a previous visit to a doctor in the hospital.
Surgery	Intervenção cirúrgica	One or more surgical operations with the same therapeutically objective and/or diagnostics, performed by one or more surgeon in one operating room within the same session, under anesthetics with or without an anesthetist.
Tranfer a patient	Tranferir paciente	Change patient location. For example: Urgency to inpatient.
Urgent appointment	Consulta urgente	A kind of appointment that only physicians can schedule because of it's emergency.

Abbreviations

ACSS	Administração Central do Sistema de Saúde
ADT	Admit, Discharge and Transfer
API	Application Programming Interface
ASP	Active Server Pages
ASP.NET	Microsoft platform for development of web applications
CT	Computerized Tomography
EM	Encounter Management
GPA	Global Patient Access
HIS	Hospital Information Systems
IDE	Integrated Development Environment
CMDT	In Portuguese *Meios Complementares de Diagnóstico e Terapêutica*, the English direct translation is Complementary Means of Diagnostic and Therapeutic
MPI	Master Patient Index
MRD	Medical Responsible Department
MRI	Magnetic Resonance Imaging
ORM	Object-Relational Mapping
RFID	Radio-frequency identification
RIA	Rich Internet Applications
RUP	Rational Unified Process
SOA	Service Oriented Architecture
UI	User Interface
US	Ultrasound
VB	Visual Basic
WCF	Windows Communication Foundation
WF	Windows Workflow Foundation
WPF	Windows Presentation Foundation
WWW	*World Wide Web*
XAML	eXtensible Application Markup Language

Chapter 1

Introduction

This chapter briefly contextualizes this internship project, presenting the company where it was developed, the context of the project, a small explanation of the project, the motivations and objectives and the thesis structure from this point on.

1.1 About Siemens

With 500 production centers in 50 countries and representation in 190 countries, Siemens is spread all over the world. In Portugal, Siemens S.A. encloses two factories, software research & development centers (Lisbon and Porto) and has a significant representation all over the country through its partners and company headquarters. Since 2008, the company is organized in three major sectors: Industry, Energy and Healthcare.

The **Industry Sector** and its solutions address Industry customers regarding production, transportation and building systems. This Sector is organized in five divisions: Industry Automation and Drive Technologies, Building Technologies, Industry Solutions, Mobility and OSRAM.

The **Energy Sector** offers products and solutions for generation, transmission and distribution of electrical energy. This Sector is organized in six divisions: Fossil Power Generation, Renewable Energy, Oil & Gas, Energy Service, Power Transmission and Power Distribution.

The **Healthcare Sector** stands for innovative products and complete solutions, as well as service and consulting in healthcare industry. This Sector is organized in three divisions: Imaging & IT, Workflow & Solutions and Diagnostics.

The Imaging & IT Division provides imaging systems for early diagnosis and intervention, as well as for a more effective prevention. These systems are networked with high-performance healthcare IT to optimize processes (such as hospital data systems like **Soarian**®, image processing systems like **Syngo**®, and knowledge-based technologies for diagnoses support).

The Workflow & Solutions Division provides complete solutions for fields such as cardiology and oncology and neurology. This Division offers solutions for, e.g. women's health (mammography), urology, surgery and audiology. It also provides turnkey solutions (including national health IT systems, complete solutions for health care providers), and consulting. In addition, Workflow & Solutions is responsible for the Sector's service business and for managing customer relations. The Diagnostics Division covers business with in-vitro diagnostics, including immune diagnostics and molecular analysis. The Division's solutions range from point-of-care applications to automation of large laboratories.

First fully-integrated diagnosis company Siemens IT Solutions and Services, leader in Information Technologies services, works as a transverse business unit.

In Portugal, Siemens SA Healthcare Sector is a market leader in the health care area, known for its competence and innovation skills in diagnostic and therapy systems, as well as information technologies and systems' integration.

Recent Milestones in Portugal

- Breast Pathology Service in Hospital de São João in Porto, Hospital da Luz in Lisbon and Clínica Dr. João Carlos Costa in Viana do Castelo – first total patient focus units, including all necessary technologies for the complete clinical process;
- Hospital da Luz in Lisbon – first hospital in Portugal with SOARIAN®clinical information system, becoming one of the most modern health care installations in Europe;
- Clínica Quadrantes, in Lisbon – in-vitro diagnostics and information technology systems, which together with a PET/CT system, complemented the existing Siemens in-vivo diagnostic systems at the clinic;
- Universidade de Coimbra – 3 Tesla Magnetic Resonance Imaging System exclusively for neuroscience research. This unit is part of the Brain Imaging Network Grid, a scientific cooperation network which integrates the Universities of Coimbra, Aveiro, Porto and Minho.

1.2 Context

The application of technology in healthcare has a long history. These two separate areas have become evolved in many ways. In one hand the monitoring equipment and the diagnostics equipments (CT or MRI Scanners) have evolved with the latest achievements in technology research and on the other hand the software solutions that supports all the activities within a healthcare facility. This healthcare related software has many roles, it helps provide electronic patient data information to the Hospital Information Systems (HIS). Today the operations inside a healthcare facility are all supported by some kind of software.

The healthcare facilities provide a variety of services to the patient. These services range from diagnostics, consults, exams, surgery and many others. All these services need a variety of support operations to happen. The software takes a relevant part in the monitorization of all these activities.

When these procedures start, the beginning of the process is generally preceded by an admission. Within the procedure, transfers can occur and in the end, within certain rules the patient is discharged thus completing the cycle. These basic administrative healthcare procedures are called Admit, Discharge and Transfer (ADT)

1.3 Project

An Encounter Management system allows to administratively process visits, admissions and encounters of a patient in a healthcare facility. Typically there are admissions in the emergencies, inpatient transfers or appointments all with the required discharges. During that period there are healthcare services that are provided to the patient like appointments, diagnostics or treatments. The objective of this application is to record and track these encounters. One component of the applications' infrastructure implies sending special messages that reflect the events that are recorded by the application, so that other applications can be informed of those records. To achieve that objective, HL7 messages of the ADT[1] type (among others that are sent to other HIS applications) of the international standard Health Level 7 (HL7) are used.

1.4 Motivation and Objectives

The evolution in the informatization of the healthcare facilities is paving the road for new approaches for the patient treatments and handling. The possibility to create an application that can unify the entire medical responsible departments (MRD) administrative task is motivating.

The program to be created has to be designed in a modular way so that a medical analysts can create his own workflows that support the various activities of the healthcare facility. This workflow would be created by selecting the desired features, setting the order between them and the transition conditions. All modules would work together to provide a sandbox in which the analyst can create the necessary workflows. A workflow is a set of functionalities that are ordered to achieve a certain objective.

[1]Type of messages in the HL7 standard that reflects the Admit, Discharge and Transfer operations inside a healthcare facility

The objectives of the project are:

- Project web page with authentication system;
- Validate, research and document the state of art on technologies and medical related standards regarding the type of the problems proposed and existing solutions for the problem or specific functionalities;
- Define an architecture that supports the *Encounter* concept;
- Produce a prototype that supports the workflow philosophy;
- Design and implement the Encounter Management;
- Use state of art technology when possible.

To pursue these objectives the Scrum Methodology is going to be used. In today's software development requirements often change during the product development life cycle to meet shifting business demands, creating problems[1]. Scrum software development process addresses these concerns[2]. To assist in the documentation process the Rational Unified Process (RUP) is going to be used. This approach provides the best balance between the working periods in Scrum and the well defined documentation that is necessary in RUP [3].

1.5 Thesis Structure

The thesis apart from the introduction contains 5 more chapters. In Chapter 2, it is described the state-of-art and some existing applications. It is also detailed the used frameworks and why they were chosen. In Chapter 3, it is presented a description of the problem by giving a deeper understanding of what this project aims to accomplish. Chapter 4 presents a high level solution of the problem that was described in the previous chapter. It starts with the clear definition of the used medical terms that were defined and adapted according to the needs of this project in particular and then shows and explains the framework that supports the system. In Chapter 5 the design of the applications is presented and the implementation details of each module is described. In Chapter 6 reviews the project, draws the necessary conclusions, the achievements and some future work to further improve the project.

Chapter 2

State of Art

This chapter starts by introducing the healthcare standards. Then it shows the existing technologies on the market that can help solve the problem that was briefly described in the last chapter. It also shows some guidelines that the healthcare industry is currently using in the various perspectives (Globally, in United States of America and in Portugal. Finally a few solutions to the Encounter Management issues are described.

2.1 International Healthcare Standards

This project is developed in the Healthcare area. This area requires the understanding of some existent International Healthcare Standards that the program must respect in order to be fully integrable with the existent platforms. In the next sections it will be described the standards that apply to the Encounter Management in particular. Such standards set the language, structure and data types required for seamless integration from one medical information system to another[4].

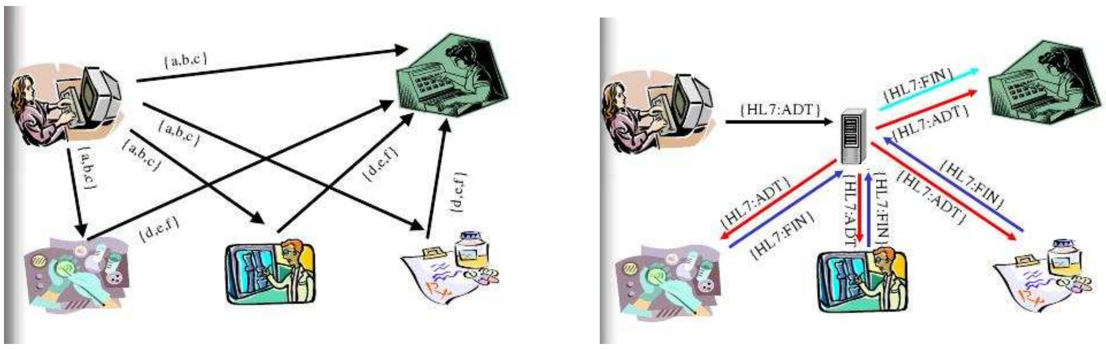
2.1.1 HL7

HL7[5] is a voluntary non-profit organization that is dedicated to the development of standards in the health industry. The name Health Level Seven(HL7) is a reference to the seventh layer of the OSI model. The name indicates that the HL7 clearly focus on the protocols of the application layer that is independent of the lower layers. The final purpose is the creation of coherent standards that allow the interoperability between all the medical information systems, allowing the experts in the healthcare area to participate in its development.

The HL7 develops conceptual specifications (*e.g.* HL7 RIM), documentation specification (*e.g.* HL7 CDA), application specification (*e.g.* HL7 CCOW) and messaging specification (*e.g.* HL7 v2.x and V3.0). This last specification is of extreme importance because it is what defines how the information should be grouped and communicated between all the health related applications that are used in healthcare facility activities, thus allowing a much greater interoperability between medical information systems[5].

2.1.1.1 V2.X

This version defines a series of electronic messages used to support not only clinical process but also administrative, logistics and financial. Since 1987 this specification was developed and was regularly updated, reaching from version 2.1 to version 2.6. The existence of this new protocol allowed application to communicate using only one language, as it can be seen in figure 2.1.



(a) Medical Information Systems Without Hl7 (b) Medical Information Systems With HL7

Figure 2.1: Before and After the HL7

Knowing which version is being used by an application is not crucial because all version 2.X are generally compatible with each other. The philosophy of this specification says that all new version must be compatible with the existing ones. Every time new data and message types are added, they are marked as optional elements, that means that recent version can process a message of an application that uses an older version and vice-versa.

Although it exist a newer 3.0 version in the market, the version 2.X continues to be the most used and is still under development. Despite all the efforts, this standard still has a few problems:

- The lack of a consistent data model that implies that the storage of data done by an clinical application has a direct impact over the pieces of the message that can be implemented;
- Lack of formal methodologies to model the data elements and messages, this can cause inconsistencies within the specification and create problems in the way that each element of the message relates with the other elements;

- The good specification flexibility leads to a vague and less precise specification. There are no defined user roles for its usage, so the implementation of this standard can vary across applications.

Nonetheless, despite this disadvantages, this version was used in the project because it is the most used in the healthcare organizations. Besides, there are open source tools that implement this specification, it eases the development of the HL7 communication module.

This standard is supported by every major medical information systems vendor [5].

2.1.1.2 V3

Being this version of the HL7 specification that is based on a model, it is only logic to begin to explain the information model that supports the generation of the standards of this version: the Reference Information Model (RIM). This means that both the messaging standards in V3 and the docummentation standards in V3 (e.g. CDA, CCD, etc.) are all based in RIM.

RIM is the corner stone of the HL7 V3 development and a fundamental part of their methodology. This model express the necessary data in each clinical or administrative context and provides a representation of all the existing connections between the message fields.

We can say that, if HL7 is a language, the Reference Information Model is the grammar, that allows the creation of the correct phrases to the language in question and specifies the relation between words.

RIM appears to be pretty simple as it's core is only made by just six classes:

- *Act* - Represents the actions that are executed and need to be catalogued when the services are provided;
- *Participation* - Represents the context of an act: who has done it, for whom it was executed, etc.;
- *Entity* - Represents the materials and physical beings that are of interest, and take part on the medical care;
- *Role* - Establishes the roles that the entities perform as participants in medical care services;
- *Act Relationship* - Represents the links between two acts, i.e., the relationship between an order for an observation and the occurrence of said observation;
- *Role Link* - Represents ten links between each individual role.

This standard defines, as the 2.x versions did, a series of electronic messages but in version 3 these messages are based on an XML encoding syntax. The complete shift in paradigm of the standard make it not backwards compatible. Switching to the new version of the HL7 is a very complex operation[6]

2.1.1.3 HL7 Parser

To make use of HL7 it's necessary to create and parse a great number of messages. To do this manually would be time-consuming because of the number of existent HL7 messages (over 500[7]). To address this issue an API called NHapi will be used.

NHapi is a .Net Version of Hapi (an HL7 framework for java) used to build and decode HL7 V2.X messages. It acts as a parser of the HL7 messages. This object model allows for parsing and encoding HL7 2.X data to/from pipe(—) delimited or XML formats.

2.2 User Interface

The user interface is an essential part of the program. It presents information to the user as a form of output and it acknowledges the input from the user. This allows the system to indicate the effect of the users input and other information.

2.2.1 Rich Internet Applications

Rich Internet Applications (RIA) are web applications that have some characteristics of desktop applications. These similarities are in the interface and seamless communication. These applications support such as Ajax, web browser plug-ins (such as adobe flash and Microsoft Silverlight) and JavaScript, or via sandboxes or virtual machines.

Microsoft Silverlight

Microsoft Silverlight is a programmable web browser plug in that enables advanced features that characterize rich internet applications. Microsoft Silverlight user a familiar technique to go beyond the capabilities of standard web pages: a lightweight browser plug-in. Microsoft Silverlight was created to improve the way .NET developers create websites without ever needing to leave the same IDE[1]. This was developed to facilitate the creation of RIA content inside the .NET framework[8, page xxv]. Silverlight exists in two versions. The first version, Silverlight 1, is a relatively modest technology. It includes several features but lack support for Common Language Runtime (CLR) and .NET languages, so any code written by the programmer has to be in JavaScript. Currently the second version, Silverlight 2 adds the .NET-powered features such as CLR and a subset of

[1]Flash needed a completely different language (ActionScript) and a different IDE (Flex)

.NET Framework classes, and a user interface model based on WPF[8, page xxvii]. A Silverlight version 3 Beta is under production and it will improve the Silverlight capabilities even further[9].

Adobe Flash

Adobe Flash[10] is a popular multimedia platform that is used to add animation and interactivity to web pages. Flash is used for a variety of things, most recently, to develop rich internet applications[11] (RIA) using the web browser plug-in.

2.2.2 Desktop Applications

Desktop Applications run on top of the operating system and are not dependent on a browser or on the internet to start. Normally the program is by itself a client connected to a server using centralized databases and operations.

Windows Presentation Foundation

Windows Presentation Foundation (WPF)[12] is a graphical subsystem in .NET Framework 3.0 that uses a markup language (XAML) for the interface development. WPF provides a consistent programming model for building applications and provides clear separation between the user interface and the business logic. It enables several rationalities in a single solution such as user interface, 2D and 3D drawing, fixed and adaptive documents, advanced typography, vector graphics, raster graphics, animation, data binding, audio and video.

WPF is considered the successor of the Windows Forms and is promoted for line-of-business applications.

2.3 Back-End Technology

The back-end is related to the business logic of an application. It drives the user interface, providing and collecting all the necessary data for the program in question. It involves all the technology that is hidden from the end user.

2.3.1 Windows Communication Foundation

Windows Communication Foundation (WCF) is a programming framework used to build applications that inter-communicate. WCF unifies the various communication programming models supported in .NET 2.0, into a single model. Released in November 2005,

.NET 2.0 provided separate APIs for SOAP-based communications for maximum interoperability, binary-optimized communications between applications running on Windows machines, transactional communications, and asynchronous communications. WCF unifies the capabilities from these mechanisms into a single, common, general service-oriented programming model for communications[13].

WCF is designed in accordance with Service oriented architecture (SOA) principles to support Distributed computing where services are consumed by clients. Clients can consume multiple services and services can be consumed by multiple clients.

2.3.2 Windows Workflow Foundation

Windows Workflow Foundation (WF) is a framework for defining, executing, and managing workflows. Workflows have several advantages over traditional programs[14, page 2]. They can handle long running work by persisting to a durable store when idle and loading again once there is new work to do, the instances of the workflows can be changed dynamically in run-time, they allow to declare business rules that are separated from the code making it easier future changes.

Pageflow

Pageflow if a specific implementation of the WF, it is designed with the objective of modelling the user interaction with an application using a workflow. Microsoft showed Pageflow at TechEd[15] and released it as a sample application. It implements a generic navigation framework that can support at the same time multiple UI (such as ASP.NET or WPF) Currently the Pageflow sample only supports one active workflow per instance and by default supports ASP.NET and WPF as UI on top of the workflow.

2.4 Object-Relational Mapping

The concept of Object-Relational Mapping (ORM) is a database abstraction layer is an application programming interface which unifies the communication between a computer application and databases. These communications are handled via an API that provides consistent calls to any database while hiding the database specifics as much as possible. There are many abstraction layers with different interfaces in numerous programming languages.

2.4.1 NHibernate

Nhibernate is a C# based ORM framework based on the original Hibernate for java that provides a abstraction layer between the database and .NET application. It provides provides an abstraction that is important because it allows the use of a variety of relational databases while maintaining the same code in the application.

NHibernate was chosen as a Object-relational mapping framework because it provided out of the box compatibility with all the major relational databases vendors.

2.4.2 Language Integrated Query

Language Integrated Query (LINQ) is a Microsoft .Net Framework component that adds native data querying capabilities to .Net languages. LINQ can also provide a abstraction, like NHibernate, between the database and the back-end. It has many other functionalities that are out of the scope of this project.

2.5 Patient Administrative Management Implementations

2.5.1 Integrating the Healthcare Enterprise - Patient Encounter Management

To develop Encounter Management (EM) software, an HL7 standard it's needed. It is used to inform all the patient administrative events, enabling them to synchronize their application database with the latest information available. The group of messages that is sent from EM are the HL7 ADT messages (admission, discharge and transfer). The Integrating the Healthcare Enterprise (IHE)[16] creates profiles that support certain functionalities. To implement this profiles a set of HL7 messages need to be supported. These messages are called triggers, and there are triggers for every event that occurs in a hospital such as admit, transfer, discharge or even a change in the patient demographic information. These triggers represent actions that can be executed within the HL7 standard (more information in the appendix A). The most useful configuration for each trigger is going to be studied and analyzed to provide state-of-the-art HL7 compatibility.

The standard does not explicit a sequence in which these triggers can be called, but is quite clear that a patient has to be admitted before he can be transferred and discharged. In the appendix A are defined several subsets of trigger events[17] that are required to implement said features.

Typical Clinical Workflows

Typical scenario:

1. Patient received at Emergency room by attending doctor U (The sent HL7 message is the ADT_A04 – Register outpatient).

2. Doctor U admits de patient (The sent HL7 message is the ADT_A06 – Change patient class to inpatient), into location BB, referring him to attending doctor X.

3. The patient is moved to location GG (The sent HL7 message is the ADT_A02 – Transfer), keeping X for attending doctor.

4. The patient is discharged and leaves de hospital (The sent HL7 message is the ADT_A03 – Discharge).

These 4 real world events are expresses with 5 trigger events / messages, two of which occur at the same time (step 2). Here the episode will be divided into 3 encounters as shown in table 2.1 at page 12. For more information on the sent HL7 messages refer to the appendix A.

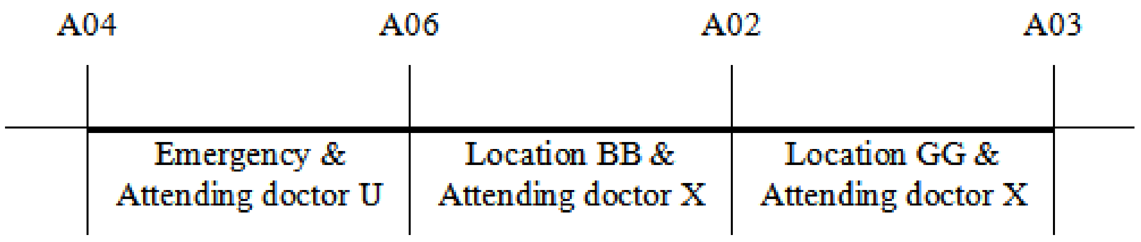
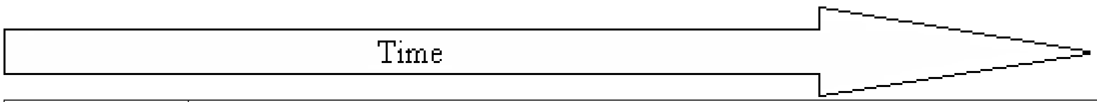

A04
A06
A02
A03
Emergency & Attending doctor U
Location BB & Attending doctor X
Location GG & Attending doctor X

Table 2.1: Hospital Workflow Example

2.5.2 Global Patient Access (Siemens) - Encounter Management

Global Patient Access is a software for Patient Management. It is going to be analysed only the part of the software related to the Encounter Management.

According to GPA – EM every independent visit to a healthcare facility can be grouped in episodes of care. In the table 2.2 at page 12 is an example of the episode of care workflow.



Episode of Care	Episode of Care Patient A				
Encounter	Emergency Encounter	Inpatient Encounter		Outpatient Encounter	Outpatient Encounter
MRD	Emergency Department	Medical Responsible Department 1	Medical Responsible Department 2	Medical Responsible Department 3	Medical Responsible Department 3
ARHO	Administratively Responsible Healthcare Organization				

Table 2.2: Example of Episode of Care Workflow

2.5.2.1 Encounter

Encounter replaces the usual concept of patient visit. Encounter is the smallest unit of the patient's interaction within a healthcare organization. It is located in time: past, current, planned meeting or interaction. It takes place between a patient and one or more healthcare professionals. The encounters exist to provide health-related services or for assessing the health status of a patient.

The status of each encounter can be:

- Checked Out;
- Cancelled;
- Deleted;
- Checked In;
- Scheduled.

The encounters are assigned to an healthcare professional, such as admitting, attending, and consulting. There are a few restrictions on these assignations and a history is retained of the changes.

The types of encounters that can occur within a hospital context are:

- Inpatient;
- Emergency;
- Observation;
- Outpatient;
  - Recurring;
  - Office/Clinic;
  - Day Surgery;
  - Pre-Admission Test;
- Visitor.

2.5.2.2 Episode of Care

Episodes of care groups all the encounters for one patient. These episodes comprise a continuous contact with the patient by a Healthcare Organization. Episode of Care replace the concept of a *master visit* with *sub-visits*. An Episode of care may consist of multiple encounters (such as Emergency Room to Inpatient, Medical, Surgical, and Neurology). Episodes of Care can also be linked (example, Mother → newborn baby) to another.

An Episode of Care belongs to one, and only one, administratively responsible healthcare organization. The entire encounter in an Episode of Care must be time-continuous, with no time gaps. However, overlaps are permitted. Pending discharge indicator can only be true if the Episode of Care has an inpatient encounter with a status of checked in.

2.5.3 ACSS - Sonho - The Portuguese Database

The *Administração Central do Sistema de Saúde* (ACSS) is the Portuguese government entity that manages the Portuguese Healthcare System. They have just released[18] a new specification of an application to integrate the Portuguese healthcare database named *Sonho*. This database is used in almost every Portuguese hospital or healthcare facility. The Glossary introduces some of the Portuguese concepts.

2.5.3.1 Functional Description

The following subsections show the various concepts that ACSS defines. There is a short description and a list of activities that are involved in each concept.

Emergency Episode

The main features of this module are:

- Admission;
- Complete/Change case details;
- Associate an episode to a patient (error or other situations);
- Discharge;

Outpatient Episode

The main features of this module are:

- Pre-visit;
- Appointment Check-In;
- Discharge of Appointment;

Inpatient Episode

The main features of this module are:

- Admission;
- Complete/Change case details;
- Discharge;

Ambulatory Episode

The main features of this module are:

- Appointment/Session Check-In;
- Discharge bulletin;

Surgery Episode

The main feature of this module is:

- Transfer/Admit to surgery;

2.6 Planned Architecture

The ideal architecture for the EM system would be a distributed, heavily service oriented architecture (SOA).

The figure 2.2 at page 16 shows this architecture and the way it implements all the services in a modular way. The client (WPF App) requests the next activity of the workflow from the Client AppController and the XML for the XAML (User interface) would be downloaded from the IWizard Builder that generates the necessary XAML code to be used in the presenter. The available workflows are distributed by the Server AppController. This can be shown in any presenter (in this case Windows Presentation Foundation is used) and all the information can come from various related services.

An example of an application running can be seen in figure 2.3 at page 16 and clearly shows the workflow activity that runs inside the WPF Window. The window to be shown is built using a WPF Wizard Builder (or ASP.NET if it is a Web Page) and provided through an Event Broker that handles the communications between all the modules. The information of the window that is displayed comes from the App Controller that communicates between the WF, the Wizard Builders and the Event Broker.

The final purpose of this architecture is the separation of work between the designer, the developer and the medical analyst. In the figure 2.4 at page 17 it is clearly shown the

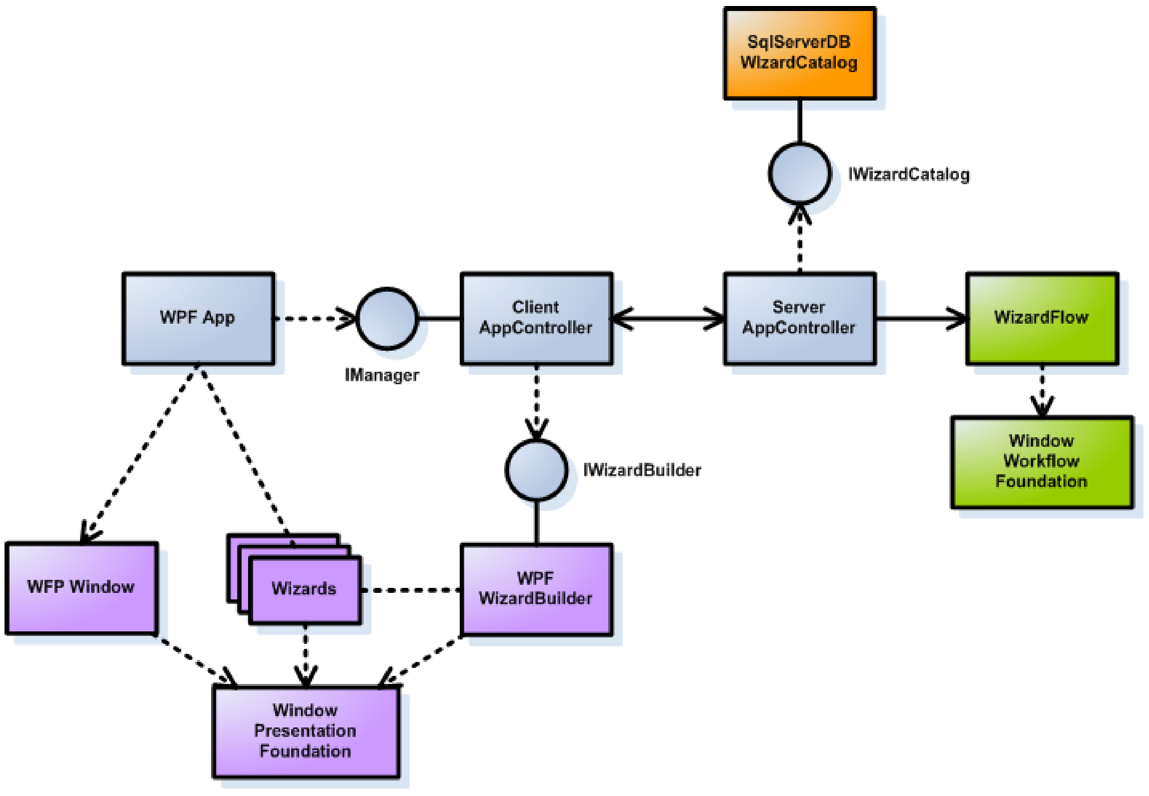


Figure 2.2: Ideal Architecture Overview

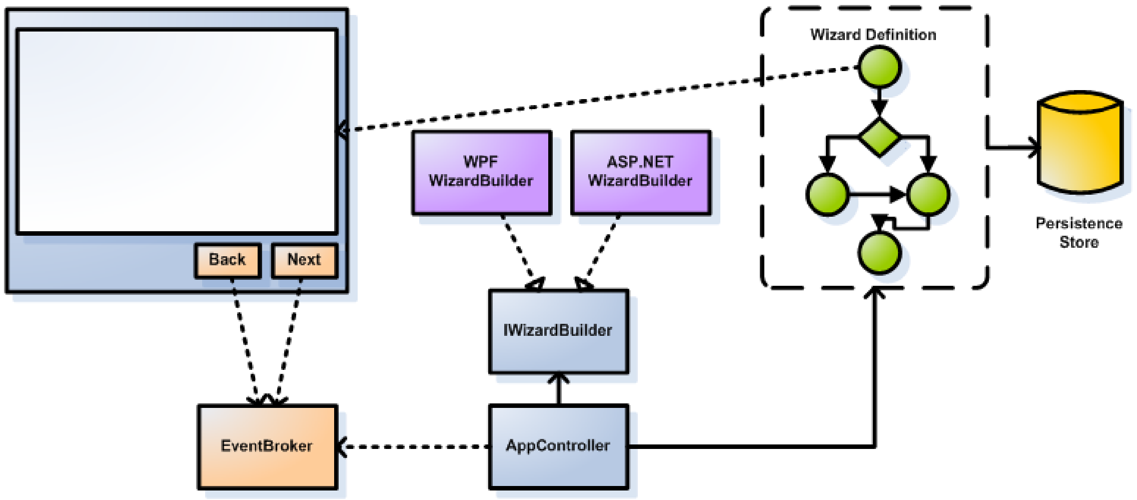


Figure 2.3: Workflow execution

separations between the three roles. A designer can only work in the interfaces, the developer can make the connections between the workflow and the interface, and the workflow can be build by a consultant from the healthcare facility and improve its accuracy with the models of the healthcare facility. This way the software becomes fully modular with role separation in the all stages of development.

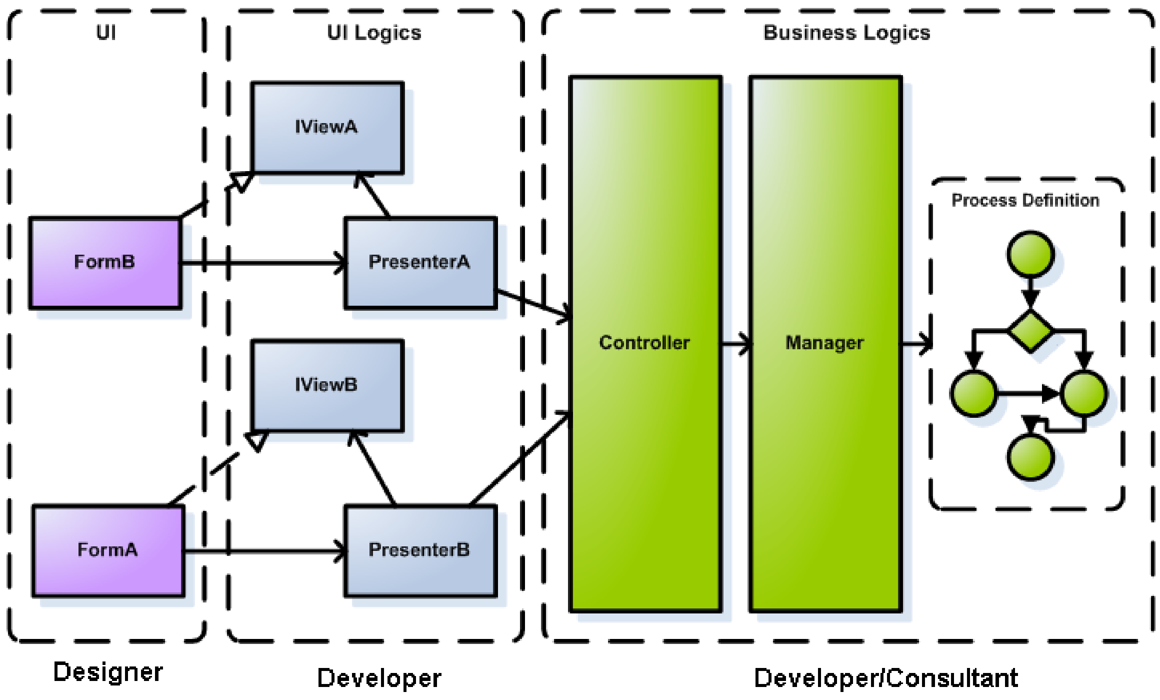


Figure 2.4: Architecture Separated by roles

2.7 Solutions on the Market

2.7.1 Alert (Alert Life Sciences)

Alert is a Portuguese software house that develops software related to healthcare. The current interoperability policy makes alert compatible with IHE, HL7 and other medical related standards[19].

ADT functionality is spread across the modules that are related to ADT (such as Alert Outpatient, Alert Inpatient, Alert EDIS and Alert Oris)

2.7.2 E-Doctor (Hewlett-Packard)

This software was developed around the idea that paper is cumbersome and expensive. The thought that a digital record is far easier to manage and also cheaper is the foundation of this software. That said, the HP HIS relies on a proper infrastructure to provide its full functionality.

One of the innovation feature is the integration of RFID on surgery and on transfers. The identification of the patient can be made directly from the patient's RFID.

2.8 Conclusions

This chapter started by presenting the existing healthcare standards, the advantages and disadvantages between them. It presented multiple options for user interface development and several back-end technologies. It detailed a few implementations of the healthcare standards. It also introduced the ideal architecture for the software and some solutions in the market.

The chosen technologies for the implementation of the Encounter Management are showed in section 5.1.1.1.

There are a lot of studies in the healthcare standards. These provide critical information for the design phase that it is going to concern with concepts and the modelling of the solution. There are also a great number of technologies that can help the development process of this project. These technologies range from back-end frameworks (e.g Windows Workflow Foundation) and front-end frameworks (e.g. Windows Presentation Foundation). These frameworks help to speed up the developing process by enabling fast prototyping and providing rich APIs.

Chapter 3

Problem Description

This chapter presents the problem that is presented by this project, the project requirements and what it aims to achieve. Finally it includes priorities and a schedule for the task of the project.

3.1 The Problem

The problem that is presented by this project is tied to the need of hospital informatization[20]. This accounts for electronic patient data and to find a way to unify the health system so that is fully integrated and the information can be exchanged easily between medical responsible departments and ultimately between hospitals[1]. This leads to the need of standardization of the information.

The problem in the Encounter Management project involves a couple of steps. First it is necessary to define the architecture that supports these encounters in a way that would turn out to be flexible and adapted to all the medical activities that are performed in a healthcare facility. Then it is needed to implement the architecture defined earlier in a way that is flexible enough to be deployed in all the necessary scenarios, with localization so that the software can be available in different languages dynamically in a way that it does not need a completely separate program for each language. It is also necessary to develop the user interface that satisfies the end user.

The end-goal is to provide a set of abstract activities that are performed in the context of a healthcare facility. These activities can then be tied together so that they perform a *workflow*. These workflows are activities in a healthcare facility such as patient check-in (for inpatient, for emergency, for external appointments, etc), patient check-out, transfers, edition of patient's demographics and other administrative tasks inside a healthcare facility. These workflows have common activities[2] that are shared among them, reducing the

[1] According to current laws

[2] Such as Editing Patient Demographics of Patient selection

number of needed interfaces and simplifying the construction of new workflows or the adoption of existent ones to new business rules.

Episode of Care, Episodes and Encounters

An essential step in defining this problem is a clear definition of each term. This definition has to be found and the relation of each of the terms completely defined so that a standard can be defined and the previously mentioned interactions can be accomplished.

The very name of the project, Encounter Management points to a focus in the encounters[3], the definition of those encounters are ambiguous because an encounter can be an interaction from a patient with a healthcare facility that endures through the day and also that each encounter is one individual interaction from that patient with the healthcare facility.

3.2 Project Requirements

An important step in software development is the Project Requirements. The Project Requirements are essential for a good project outcome[21]. In this chapter is going to be presented the use cases[4] needed to implement the Encounter Management.

3.2.1 Encounter Management

The workflow module is going to be the core of the entire system. It provides features that support the Encounter Management concept. It is going to provide an interface that follows the intentions of the users through the workflow. It provides useful features such as suspending a workflow in a certain state and resumes it later. It provides a workflow based interface to execute the desired Encounter Management Related Activity.

The features of the Encounter Management are separated in two main areas, the Inpatient (for Patients that stay at the hospital for more than twenty four hours and are in the Inpatient state) and the Outpatient (for Patients that attend appointments, go to emergencies, hospital daycare staying in the hospital no more than twenty four hours and are in the Outpatient state).

The following use cases will show the most common solutions that suit a healthcare facility. Custom development can be made to meet the clients needs and published via the workflow builder that is presented in the next section.

[3]Encounters some times are called Visits

[4]A use case is a description of a system's behaviour as it responds to a request that originates from outside of that system.

Use Cases

The use cases are divided in three areas. In the figure 3.1 at page 21 is showed the general use cases of the Encounter Management.

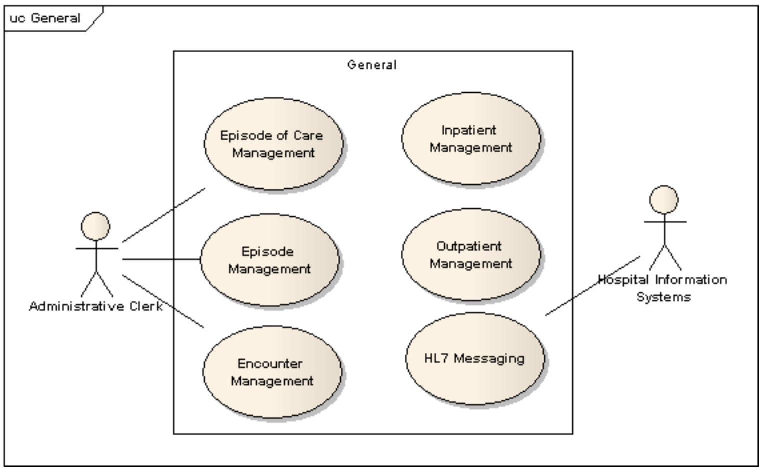


Figure 3.1: Encounter Management - High level

This figure shows the basic use cases for the system. The Encounter Management software needs a secure User Authentication system. It also needs to be able to perform the basic tasks referred as Episode Management (Creation, Changing, Delete and the other use cases in the figure). These use cases are the base for the Inpatient and Outpatient use cases. They allow that patients can be admited/checked in by adding patient episodes and discharged/checked out by changing a patient episode defining the end of the episode. Other particular systems can be developed and created using the workflow builder.

The figure 3.2 at page 22 shows the use cases for the Inpatient.

The Inpatient is a particular model of providing healthcare knowing that the patient is allays under healthcare surveillance. This surveillance extends to every patient care such as drug administration, critical care and bed tracking. In this figure, Patients can be admitted to the system, using the patient listing, viewing and editing patient demographics. A bed management software handles the beds that have been booked through the admit process. This process can be cancelled through the Cancel Admit Patient. When necessary a patient can be discharged from the facilities ending the encounter. This action can also be cancelled. An Inpatient can also be transferred to another Medical Responsible Department (MRD) or other healthcare facility. This activity can be cancelled. There is a use case for a Pre-Admit Patient that is used when the patient needs to do exams prior to the actual Admit. This activity can also be cancelled. After each activity an HL7 Message can be sent to inform the registered systems.

The figure 3.3 at page 23 shows the use cases for the Outpatient.

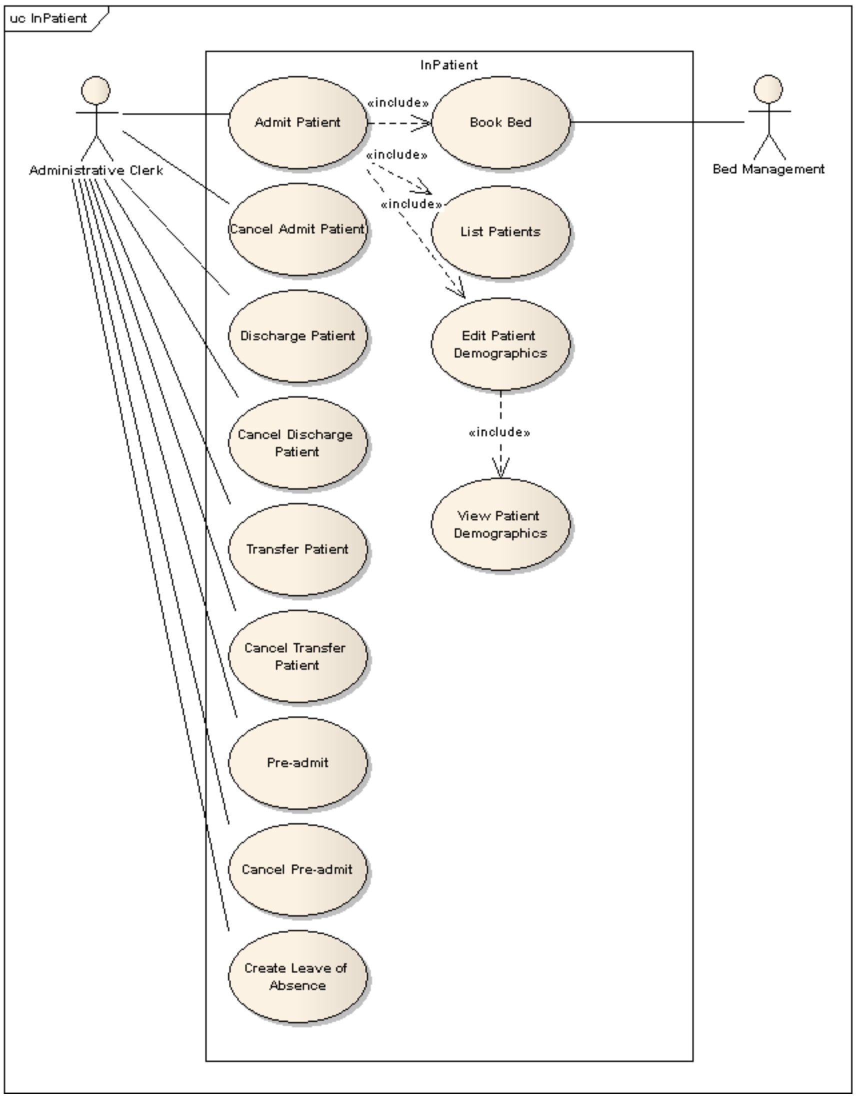


Figure 3.2: Inpatient - Detailed Use Case

The Outpatient is a particular model of providing healthcare knowing that the patient isn't under healthcare surveillance. The patient is not under constant surveillance, does not have an assigned bed and can only remain inside the healthcare facility for a duration no longer that twenty four hours. In this figure the patient can be checked in for appointment that makes the patient available to attend the appointment, this uses the patient list, checks the insurance for the appointment in question and can also view and edit the patient demographics. The check in can be cancelled if desired. The check-out is also available with the restriction that the patient has to be checked in to be checked-out. This activity can also be cancelled. All these four activities are followed by an HL7 Messaging system to all the registered systems.

For example in case of Siemens an external application called Soarian Scheduling

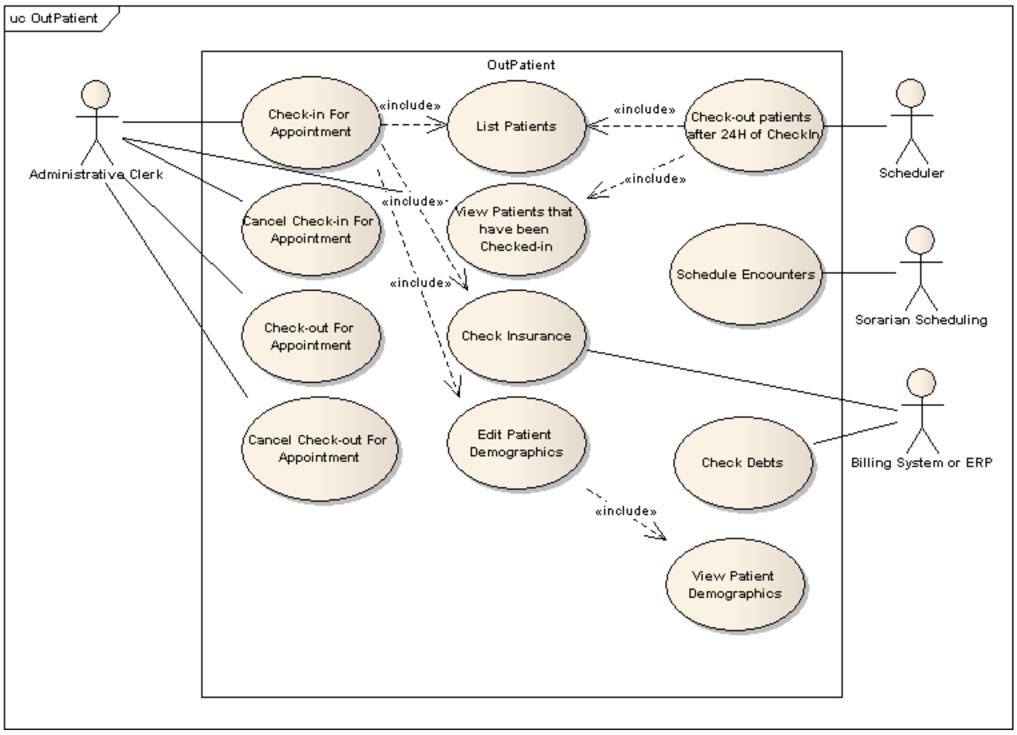


Figure 3.3: Outpatient - Detailed Use Case

schedules the encounters that are inserted into the system. A Billing System or ERP validates the Insurance and checks the patients debts. A periodic scheduler checks if the patients stay in the healthcare facility longer that twenty four hours. If there is any patient in this condition a automatic check-out is made. This is made by checking the patient list and the patients that have been checked in.

3.2.2 Workflow Builder

The Workflow Builder is a separate application that builds and deploys the workflows for use in the Encounter Management. This allows the medical analyst to tweak existing workflows and seamlessly develop new workflows to support the new medical workflows that emerge in the needs of the healthcare facility. The creation of these workflows would be done by dragging existent activities (that were previously created by the development team) into the workflow and creating the connections between the activities. This allows for a complete independent platform building without the need of a programmer and without needing any programming knowledge. This allows the role separation that was mentioned in section 2.6.

Use Cases

The figure 3.4 at the page 24 shows the use cases for the Workflow Builder.

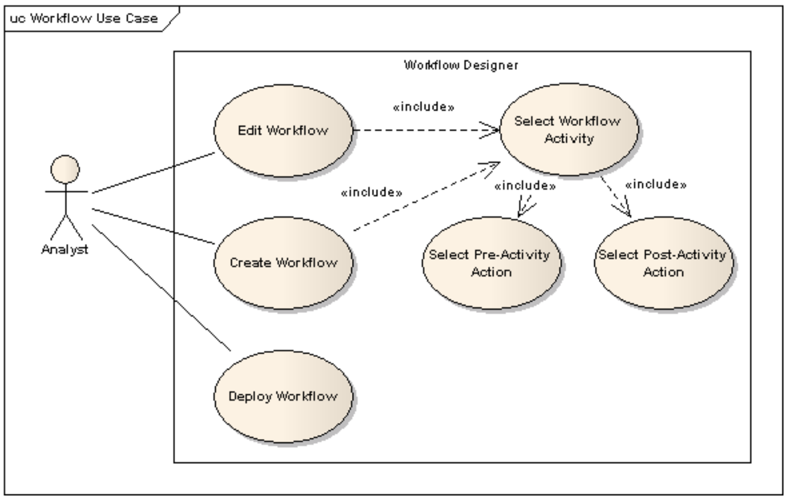


Figure 3.4: Workflow Builder - Detailed Use Case

These workflow builder use cases is a workflow editor that can deploy workflows to the program. This feature allows medical analyst to adjust and create new workflows. The creation of the workflows is done by selecting the required activities and the correspondent Pre-Activity action (that executes when the workflow action is called before the user interaction) and the post-activity action (that executes after the workflow activity when the user has done the input for the required user interface).

3.2.3 HL7 Messaging Interoperability

The HL7 Messaging standard is need to provide the communication between the Encounter Management Software and other clients such as Master Patient Index (MPI). This communication is essential to accomplish the interoperability needed by this application.

Use Cases

The figure 3.5 at the page 25 shows the use cases for the HL7 messaging interoperability.

The HL7 messaging interoperability module represent the HL7 communication that is used across the application to notify all the required applications of the informations that are processed inside the Encounter Management. It Sends HL7 messages, that are requested by the EM. This needs to generate the HL7 message and to receive the respective HL7 response. It also has a standalone server that is waiting for incoming HL7 messages. It receives these HL7 Messages, parses the HL7 message and then sends the right response again in HL7 format.

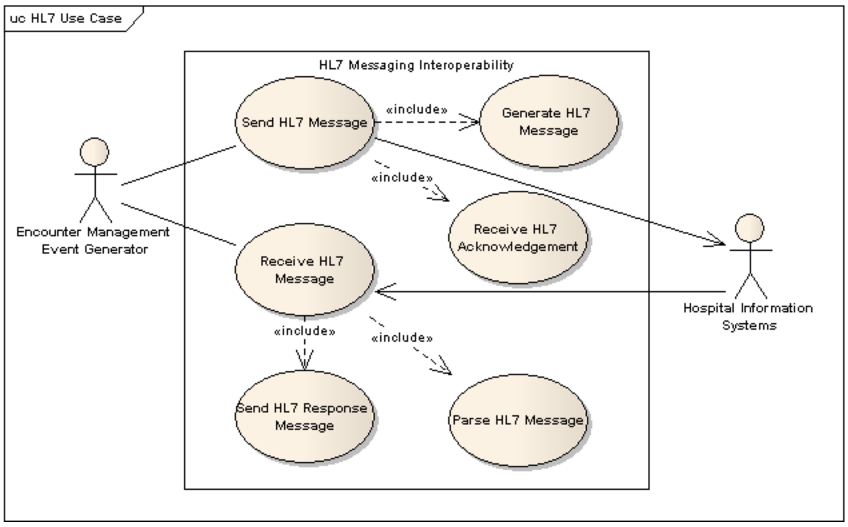

uc HL7 Use Case
HL7 Messaging Interoperability
Send HL7 Message
«include»
Generate HL7 Message
«include»
Receive HL7 Acknowledgement
Encounter Management Event Generator
Receive HL7 Message
Hospital Information Systems
«include»
«include»
Send HL7 Response Message
Parse HL7 Message

Figure 3.5: HL7 Messaging Interoperability - Use Cases

3.3 Schedules and Deliverables

The definition of a schedule and the priorities list for the use cases is one important step in every project. The milestones were already established in the beginning of the project, providing a good understanding and control over the project.

3.3.1 Priorities

The tasks in a project need to be prioritized, this step is very important to insure the correct focus of the developer.

There were several factors that were taken into account to decide the priority of each use case such as the developing difficulty, importance for the project to be developed and estimated task duration. The priority levels are Low, Medium and High were used.

The priority of the General Use Cases for the Encounter Management are shown in the table 3.1 at page 25.

Name	**Priority**
Episode of Care Management	High
Episode Management	High
Encounter Management	High
Inpatient Management	Medium
Outpatient Management	High
HL7 Messaging	High

Table 3.1: Priorities for the General Use Cases for the Encounter Management

The priority of the Inpatient Use Cases for the Encounter Management are shown in the table 3.2 at page 26.

Name	**Priority**
Admit Patient	Medium
Cancel Admit Patient	Low
Discharge Patient	Medium
Cancel Discharge Patient	Low
Transfer Patient	Medium
Cancel Transfer Patient	Low
Pre-admit Patient	Low
Cancel Pre-admit Patient	Low
Book Bed	Low
List Patients	High
Edit Patients Demographics	High
View Patients Demographics	High

Table 3.2: Priorities for the Inpatient Use Cases for the Encounter Management

The priority of the Outpatient Use Cases for the Encounter Management are shown in the table 3.3 at page 26

Name	**Priority**
Check-in For Appointment	High
Cancel Check-in For Appointment	Medium
Check-out For Appointment	High
Cancel Check-out For Appointment	Medium
List Patients	High
View Patients that have been checked-in	High
Check Insurance	High
Edit Patients Demographics	High
View Patients Demographics	High
Check-out patients after 24H of check-in	Medium
Check Debts	Low

Table 3.3: Priorities for the Outpatient Use Cases for the Encounter Management

The priority of the Use Cases for the Workflow Builder are shown in the table 3.4 bellow at page 27.

Name	Priority
Edit Workflow	Medium
Create Workflow	Medium
Deploy Workflow	Low
Select Workflow Activity	Medium
Select Pre-activity Action	Medium
Select Post-activity Action	Medium

Table 3.4: Priorities for the Workflow Builder

The priority of the Use Cases for the HL7 Messaging Interoperability are shown in the table 3.5 at the page 27.

Name	Priority
Send HL7 Message	High
Receive HL7 Message	Medium
Send HL7 Response Message	Medium
Generate HL7 Message	High
Receive HL7 Acknowledgement	High
Parse HL7 Message	Medium

Table 3.5: Priorities for the HL7 Messaging Interoperability

3.3.2 Schedule

The schedule of this project was divided in 5 phases:

1. Validate existing outputs for the project;
2. State of Art study;
3. Implementation;
4. Development of necessary documentation;
5. Final Development.

In the end of each phase a milestone is achieved.

Validate existing outputs for the project

The validating phase took about one week (2 March - 6 March). It consisted on understanding the problem, analysing existent documentation and to get familiarized with the medical terms. The result of this phase was a complete integration in the development team, better understating of the company's philosophy and the medical terms involved in the project. The milestone of this phase was the clear knowledge of the most important medical terms.

State of Art study

The State of Art study phase took about three weeks (9 March - 27 March). It consisted in the study of all the available technologies, international health standards, existing applications and a prototype that identified the phases of some process in a healthcare facility. This phase resulted on the State of Art document and the Vision document, this phase also produced a presentation for the promotion of the project inside Siemens. The milestone of this phase was the state of art document.

Implementation

The implementation phase was the longest one with ten weeks (30 March - 6 June). It consisted in the implementation of the architecture that supported the application and the creation of some hospital workflows on top of the architecture. Some project related documentation was updated or adjusted to fit the time frame. The milestone of this phase was a working prototype.

Development of necessary documentation

This phase took about a month (8 June - 29 June). This time was reserved for the production of software related documentation and this thesis. The milestone of this phase was this thesis.

Final Development

This phase will take the remaining time, one month (30 June - 31 July). It will consist in the creation of the final presentations to be showed at FEUP and at Siemens. Small adjustments in the implementation and documentation could also be necessary until the end of the duration of this project (31 July). The milestone of this phase is going to be a reviewed thesis and a new version of the prototype.

3.4 Conclusions

This chapter began by presenting this project's goal of creating an Encounter Management software. It presented the requirements of this software. It ended with the schedule and the deliverables.

This chapter clearly described the dimension of this project. The healthcare software Encounter Management is used in all the medical responsible departments (MRD) inside a healthcare facility needing a complete specification of the activities that can happen in each MRD. These activities were prioritized in order to focus on each MRD making the top priority the outpatient MRD.

Chapter 4

Solution Specification

Based on the requirements for the project and on prototyping work done during the research and technology evaluation phase, a specification of the proposed solution was produced.

4.1 Medical Terms

To define a solution for the problem, first it is necessary to correctly define the medical terms or concepts that are going to be used in the implementation. These definitions were defined based on extensive research of existing healthcare programs and the European and Portuguese legislation in the area of hospital administration and specific encounter terms.

4.1.1 Episode of Care

An Episode of Care acts as a container that holds Episodes (going to be explained in the next chapter). The episodes that are inside an Episode of Care are all tied to the resolution of a specific condition with the patient.

If the patient goes to the hospital because of a new problem then a new Episode of Care is opened for that condition in particular and then all the actions that are taken regarding that condition will be stored in that Episode of Care in particular.

This definition supports the idea that one Episode of Care contains every action that was made to improve a certain condition[1] of a patient. This way a patient can have multiple Episodes of Care that are opened at the same time and are addressing different conditions. This definition provides the flexibility and adjustability that are necessary to support all the patients' conditions.

An episode of care has these common properties:

- **Referral** - Who referred to this episode of care;

[1]This condition could be a group of problems that affect the patient's health

- **Start Date** - Date on which the episode of care starts;
- **End Date** - Date on which the episode of care ends;
- **Description** - Information regarding the episode of care.

4.1.2 Episode

The episodes are contained inside an Episode of Care as mentioned in section 4.1.1. The Episodes are a container for Encounters (that represent the smallest interaction within a healthcare facility). The Episodes have an associated type which refers to what kind of Encounters they have (or can have). These types refer to every medical responsible departments (MRD) inside a healthcare facility and within them can be inserted (via Encounters) the operations that can be executed in said departments.

Here is a list of episode types and some properties that they hold:

- Emergency Episode . Described in figure 4.1 at page 31
  - Does not have an assigned bed;
  - Maximum duration of 24h[2].

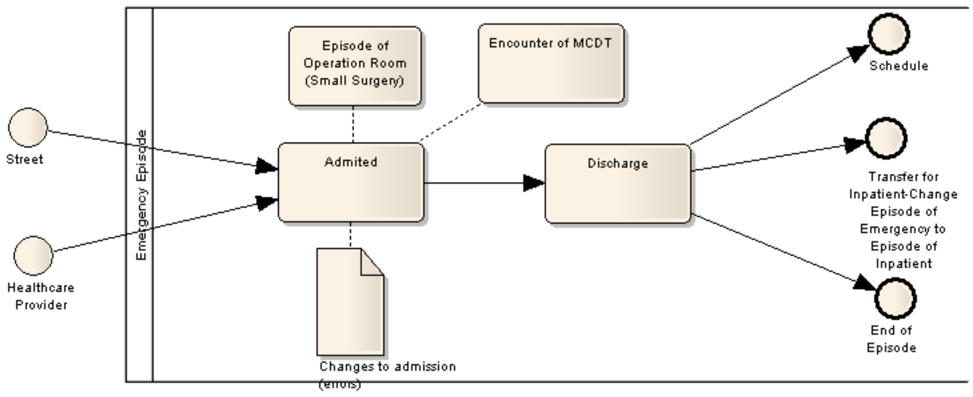

Emergency Episode
Episode of Operation Room (Small Surgery)
Encounter of MCDT
Schedule
Street
Admited
Discharge
Transfer for Inpatient-Change Episode of Emergency to Episode of Inpatient
Healthcare Provider
End of Episode
Changes to admission (errors)

Figure 4.1: Flow of the patient in the Emergency Episode

- Outpatient Episode. Described in figure 4.2 at page 32
  - Does not have an assigned bed;
  - Typically to perform appointments and exams through pre-admission.

- Inpatient Episode. Described in figure 4.3 at page 32
  - Must have an assigned bed;
  - During periods greater than 24h for the various reasons.

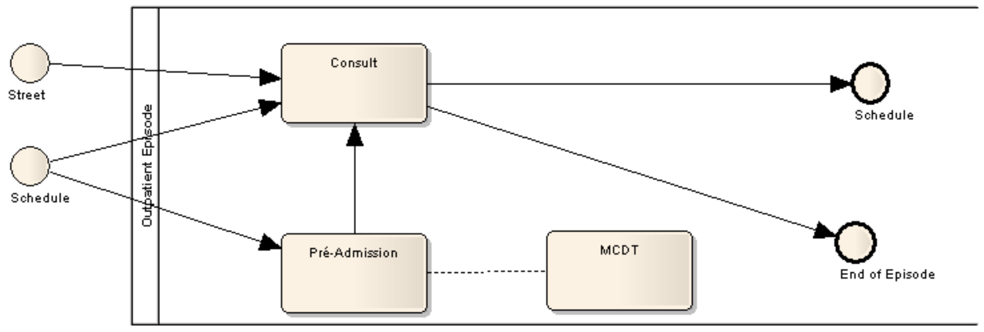


Figure 4.2: Flow of the patient in the Outpatient Episode

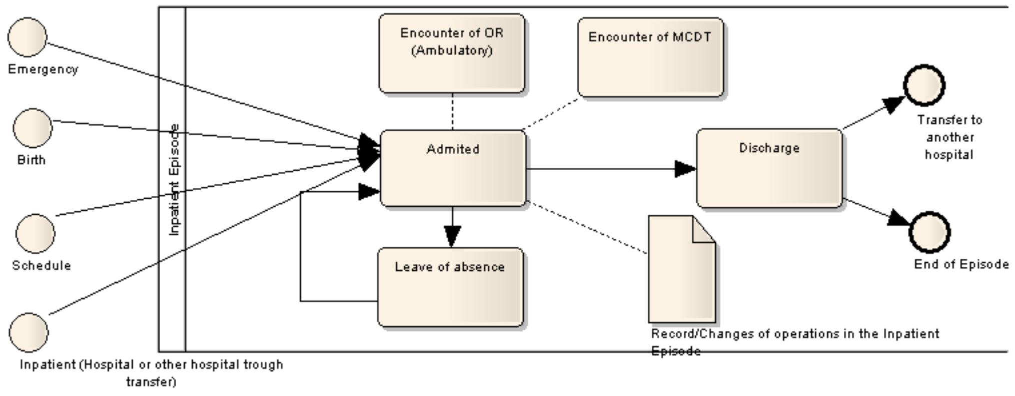

Inpatient Episode
Emergency
Birth
Schedule
Inpatient (Hospital or other hospital trough transfer)
Encounter of OR (Ambulatory)
Encounter of MCDT
Admited
Leave of absence
Discharge
Record/Changes of operations in the Inpatient Episode
Transfer to another hospital
End of Episode

Figure 4.3: Flow of the patient in the Inpatient Episode

- Day Care Episode. Described in figure 4.4 at page 32
  - Does not have an assigned bed;
  - For small treatments and procedures.

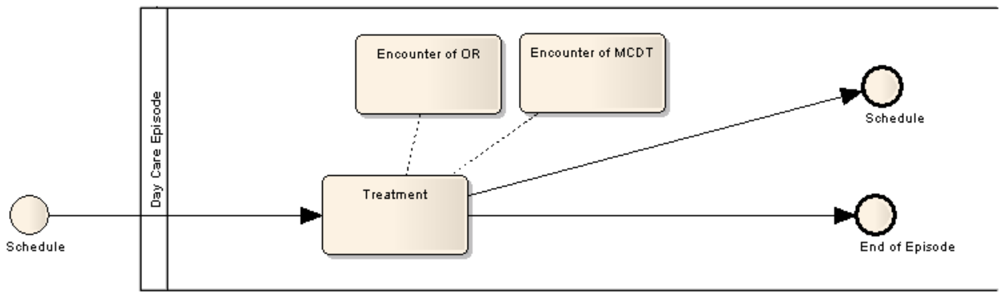


Figure 4.4: Flow of the patient in the Day Care Episode

- House Care Episode. Described in figure 4.5 at page 33
  - Does not have an assigned bed;

[2]According to legislation

  - Similar to daycare but it is done in the patient's location.

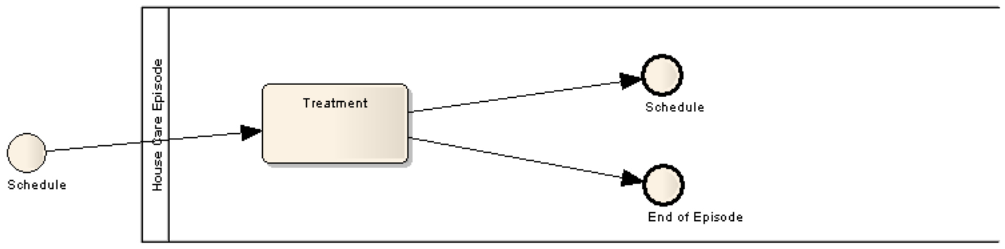


Figure 4.5: Flow of the patient in the House Care Episode

- Continuous Care Episode. Described in figure 4.6 at page 33
  - Does not have an assigned bed;
  - Continuous care to recover from a condition.

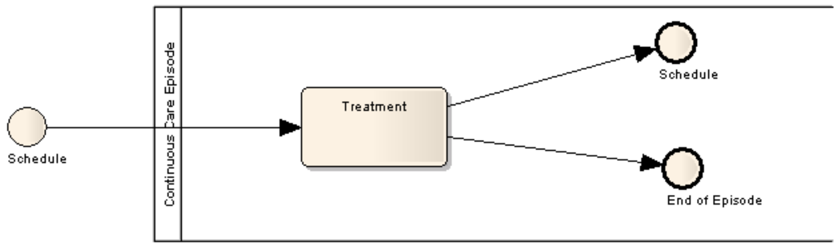


Figure 4.6: Flow of the patient in the Continuous Care Episode

An episode has these common properties:

- **Type** - Can be any of the defined above;
- **Start Date** - Date on which the episode starts;
- **End Date** - Date on which the episode ends;
- **Description** - Information regarding the episode.

4.1.3 Encounter

An Encounter is the smallest interaction between the patient and the healthcare facility. There are many type of encounters such as a simple appointment, to CMDT or surgery. These encounters are grouped into episodes that associated with the MRD.

Here is a list of a few[3] Encounters types and some properties that characterizes each type:

[3]Custom encounters can be created to better adjust to the healthcare facility custom procedures

- CMDT Encounter
  - Involves tracking inside the hospital;
  - Groups of exams and treatments of the same type that are done. consecutively
- Emergency Encounter
  - Typically included in the Episode of Emergency;
  - Does not have an assigned bed.
- Appointment Encounter
  - A scheduled appointment between a physician and a patient;
  - Does not have an assigned bed.
- Internal Medicine Encounter
  - Typically included in the Episode of Inpatient;
  - Must have an assigned bed.
- Intensive Care Encounter
  - Typically included in the Episode of Inpatient;
  - Must have an assigned bed.
- Surgery Encounter
  - Included in many types of Episodes, but each one with particular conditions;
  - Does not have an assigned bed.
- Treatment Encounter
  - Typically included in the Episode of Emergency;
  - Does not have an assigned bed.
- Traumatology Encounter
  - Typically included in the Episode of Inpatient;
  - Similar to Encounter of Internal Medicine but in a different MRD;
  - Must have an assigned bed.

All the encounters have these common properties:

- **Type** - Can be any of the defined above;
- **Facility** - Where the encounter occurred;

- **Start Date** - Date on which the encounter starts;
- **End Date** - Date on which the encounter ends;
- **Bed Number** - Bed that the patient occupies during the encounter (only for encounters that need an assigned bed);
- **Description** - Information regarding the encounter.

4.1.4 Interactions

To use this concepts of Episodes of Care, Episodes and Encounters they have to be linked in a way that was described in the sections above. In figure 4.7 at page 35 is an example of a Episode of Care to resolve a certain condition with the patient.

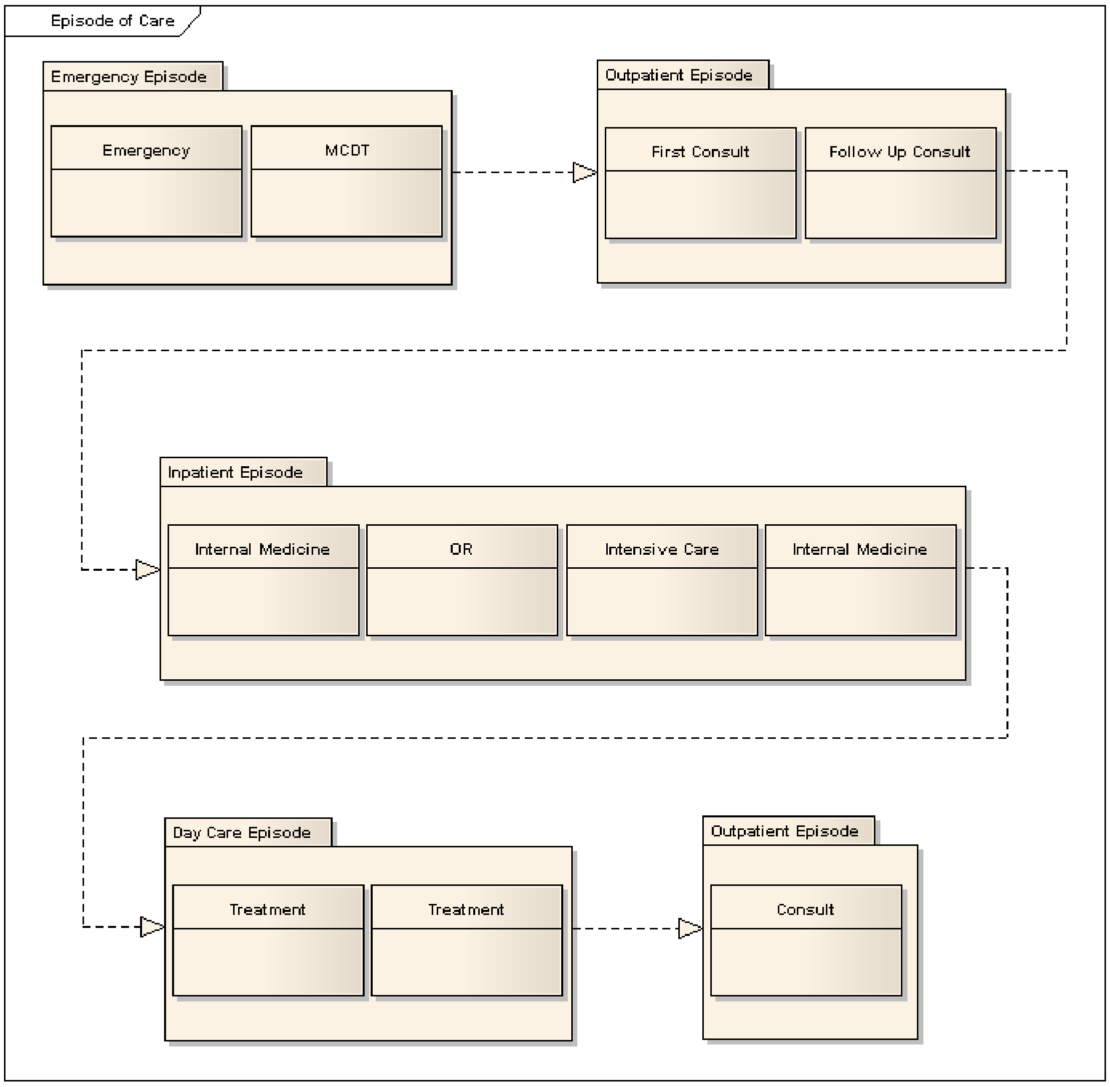


Figure 4.7: Example of Episode of Care

The relationships between the above medical terms is showed in figure 4.8 at page 36 with a brief description.

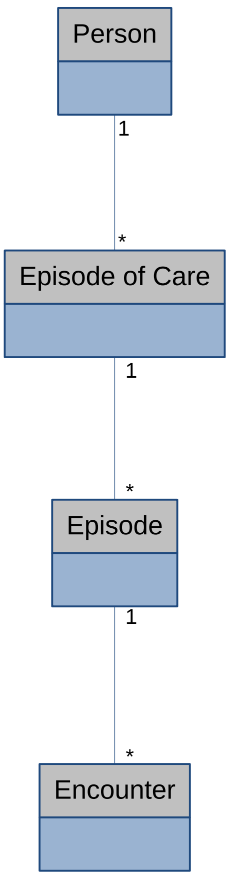


Figure 4.8: Relationship between the Medical Concepts

Person

This represents the person that interacts with the healthcare facility. It has all the associate demographics. Some information is linked from the master files[4](such as administrative gender and marital state). It also contains the information about the patient's insurance.

Episode of Care

The Episode of Care groups all the necessary information about one problem with the patient, from the detection of the problem to the final resolution. A person can have multiple Episodes of Care each one for each condition.

[4]These master files save the information that cannot be changed in a healthcare facility

Episode

The Episode saves the information that is affected to each process that exists inside each medical responsible department (MRD). An Episode groups all the encounter that happened consecutively inside a MRD.

Encounter

The encounter represents the smallest possible interaction between the patient and the healthcare facility.

4.2 Study of Architectural Solutions

To solve the proposed architectural problem two technologies were studied. First a solution based in Windows Workflow Foundation that would be adapted to allow communication between a presenter module. Then, a solution based on the Pageflow framework is goin to be presented.

4.2.1 Windows Workflow Foundation

A prototype was made that was supported by state machine running inside WF that defined the needed workflow. Then the state of the workflow was then transmitted through WCF to Microsoft Silverlight that showed the correspondent user interface.

The communication between the workflow and the WCF has to be custom developed. All the created workflow activities have to contain a communications module to report its state and get the required information from the WCF communication interface. Tests were made and the development of this part could be troublesome because all the communication has to be verified on a case by case basis to ensure the correct data types and values attributed to said variables.

The communication between the WCF and Microsoft Silverlight was successfully tested through http secure communication (*https*). This would allow the communications to be processed in a secure environment. Basic navigation commands would be asked by the user interface (sending the necessary query information) and the web service would reply with the id of the interface to be showed and the content that should be displayed in said interface.

This approach was tied by a lot of hard coding and didn't provide the necessary flexibility and scalability that the project needed so it was discarded.

4.2.2 Pageflow

The second solution that was studied involved a framework that was built on top of the WF framework.

The framework that supports the current solution is the Pageflow implementation on top of the Windows Workflow Foundation(WF) framework. The reasons for this selection of this framework are explaned here.

This framework enables a rapid prototyping that can use the existing framework as a base for the workflows that define the various operations that can exist within a hospital. By adopting this solution the ability to use persistence[5] and tracking[6] is greatly facilitated recurring to the built in support for these two features.

This solution has several advantages but few limitations. The following advantages are recognized to the Pageflow framework:

- Ability to create the workflows that represent an activity of the healthcare facility;
- The workflows that are created have similarities with state machine workflow;
- Each activity inside the workflow can be mapped to one user interface;
- Persistence allows to suspend and resume workflows at any given time[7];
- Possibility to attach to each activity a operation that it is done before the activity is executed and after it is executed[8];
- Possibility to define the expiration date on the workflow so that the suspended workflows can be erased after the specified time;
- Ability to use multiple user interfaces to execute the same workflow[9].

There are also a few limitations to this implementation[10]:

- The brokers included for the communication with ASP.NET and WPF only support one workflow definition limiting the usability of this solution[11];
- Complex definition of variables that support the workflow mechanics[12].

[5]Enables long running applications that can persist the entire workflow even if the operation is stopped

[6]Enables the monitoring of the workflow enabling to show to the user the information of the current running workflow and all the workflows in the historic of the application

[7]An admission can be suspended because a more important operation has to be done and then when the operation is finished the previous admission is still in persistence and can be resumed when it was suspended with no loss in the data

[8]This allows the creation of procedures that call external evaluations that can alter the data inside the workflow, or even notify external applications of specific events

[9]In the current pageflow framework there is support for WPF and ASP.NET

[10]A solutions to some of these limitations was developed

[11]A solution was developed to address this limitation

[12]These variables make the data available to the UI and allow persistence of the workflow

- Platform limitation[13]

Pageflow has the ability to create custom activities that support various operations that occur inside a healthcare facility, encapsulating these custom activities into a library. This library can then be provided to the medical analyst of the healthcare facility and the workflows can be custom defined by the analyst making a perfect fit for the needs of every healthcare facility.

To create, edit, publish the workflows a custom tool can be used to prevent the analyst for seeing unnecessary code. The activities[14] for a certain workflow can be dragged to a canvas, tied to each other to create the flow of information using a specially developed tool. This workflow could be published or saved according to the analyst's will.

4.2.2.1 Navigation Architecture

Now we are going to analyse a simple implementation of the Pageflow to better understand the Pageflow's functionality and powerful activity based construction.

In figure 4.9 at page 40 it is described a workflow with 3 activities that are running on WPF with 3 user interfaces. The numbers close to the captions represent the order of sequence that the workflow is executed. The description for each numbers is:

1 This represent the start of the workflow, the interface calls the desired workflow;

2, 10 and 18 The workflow activity reads the data that is needed from the context variables;

3, 11 and 19 The workflow activity executes some code that is predefined to execute before the execution of the activity;

4, 12 and 20 The workflow activity[15] sends the variable to the user interface;

5, 13 and 21 The user interface shows the necessary data and waits for user interaction;

6, 14 and 22 The user interface[15] sends the required data and requests the next workflow activity;

7, 15 and 23 The workflow activity executes some code the is predefined to execute upon data arrival;

8 and 16 The activity writes (updates) new data into the Context Variables;

9 and 17 The workflow activity[15] passes the control to the next workflow activity;

[13]This framework is tied to the Windows Workflow Foundation that only is available in the .Net framework for Windows

[14]These would include workflow activities and correspondent User Interface

[15]Performed by the Communication Broker, the WPF Broker in this example

24 and 25 Return the control to the workflow selection menu.

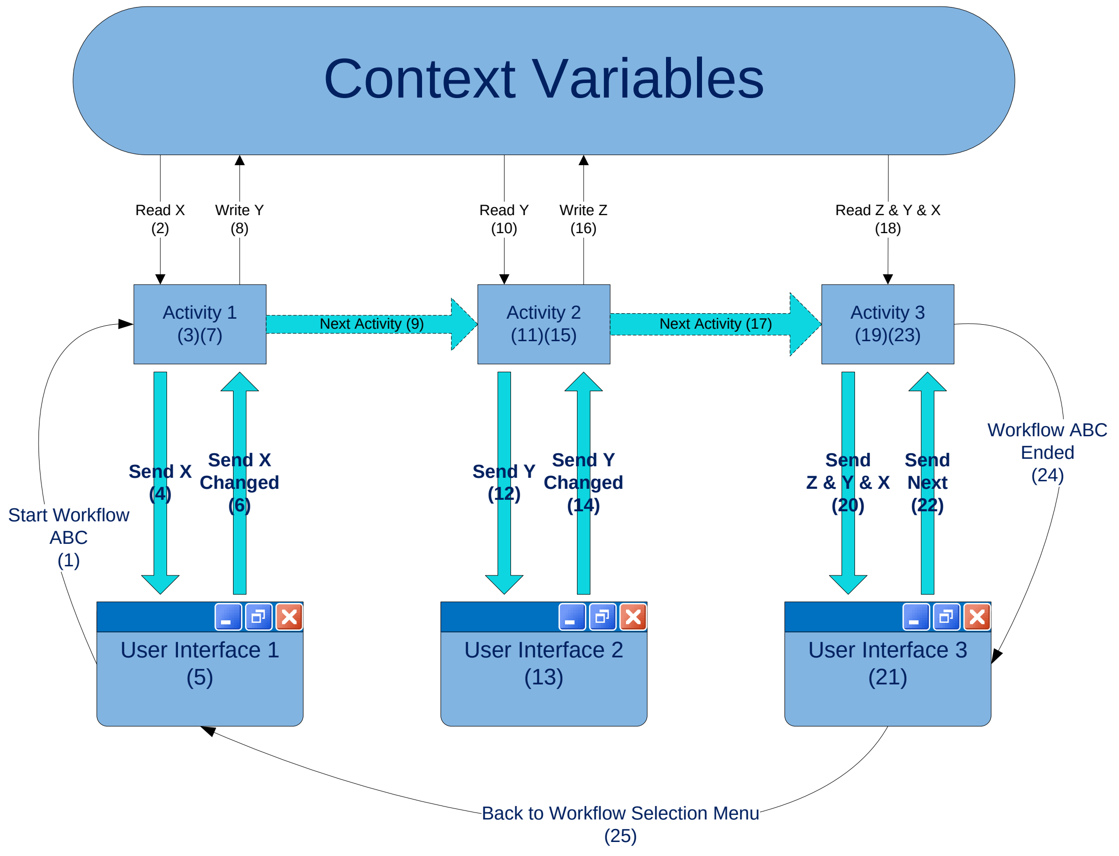


Figure 4.9: Interactions of a workflow with 3 activities

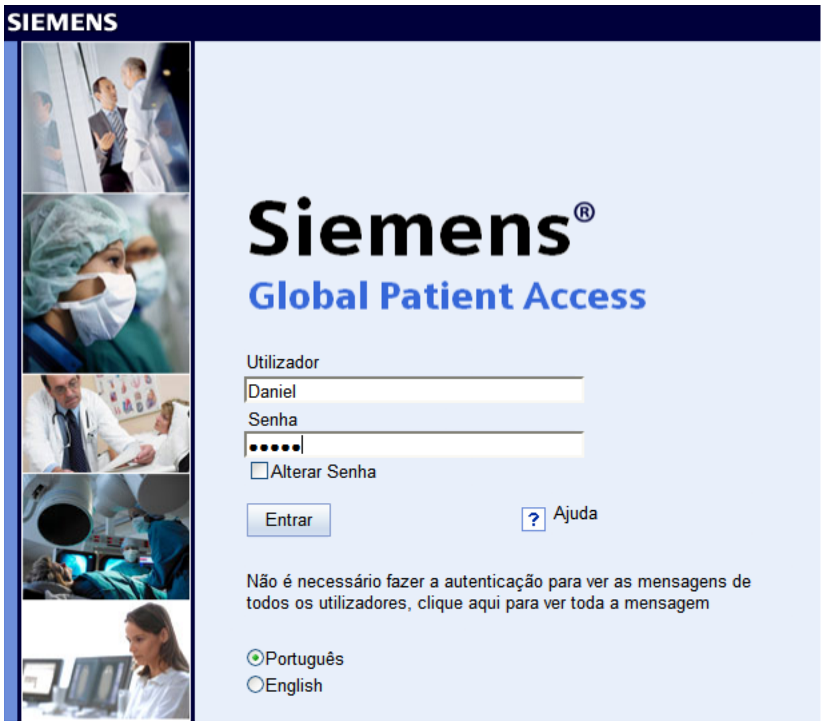
This architecture allows the normal navigation through the workflow activities but also allow to get back to the previous workflow if the user wants. It is allowed to go to the next and previous user interface screen. It also allows to suspend a running workflow and all the context variables are saved into the persistence store, making it available to restart the same workflow in the exact step it was suspended. A workflow can also be cancelled and returned to the menu erasing the current progress on the workflow.

4.3 Prototype

A prototype was developed as a feature showcase that describes the process of the Patient Check-in for appointment and the correspondent check-out. This prototype was developed before the beginning of the internship. This was the basis for the development of a workflow that supported a Patient Check-in for appointment and the corresponding check-out.

This prototype features full patient demographic edition, multiple reasons and types of appointments and healthcare insurance validator.

This prototype was used as a base for the development of all the Outpatient use cases. The figures 4.10, 4.11 and 4.12 show the user interface of this prototype.



Utilizador: Daniel

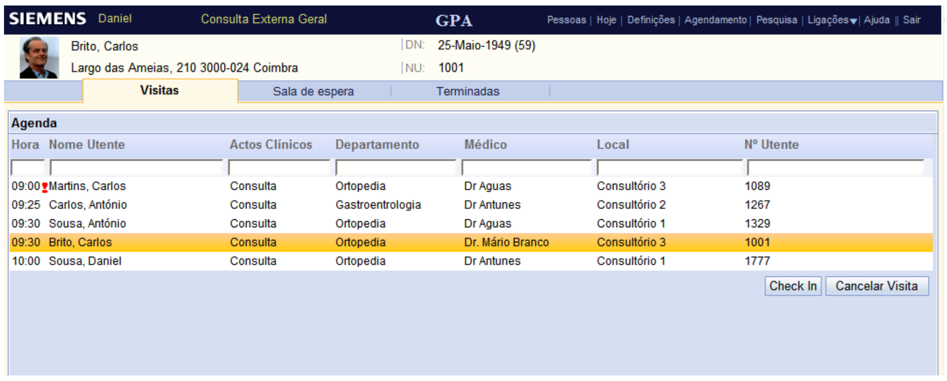
Senha: •••••

☐ Alterar Senha

Não é necessário fazer a autenticação para ver as mensagens de todos os utilizadores, clique aqui para ver toda a mensagem

[✓ Português]
English

Figure 4.10: Administrative clerk log in screen

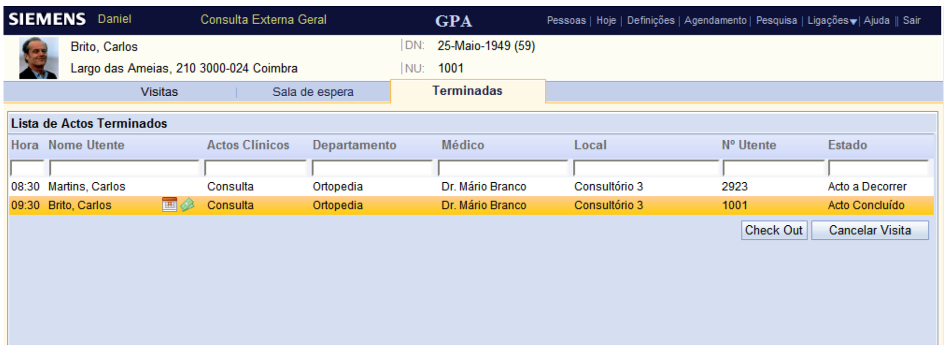


Brito, Carlos | DN: 25-Maio-1949 (59)
Largo das Ameias, 210 3000-024 Coimbra | NU: 1001

Agenda

Hora	Nome Utente	Actos Clínicos	Departamento	Médico	Local	Nº Utente
09:00	Martins, Carlos	Consulta	Ortopedia	Dr Aguas	Consultório 3	1089
09:25	Carlos, António	Consulta	Gastroentrologia	Dr Antunes	Consultório 2	1267
09:30	Sousa, António	Consulta	Ortopedia	Dr Aguas	Consultório 1	1329
09:30	Brito, Carlos	Consulta	Ortopedia	Dr. Mário Branco	Consultório 3	1001
10:00	Sousa, Daniel	Consulta	Ortopedia	Dr Antunes	Consultório 1	1777

Figure 4.11: Patient selection for check-in



Brito, Carlos
Largo das Ameias, 210 3000-024 Coimbra
DN: 25-Maio-1949 (59)
NU: 1001

Lista de Actos Terminados

Hora	Nome Utente	Actos Clínicos	Departamento	Médico	Local	Nº Utente	Estado
08:30	Martins, Carlos	Consulta	Ortopedia	Dr. Mário Branco	Consultório 3	2923	Acto a Decorrer
09:30	Brito, Carlos	Consulta	Ortopedia	Dr. Mário Branco	Consultório 3	1001	Acto Concluído

Figure 4.12: Patient selection for check-out

4.4 Conclusions

In this chapter the definition of each term that is going to be used in the software was deepened and the relations between all the terms clearly established. The functionalities of the base framework that is going to be used in the implementation were explained. The interaction between the activities inside a workflow was also explained.

These terms and their relation can clearly map all the occurrences inside a healthcare facility. The Pageflow framework enables fast prototyping and a great workflow management.

Chapter 5

Design and Implementation

This chapter presents the design and implementation process. Special attention is going to be given to complex tasks that are going to be shown.

It starts by presenting the design of all the modules involved in the development and the implementation details of each. It ends with the presentation of the development methodologies and the tools used in the project.

5.1 Design

Software design is a process that solves a problem by planing a solution for the problem. This is done when the purpose and specifications of the software are determined. The Design includes a low-level component and algorithm implementation issues as well as the architectural view.

5.1.1 He-ncounter Software

The He-ncounter was the name chosen for the software developed in the project Encounter Management. The He-ncounter has to be used in multiple clients running multiple workflows that are deployed inside a healthcare facility. Mostly, these workflows consist in a sequence of activities that combined with user input provide the necessary data to make the workflow proceed to the next state. Apart from the user interaction, some workflows activities involve a few operations that are called before said activity executes and after the input of the user has arrived. These pieces of code perform a variety of custom actions that range from making queries to a database (or saving data to a database) to sending HL7 messages to the applications that need the information.

This part of the software is the one that incorporates the running workflows, the user interface of all the workflow activities involved and the communication between the workflow and the user interface that is called WPF Broker.

There are many support systems around this primary functionality, such as the object-relational mapping used (NHibernate), the NHapi for the generation/parse of HL7 Messages. There is a Database that supports the whole business logic of the Episodes of Care, Episodes and Encounters, a Persistence Store to save the state of the workflow and also a Tracking Store that saves the information of all the running and completed workflows.

5.1.1.1 Architectural View

The architectural view shows the interaction between all the modules and frameworks. This view summarizes the implemented architecture of the He-ncounter Software.

The Architecture is showed in the figure 5.1 in the page 44 with the correspondent description.

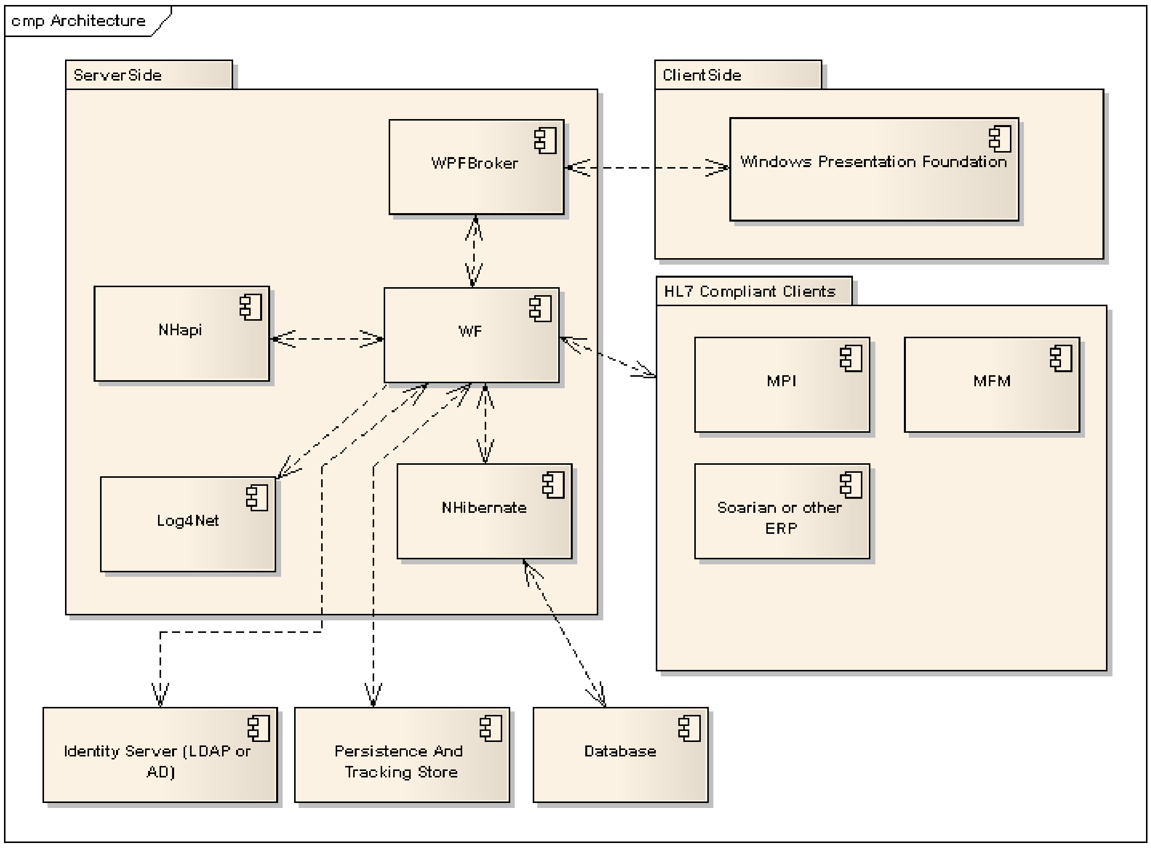


Figure 5.1: Architecture of the He-ncounter Implementation

As it can be seen in the above figure, the principal component of the architecture is the WF that runs the Pageflow framework on top of the Windows Workflow Foundation. This contains all the available workflows that are currently deployed in the WPF and registred in the Interface (created with Windows Presentation Foundation). The communications is established in both directions using the WPF Broker. When the user clicks in the interface to proceed, the next page is requested to the WF through the WPF Broker and the response goes from the WF to the interface through the WPF broker. This approach allows to multiple interfaces with multiple brokers maintain communication with the WF at the same time in a seamless way.

As said in previous chapters the WF allows the workflow to suspend and be resumed later. Whenever a workflow is in idle[1] all the its context is saved into a database that then supports the resume workflow function.

Some workflow activities use a post execution code that sends HL7 messages. These messages are constructed with the NHapi framework, it allows to create all the standard messages up to the version 2.5 of the HL7 standard. All these activities are logged using Log4Net and stored for later debugging.

5.1.1.2 Process View

The process view evaluates non-functional aspects such as performance, scalability and throughput. The main process system was identified in the section 4.2.2.1 in the figure 4.9 at page 40.

A few bottlenecks can be identified that can slowdown the entire system. Write and read operations are executed synchronously and that can lead to the slow down the entire system. These have to be kept to a minimum to improve the speed of the navigation in the workflow.

5.1.1.3 Physical View

This view shows the base hardware topology that supports the system execution. It shows the distribution of data and the communications between the various machines.

In the figure 5.2 at page 46 it is shown a deployment scenario.

5.1.1.4 Database View

This view shows the database that supports the entire concept of the chain Episode of Care, Episode and Encounter. The figure 5.3 at page 47 shows the designed database.

This design clearly maps the concepts of Epsiode of Care, Episode and Encounter that were shown in figure 4.8 and described at section 4.1.4.

The part of the database that is linked to the patient and not to the episodes of care allow to completely define the patient with the needed properties. These properties range from healthcare insurance, disabilities and chronic diseases (each person can have more than one) to administrative gender and martial status (each person has one).

This database was a simplified version, developed to be used in this prototype.

[1] In Windows Workflow Foundation a workflow being idle is the moment that the workflow is in a delay stage or waiting for a input.

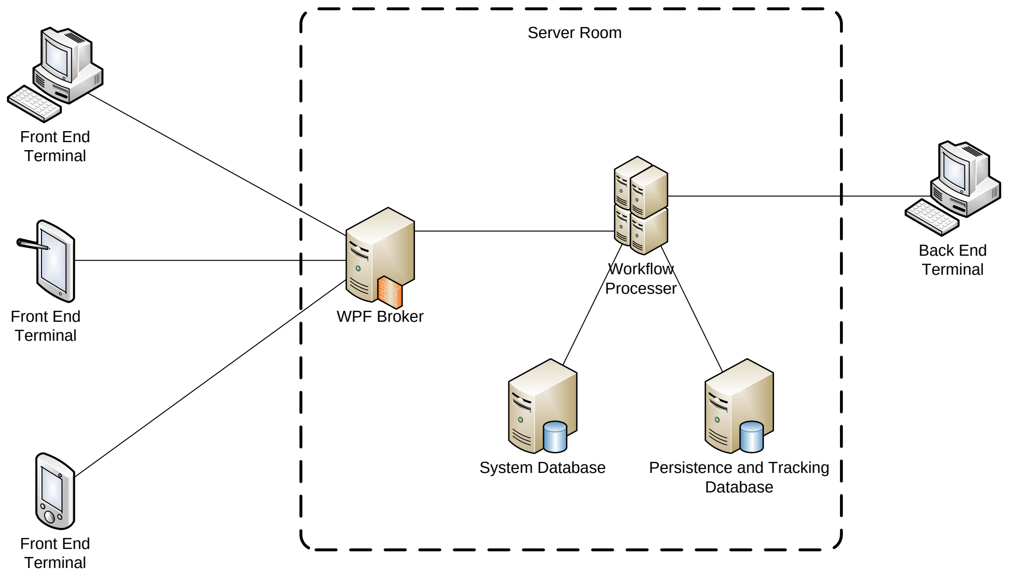


Figure 5.2: He-ncounter Deployment Scenario

5.1.2 Workflow Builder

This module allows the creation of workflows for healthcare facilities and the deployment of those workflows in the He-ncounter module. This module was described in section 2.6 and provides a complementary support module to be integrated with the He-ncounter module presented in the previous section.

5.1.3 HL7 Communication

The HL7 communication is a fundamental part in creating a truly integrable system. This communication is established using the HL7 standard. The messages that are when a trigger action occurs.

The HL7 Communications infrastructure is divided in two parts, message sending and message receiving.

Sending HL7 Messages

The sending of HL7 messages is triggered by worklflow activities that have HL7 messages in the end of their executing cycle. These usually inform other applications that have subscribed to such notification. An example is a communication of changes in patient demographics.

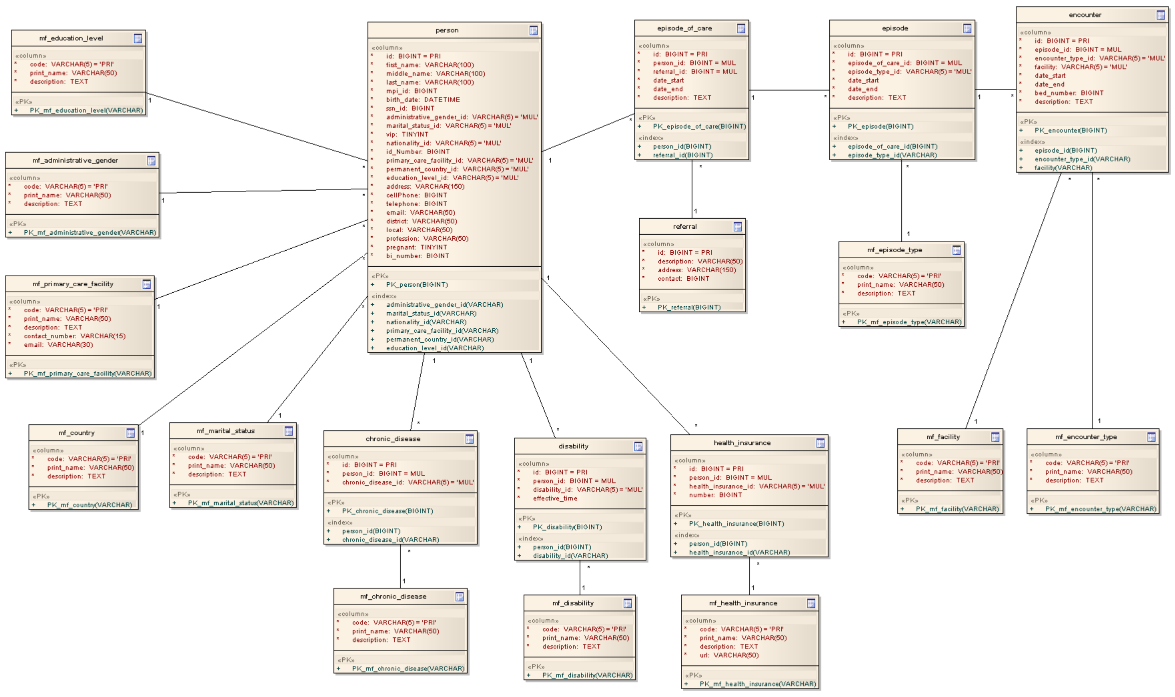


Figure 5.3: The Database View

Receiving HL7 Messages

To receive an HL7 message it is necessary to have an independent application that waits for the communication. This application is independent of the running workflows at any given the moment.

This application has a direct connection to the database. These messages typically request patient demographics data and schedule new encounters.

5.2 Implementation

The implementation phase of a software development is the realization of the application that was defined in the design phase. This section discusses in greater detail how the implementation actually was done. It is going to be described the main classes involved in the program and the respective features.

5.2.1 He-ncounter Software

The He-ncounter Software does all the management of the episodes inside a healthcare facility. This software relies on three layers in order to work.

It contains a set of workflows that can be defined to completely execute the needed operations inside a healthcare facility.

These workflows can be executed by a variety of user interfaces. In this program the user interface that executes the workflow is based on Windows Presentation Foundation (WPF). This interface has all the necessary pages that are going to be called by the workflow, making them completely independent from the workflow activity in question.

To make the connection between the workflow and the user interface a communication broker handles all the communication between the created workflows and the WPF based user interface.

5.2.1.1 Workflow Instantiation

Instantiating a workflow involves a variety of tasks. The workflow activities have to be connected, to each other to define the various tasks, then several variables have to be created to attach to each workflow activity. Triggers can also be attached to the beginning or the end of the workflow activity. Finally a configuration file has to be deployed to the interface program that maps the workflow activities to each user interface, this file is passed to the WPF Broker and the broker maps the next activity in the workflow to the user interface on the configuration file.

Workflow Design

The creation of a workflow involves dragging available activities and connecting them to define a workflow. An example of a workflow can be seen in figure 5.4 at page 48

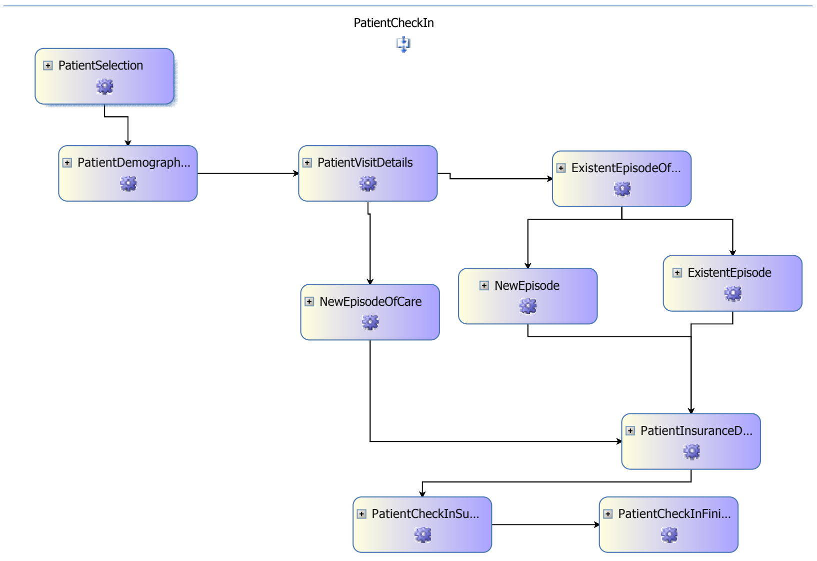
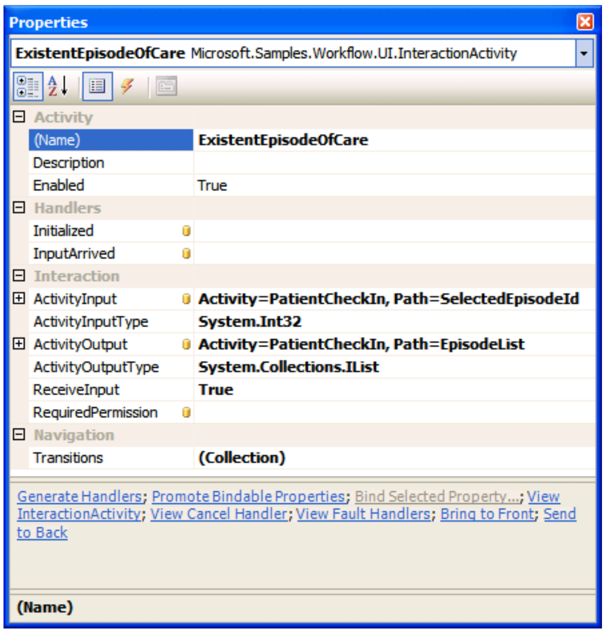


Figure 5.4: Example of a Workflow

Each of the squared areas is an activity that performs a given action. Each activity can be connected to other activities with a conditional rule that chooses an activity instead of the other. The start activity is selected so that the workflow knows where to begin. The designer generates code that stores all the connections, activities and the position of each object in the designer view of Microsoft Visual Studio.

Each activity has a set a properties that describes them. An example can be seen in figure 5.5 at page 49.


Properties

ExistentEpisodeOfCare Microsoft.Samples.Workflow.UI.InteractionActivity

Activity	
(Name)	ExistentEpisodeOfCare
Description	
Enabled	True
Handlers	
Initialized	
InputArrived	
Interaction	
ActivityInput	Activity=PatientCheckIn, Path=SelectedEpisodeId
ActivityInputType	System.Int32
ActivityOutput	Activity=PatientCheckIn, Path=EpisodeList
ActivityOutputType	System.Collections.IList
ReceiveInput	True
RequiredPermission	
Navigation	
Transitions	(Collection)

Generate Handlers; Promote Bindable Properties; Bind Selected Property...; View InteractionActivity; View Cancel Handler; View Fault Handlers; Bring to Front; Send to Back

(Name)

Figure 5.5: Example of Activity Properties

In the handlers option field (in figure 5.5), functions can be assigned to trigger in certain conditions. The handler executes the necessary code when the necessary conditions (triggers) occur. The trigger Initialized performs the function when the workflow activity is called, the trigger Input Arrived executes the function when the input from the user interface is sent back to the workflow activity.

In the interaction option it is defined what variables are sent to the user interface and where to store the data that is sent from the user interface. The Activity Output and the Activity Output Type set the variable and the type of the variable that is sent from the workflow, through the WPF Broker to the User Interface. This variable contains information that is generally displayed in the User Interface. The Activity Input and the Activity Input Type store the data that is sent from the User Interface to the workflow. The type of the data has to be known by the interface so that the correct data type can be sent to the workflow. The data is sent from the User Interface through the WPF Broker to the workflow activity.

Workflow Variables and Triggers

The variables that the workflow uses are defined in the code through a publishing procedure.

Listing 5.1: Example of a variable definition in the workflow

```
//Recieved from ExistentEpisodeOfCare
public static DependencyProperty SelectedEpisodeIdProperty
    = DependencyProperty.Register
("SelectedEpisodeId", typeof(int), typeof(PatientCheckIn));

/// <summary>
/// Method that sets and gets the
    SelectedEpisodeOfCareIdProperty
/// </summary>
[DesignerSerializationVisibility(
DesignerSerializationVisibility.Hidden)]
public int SelectedEpisodeId
{
  get
  {
    return ((int)(base.GetValue(
    PatientCheckIn.SelectedEpisodeIdProperty)));
  }
  set
  {
    base.SetValue(
    PatientCheckIn.SelectedEpisodeIdProperty, value);
  }
}
```

In the listing above it's described the variable that is used in the property Activity Input from the figure 5.5. First it is registered in the workflow a property with the chosen type. Then it is created function that has a set and get method that access and changes the variable in question. The variables that are defined this way can be chosen by the user interface and filtered by the types that exists in the current namespace.

The triggers that are executed in each one of the activities can perform simple notifying actions or more complex calculations in accordance with the parameters that exist in the current context. These can be set in the properties window showed in figure 5.5 or be defined in the initialization of the workflow.

Configuration file

The configuration file can be found in the client that is going to call the workflow. It is used to map the activities of the workflow to the various user interface pages. An example configuration file can be seen in listng 5.2

Listing 5.2: Example of the configuration File

```
<NavigationManagerSettings StartOnDemand="false">
  <Workflows>
    <add mode="Compiled" value="PageFlowTemplate.PatientCheckIn, PageFlowTemplate" name="PatientCheckIn"/>
    <add mode="Compiled" value="PageFlowTemplate.PatientCheckOut, PageFlowTemplate" name="PatientCheckOut"/>
  </Workflows>
  <Services>
    <add type="System.Workflow.Runtime.Hosting.DefaultWorkflowCommitWorkBatchService, System.Workflow.Runtime, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
    <add type="System.Workflow.Runtime.Hosting.SqlWorkflowPersistenceService, System.Workflow.Runtime, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" ConnectionString="Initial Catalog=PersistenceADT;Data Source=localhost\SQLEXPRESS;Integrated Security=SSPI;" UnloadOnIdle="true" LoadIntervalSeconds="120"/>
    <add
      type="System.Workflow.Runtime.Tracking.SqlTrackingService, System.Workflow.Runtime, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" ConnectionString="Initial Catalog=TrackingADT;Data Source=localhost\SQLEXPRESS;Integrated Security=SSPI;" IsTransactional="false" UseDefaultProfile="true" TrackXomlDocument="True"/>
  </Services>
</NavigationManagerSettings>
```

```
12 <WpfNavigationSettings>
13   <PageMappings>
14     <add bookmark="ExistentEpisodeOfCare"
          location="Patient␣Visit␣
          Details\ExistentEpisodeOfCare.xaml"
          workflowname="PatientCheckIn"/>
15     <add bookmark="SelectPatientCheckedIn"
          location="SelectPatientCheckedIn.xaml"
          workflowname="PatientCheckOut"/>
16   </PageMappings>
17   <ExceptionMappings>
18     <add type="Microsoft.Samples.Workflow.UI.␣
          WorkflowCanceledException"
          location="WorkflowSelectionWindow.xaml" />
19   <add type="Microsoft.Samples.Workflow.UI.␣
        WorkflowNotFoundExceptiom"
        location="WorkflowSelectionWindow.xaml"/>
20   </ExceptionMappings>
21 </WpfNavigationSettings>
```

This configuration file has two sections a Navigation Manager Settings (line 1) and a Wpf Navigation Settings (line 12). In the first section the are a few definitions that the broker uses to connect to the desired workflow. First a list of active workflows that are defined in another project, in this example the patient check in (line 3) and patient check out (line 4). Then there is a list of services that enable three options. First the workflow engine location (line 7), the SQL Persistence service (line 8) and the last one is the SQL Tracking service (line 9). In the Wpf Navigation Settings there is a Page Mappings section and a Exception Mappings section. In the first a bookmark is made that points at the workflow action's name, the location that points to a User Interface Page (a XAML file) and the name of the workflow, allowing multiple workflows per client. In the Exception Mappings section it is saved which user interface is showed when certain error occurs.

There are two configuration differences from the first file configuration that Microsoft proposed. The multiple workflow definitions in the Navigation Manager Settings and the field *workflowname* in the Page Mappings section. These changes were necessary to support multiple workflows per client. The implementation of these features is explained in section 5.2.1.3.

5.2.1.2 Windows Presentation Foundation based Interface

The interface of this applications was made using Windows Presentation Foundation. This technology allowed the rapid evolution of the interface because it clearly separates the designing of the application (built on top of XAML and fine tuned with Microsoft Expression Blend) and the application coding. This clearly made the task easier because of the complete separation between the visual layer and the code layer.

Workflow Calls

The navigation between the various user interfaces is made in a complete abstract way. The pages are not called directly but instead these calls are made to the WPF brooker that gets the request to the workflow and then returns the response of the workflow to the interface. Being so, the navigation is completely abstract.

The actions that can be performed are workflow management(start a workflow, cancel a running workflow and suspend a workflow) and navigation inside a workflow. Starting a workflow can initialize a new workflow and even continue an existing workflow. Bellow it's given an example that shows how to start a new workflow:

Listing 5.3: Starting a new Workflow

```
//Changes the static values of WorkflowApplication so that
//the action is the desired one (Start the workflow)
WorkflowApplication.CurrentUserInput =
  new UserInput(ActionKeys.StartAction, null, "PatientCheckIn");
//Executes the CurrentUserInput
this.NavigationService.Navigate("NavHub.xaml");
```

As can be seen in the listing 5.3 starting a workflow is a very simple task. First a static variable saves the desired operations that are going to be executed by the navigate function. ActionKeys is a sealed class that stores the options that can be executed inside a workflow. In this case to start a new workflow the action key is of type *StartAction*. The following field is null because it represents the activity input that is going to be passed to the workflow (in this case the workflow is starting and the first activity does not have any activity input property setted, as seen in section 5.2.1.1). The last field is the name of the workflow that is going to be called. The second line is the actual navigation between pages. The Navigation Service is provided to all user interfaces because they inherit from the System.Windows.Controls.Page. This navigation service is then overridden by the WPF broker. The value NavHub.xaml represents a control sequence that the overridden implementation checks in its function. Then, according to the input and the current constrains, the next step is returned and navigated to by the navigation service.

The system allows to resume any interrupted workflow[2]. The system previously queries the persistence database to acknowledge the currently running workflows and the tracking database to get the information of these running workflows such as the name of the workflow and the name of the current activity. To start a existing workflow the following command is executed.

Listing 5.4: Starting of an existing workflow

```
//Changes the static values of WorkflowApplication so that
//the action is the desired one (Start existing workflow)
WorkflowApplication.CurrentUserInput = new
    UserInput(ActionKeys.ContinueAction,
    PatientDemographics.getTextCode(workflowSelectionComboBox));
//Executes the CurrentUserInput
this.NavigationService.Navigate("NavHub.xaml");
```

As it can be seen the in the listing 5.4 the methods that are called are the same but with different parameters. The ActionKeys is now setted to *ContinueAction* so that the behaviour of the navigate method reflects this difference and the only other parameter that is passed is the GUID of the selected workflow. This GUID is stored in a combobox that is selected by the user (this combobox contains all the information of the running workflows).

When the workflows are created and are in execution, they can be cancelled if the user desires. The cancellation erases from the persistence database that running worklfow. To cancel a running workflow the following command is executed from within any user interface of that particular workflow.

Listing 5.5: Cancelling of a running workflow

```
//Changes the static values of WorkflowApplication so that
    the
//action is the desired one (Start the workflow)
WorkflowApplication.CurrentUserInput =
     new UserInput(ActionKeys.CancelAction, null);
//Executes the CurrentUserInput
this.NavigationService.Navigate("NavHub.xaml");
//Navigates to the start page
WorkflowSelectionWindow startWindow = new
    WorkflowSelectionWindow();
this.NavigationService.Navigate(startWindow);
```

[2]The interruption of the workflows can occur by the user intention (specific button in user interface) or recover the workflow from power loss,system failure and other causes.

The structure of the commands of the listing 5.5 is similar to the other commands with a few differences. The ActionKeys is now of type *CancelAction* and when the navigation service is activated it does not go to any page. A new start page is created and then the WPF navigates to it. This navigation is handled by the default navigate function after it passes the overridden function.

The running workflows can also be suspended. This suspension state provides the flexibility needed when the administrative clerk wants pause the current operation and resume it later. This suspend action is provided in a very straightforward way, since every time the workflow stops in a activity that is waiting for user input the workflow is persisted, then when the workflow is suspended executed the only thing that is done is the redirection to the start page. This can be seen in the following listing.

Listing 5.6: Suspending a running workflow

```
WorkflowSelectionWindow startWindow = new
    WorkflowSelectionWindow();
this.NavigationService.Navigate(startWindow);
```

As it can be seen by executing these two commands the workflow is already suspended and then the navigation goes to the start page.

The workflow navigation can also execute the true navigation commands such as Next and Previous commands. They allow the freely navigation of the user through the various pages that can be accessed. Every navigation saves the necessary data to the context of the workflow (so that it can be persisted) and navigates to the next or the previous page. To go to the next action following commands are executed.

Listing 5.7: Next action in a running workflow

```
//Changes the static values of WorkflowApplication so that
//the action is the desired one (Submitaction the workflow)
WorkflowApplication.CurrentUserInput =
     new
         UserInput(ActionKeys.SubmitAction, SelectedPatientID);
//Executes the CurrentUserInput
this.NavigationService.Navigate("NavHub.xaml");
```

The next action seen in listing 5.7 is structured in the same way as the above but in this case the ActionKeys is called *SubmitAction* and the next parameter passes the variables that are going to be used in the input activity of the workflow. This variable is passed through the WPF broker to the workflow. The navigation services handles all this communication and resolves what page needs to be displayed next. If a workflow does not need any activity input from the user interface, but it's still waiting for user input in order to proceed to the next workflow action. The parameter *null* can be passed if in a specific case where there is no input value required in that workflow activity.

The previous command is similar to the above but returns the page of the last workflow activity that was executed by the workflow. The previous command can be seen in the following code .

Listing 5.8: Previous action in a running workflow

```
//Changes the static values of WorkflowApplication so that
    the
//action is the desired one (Previous workflow activity)
WorkflowApplication.CurrentUserInput = new
    UserInput(ActionKeys.BackAction, null);
//Executes the CurrentUserInput
this.NavigationService.Navigate("NavHub.xaml");
```

The previous command is executed with the action keys setted to *BackAction* and executing the navigation service. This redirects the user to the last page that was displayed.

Workflow Activity Output

The activities that compose the worlflow receive their inputs from the user interface as shown in the previous section, using the submit action, sends the information for the workflow activity for processing. This links the input activity of the workflow activity to the content that is sent in the Next action. When the workflows are running in the client it is needed to pass information in the direction of the user interface. This is accomplished by linking the variable that is going to be passed to the user interface to the Activity Output property of the workflow activity. Then for each user interface page that needs this Activity Output the following listing is executed:

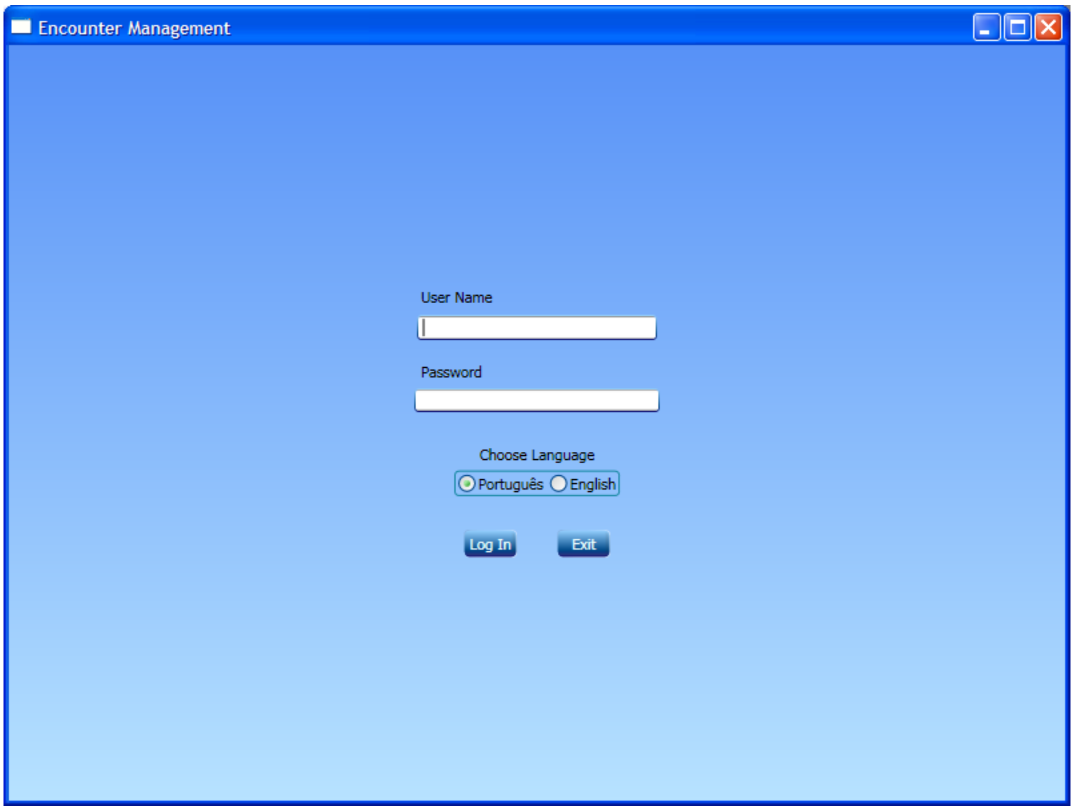
Listing 5.9: Getting a Activity Output from a Workflow Activity

```
Variable =
    WorkflowApplication.CurrentInteractionContext.ActivityOutput
    as VariableType;
```

As seen in listing 5.9 the variable is stored inside the WPF broker and is retrieved with a simple command. This clean concept allow the quick refactoring of the information that is passed from the workflow and through the communications broker.

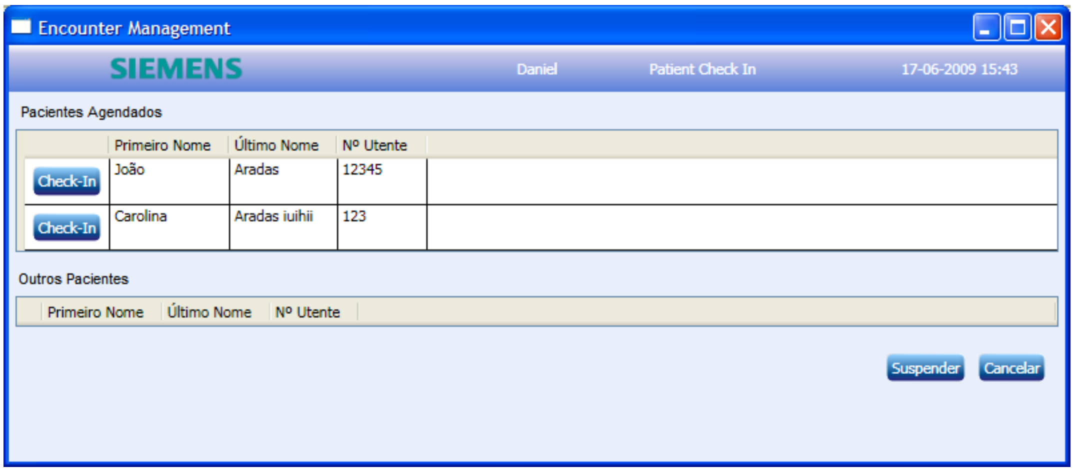
User Interface

The User interface is a very important phase in designing and implementing a program. The figures 5.6, 5.7, 5.8 and 5.9 show screenshots of the program running the developed workflow.



Choose Language
[✓ Português] English

Figure 5.6: Welcome Screen with Log-In and Language Selection



Daniel Patient Check In 17-06-2009 15:43

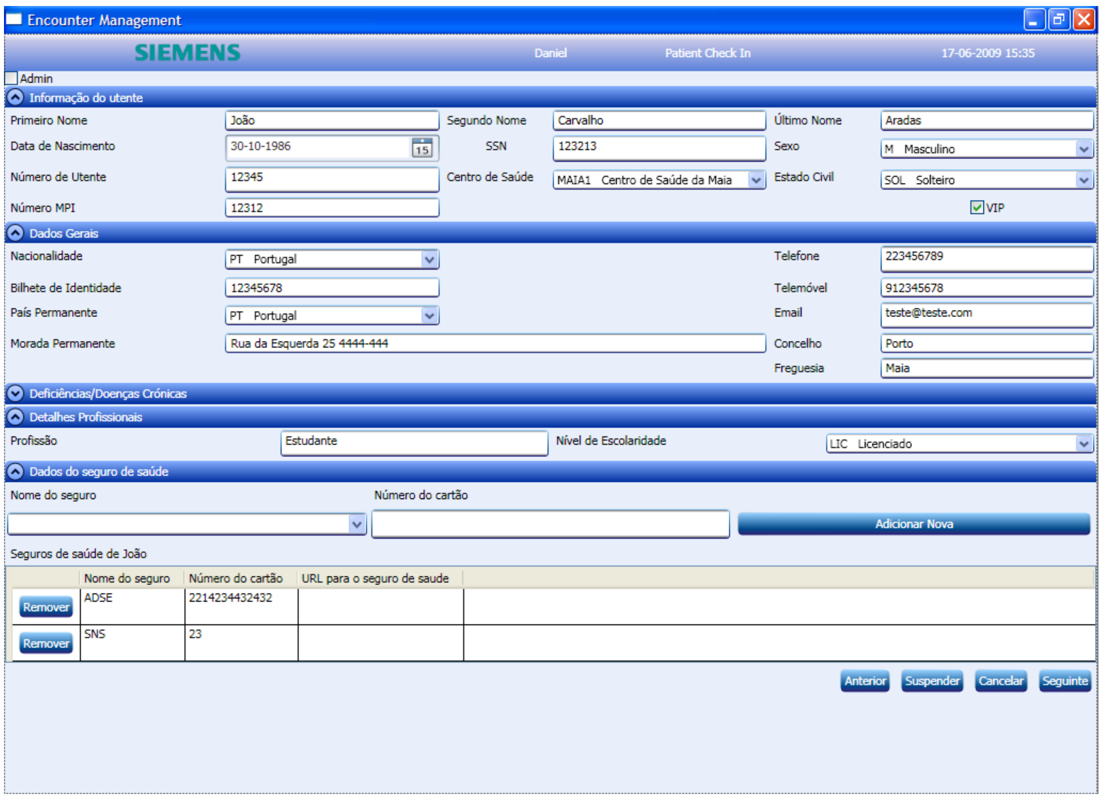
Pacientes Agendados

	Primeiro Nome	Último Nome	Nº Utente
Check-In	João	Aradas	12345
Check-In	Carolina	Aradas iuihii	123

Outros Pacientes

Primeiro Nome	Último Nome	Nº Utente

Figure 5.7: Patient Selection Screen with Scheduled Patients



Encounter Management

SIEMENS | Daniel | Patient Check In | 17-06-2009 15:35

☐ Admin

Informação do utente

Primeiro Nome	João	Segundo Nome	Carvalho	Último Nome	Aradas
Data de Nascimento	30-10-1986	SSN	123213	Sexo	M Masculino
Número de Utente	12345	Centro de Saúde	MAIA1 Centro de Saúde da Maia	Estado Civil	SOL Solteiro
Número MPI	12312				✓ VIP

Dados Gerais

Nacionalidade	PT Portugal	Telefone	223456789
Bilhete de Identidade	12345678	Telemóvel	912345678
País Permanente	PT Portugal	Email	teste@teste.com
Morada Permanente	Rua da Esquerda 25 4444-444	Concelho	Porto
		Freguesia	Maia

Deficiências/Doenças Crónicas

Detalhes Profissionais

Profissão	Estudante	Nível de Escolaridade	LIC Licenciado

Dados do seguro de saúde

Nome do seguro | Número do cartão | Adicionar Nova

Seguros de saúde de João

	Nome do seguro	Número do cartão	URL para o seguro de saude
Remover	ADSE	2214234432432	
Remover	SNS	23	

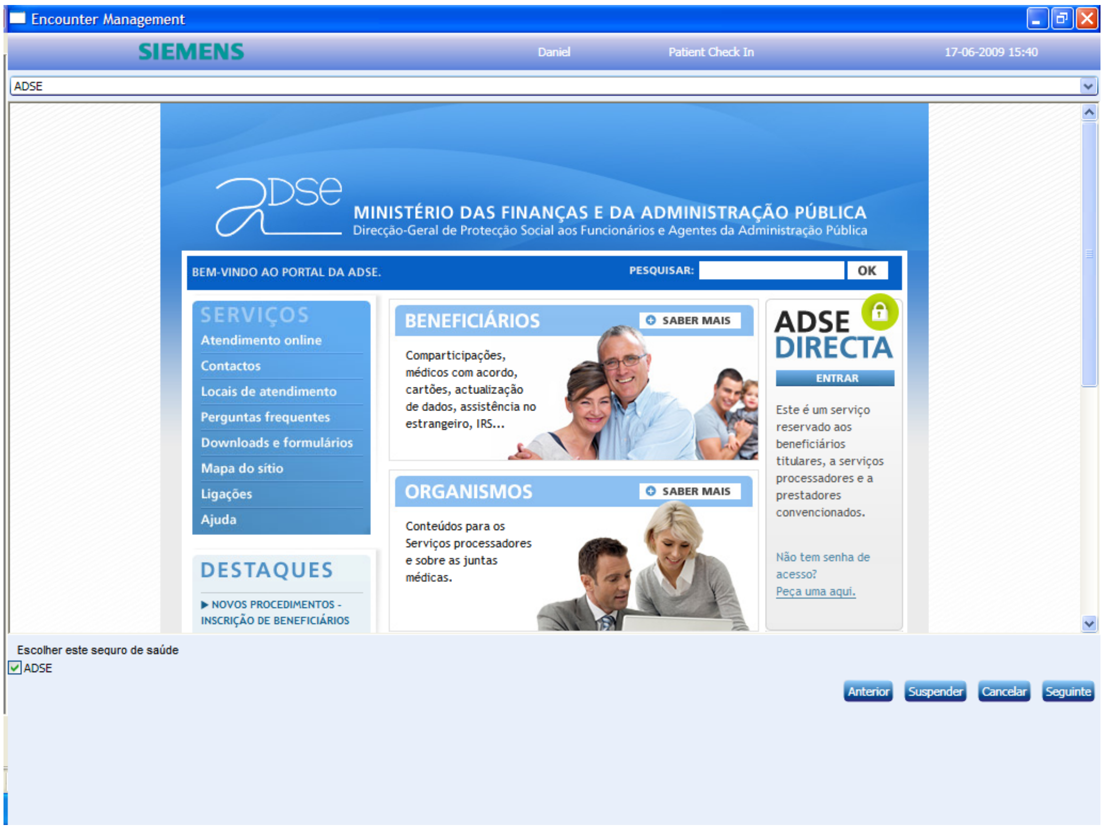
Anterior | Suspender | Cancelar | Seguinte

Figure 5.8: Patient Demographics Screen

Localization

Language localization is the process of translating a product into different languages or adapting a language to a specific country or region. This application was thought from start to support Portuguese (pt-PT[22]) and English (en-US[22]) languages. This process is supported by the .Net framework in a different number of ways. The simplest approach that provided the best results was to create a resource file and associate each resource file with the language code. Then the application has to have the UI Culture set to read the correct resource file.

The number of resource files created equals the number of languages that needs to be supported by the program. In this application because there are two target languages (English and Portuguese) two resource files were created, *Resources.resx* for the English version (it has no language code because it is the default language) and *Resources.pt-PT.resx* for the Portuguese version. These files are in *XML* and have three fields that need to be filed, the Name that holds the code that will be called in the code of the UI (has to be the same across all the resource files), the Value that holds the translation and the Comment for additional information of the translation. Here is an example of a localized object.



Direcção-Geral de Protecção Social aos Funcionários e Agentes da Administração Pública

Comparticipações, médicos com acordo, cartões, actualização de dados, assistência no estrangeiro, IRS...

Conteúdos para os Serviços processadores e sobre as juntas médicas.

Este é um serviço reservado aos beneficiários titulares, a serviços processadores e a prestadores convencionados.

Não tem senha de acesso? Peça uma aqui.

Escolher este seguro de saúde

✓ ADSE

Figure 5.9: Insurance Validation Screen

Listing 5.10: Example of localization usage in PatientDemographics.xaml

```
<Label Content="{x:Static␣
    properties:Resources.LabelPatientDemographicsName}" />
```

The correspondent English translation is:

Listing 5.11: Translation to English in Resources.resx

```
  <data name="LabelPatientDemographicsName"
      xml:space="preserve">
    <value>First Name</value>
  </data>
```

And the Portuguese:

Listing 5.12: Translation to Portuguese in Resources.pt-PT.resx

```
  <data name="LabelPatientDemographicsName"
      xml:space="preserve">
    <value>Primeiro Nome</value>
  </data>
```

All the translations are done in the two files (one for each language), the translation codes (in this case the *LabelPatientDemographicsName* is the code) are used in the program. The language can be changed in the Log-In screen. There it is asked to the user which language he wants. According the selection the according culture is set.

Listing 5.13: Code that sets the culture according to user selection

```
if ((bool)EnglishRadioButton.IsChecked)
  {
    WpfApplication1.Properties.Resources.Culture = new
       CultureInfo("en-US");
  }
  else
  {
    WpfApplication1.Properties.Resources.Culture = new
       CultureInfo("pt-PT");
  }
```

5.2.1.3 WPF Broker

The WPF Broker is in charge of communicate back and forth from the User Interface (UI) and the workflows that are instantiated by it. Using this module was very straightforward. The module was already developed by Microsoft to make the communication between the UI and the workflow. The developed solution clearly had a flaw, only one workflow type of workflow was supported per client. This communication problem existed in various levels. The configuration file wasn't able to cooperate with the definition of multiple workflows and the broker could not handle multiple workflow instantiation, it could only call one workflow, as many times as it was needed but not different workflows. To address this problem a full adaptation of the broker module was implemented and an adaptation of the Pageflow Source code was also used.

Multiple Workflows per Client

To create support for multiple workflows a few changes were made to the broker that reads the configuration file from the User Interface Module. The problem was that the broker was only looking for one workflow definition and the page mappings were not linked to any workflow in particular, they were only defined for the workflow that was defined. The changes were made in two steps, first acknowledge the existence of multiple workflows definitions. This was accomplished with the following changes in the WPF Broker.

Listing 5.14: Multiple Workflow Definitions

```
//WorkflowElement workflowElement = settings.Workflow;
WorkflowElement workflowElement = null;
foreach (WorkflowElement elm in settings.Workflows)
{
   if (elm.Name.Equals(input.WorkflowName))
   {
   workflowElement = elm;
   break;
   }
}
```

As in can be seen in listing 5.14 there was a single line that linked the workflow element to the existing one in the configuration section. The change from one workflow to multiple workflows replaced the single attribution to a cycle that finds all the existing definitions and discovers which one is being called whose name is stored in the variable *input.WorkflowName*.

There were also changes in the page mappings definitions because each page mapping has to be assigned to one workflow. This was accomplished by changing the configuration file reader in the WPF broker project.

Listing 5.15: Changes in Page Mappings in Workflow Name

```
private const string _workflowname = "workflowname";
[ConfigurationProperty(_workflowname, DefaultValue = null,
    IsRequired = true)]
public string WorkflowName
{
   get
   {
      return (string)base[_workflowname];
   }
   set
   {
      if (value == null)
      {
         throw new ArgumentNullException("value");
      }
      base[_workflowname] = value;
   }
}
```

In listing 5.15 the definition of the workflow name that was shown in page mapping property of listing 5.2 is implemented. In line 2 the Configuration Property attribute defines which options are going to be considered when reading from the XML configuration file.

The changes in the pageflow source code are of specific nature that do not apply to the scope of this thesis.

5.2.2 Workflow Builder

This module allows the creation of the workflows in an independent way and separated from the Microsoft Visual Studio 2008. This allows the application to be deployed with controlled costs in any client.

A medical analyst can then use this application to create new workflows deploy them and edit any current workflow that is saved and in production. This allows the creation and deployment of the workflows without any programmer intervention. The workflow activities that can be selected range from a variety of medical of health related activities (e.g. select patient's bed) to administrative tasks (e.g. get chargeable operations). Any custom activities can be developed by the analyst's request.

This module was not implemented due to time restrains.

5.2.3 HL7 Communication

This module allows the communication from the main application and many other existing applications through the international healthcare standard HL7 as analysed in section 2.1.1. These messages are integrated into workflow activities that perform specific actions and also as a standalone program that receives HL7 messages querying the information available in the database.

5.2.3.1 Sending Messages

The sent messages are integrated into special workflow activities that generate an HL7 message and send it to its destination (such as medical informations systems).

Sending Patient Demographics Changes

To send an HL7 message with the changes on the patient's demographics a specific HL7 message needs to be sent (the message to send patient demographics changes is called a ADT_A28). To generate this message a class was created that uses NHapi to generate the message with all the necessary headers and the required information in the necessary fields. The A28 class is a static class having only one static method that receives an Person as input and returns an ADT_A28 HL7 message type. The function uses all the data available from the Person to fill all the required information. The data is sent to the Master Patient Index(MPI). The MPI is a repository that references all the patients known by a medical department, clinic, hospital or a healthcare organization[23].

Listing 5.16: Creation of HL7 Message

```
PipeParser parser = new PipeParser();
byte[] byData = System.Text.Encoding.ASCII.GetBytes(
    parser.Encode(A28.notification(SubmitedData.person)));
connection.Send(byData);
```

The code in listing 5.16 shows the simple and clean usage that can be made by using the NHapi framework and a class that handles that specific message. The static class links the values in Person to the several fields in the message. Some examples are presented in the following listing.

Listing 5.17: Some values assignation

```
pid.GetPatientIdentifierList(0).IDNumber.Value =
    p.IdNumber.ToString();
pid.GetPatientIdentifierList(0).AssigningAuthority.NamespaceID.
    Value = "He-ncounter";
pid.GetPatientIdentifierList(1).IDNumber.Value =
    p.MpiId.ToString();
pid.GetPatientIdentifierList(1).AssigningAuthority.NamespaceID.
    Value = "MPI";
pid.GetPatientIdentifierList(2).IDNumber.Value =
    p.BiNumber.ToString();
pid.GetPatientIdentifierList(2).AssigningAuthority.NamespaceID.
    Value = "Estado␣Português";
pid.GetPatientIdentifierList(3).IDNumber.Value =
    p.SsnId.ToString();
pid.GetPatientIdentifierList(3).AssigningAuthority.NamespaceID.
    Value = "Segurança Social";
pid.GetPatientName(0).FamilyName.Surname.Value =
    p.LastName;
pid.GetPatientName(0).GivenName.Value = p.FirstName;
pid.GetPatientName(0).
    SecondAndFurtherGivenNamesOrInitialsThereof. Value =
    p.MiddleName;
```

A few values of the PID are assigned in listing 5.17.

The message that was generated can be seen in the following listing.

Listing 5.18: Example of an update demographics HL7 message

```
MSH|^~\&|He-ncounter|Development|MPI|Development|200906271611||ADT^A28^ADT_A05|16:11:30|D|2.5||||AL
EVN||200906271611
PID|||123^^^He-ncounter~123^^^MPI~123214^^^Estado Português~123^^^Seguraça Social||Aradas^João^Carvalho||198610180000|M|||&Sito^^Maia^Porto^^PT||223456789~912345678~^^^teste@teste.com|||SOL||||||||||PT|||||||200906271611|He-ncounter
PD1|||Centro de Saúde da Maia
PV1||O
```

This message updates the demographics of the Patient ”*João Aradas*”.
The listing 5.18 is described bellow:

- **MSH** - The Message Header segment contains the message type, in this case, ADTˆA28, which identifies the message type and the trigger event. The sender is the He-ncounter application. The receiving application is the Master MPI. The message was sent on 2009-06-27 at 16:11:30. The MSH segment is the initial segment of the message structure.
- **EVN** - The Event, i.e. Trigger Event segment contains the timing of the trigger event (2009-06-27 at 16:11:30)
- **PID** - The Patient ID segment contains the identifiers (3 identifiers, 1 person identifier and 2 patient identifiers) of patient João Patient as well as other demographics data (e.g. address, 2 telephone numbers and email, birthdate).
- **PD1** - The PD1 segment contains extra patient demographics such as the primary care facility.
- **PV1** - The Patient Visit segment (a mandatory segment in the ADTˆA28 message) contains information about an encounter between a healthcare provider and the patient. Given the use case no data is conveyed, the O denotes that the patient is an Outpatient.

5.2.3.2 Receiving Messages

The messaging receiving has to be workflow independent so that it can be active always to respond to the queries from other medical information systems and provide the response in an independent way from the workflow execution.

Synchronizing Master Files

The Master Files tables on the database are used to store static records that are relatively permanent. These files have to be set accordingly to be valid. To maintain a correct stored data the master files have to be synchronized by some authority that creates and manages these databases.

Between several medical applications there are shared information, usually found under the form of code or name lists (named catalogs), that should be centralized and globally available. These systems are, in general, composed by a central repository and by various entities that consult this information (the consumers). The service of the repository can be called a Master Files Management[24].

The operation of this module is independent of the rest of the program. It has a direct connection to the database and when notified changes the values in the master files. This change is transparent to the main program, when the next query happens the updated and new values will appear in the user interface.

5.2.4 Patient Check-in Workflow Implementation

One of the implemented workflows is the Patient Check-in. This was the chosen workflow to implement because it was already thought and developed in the prototype presented in the section 4.3. The mapping of this workflow was already shown in section 5.2.1.1 in figure 5.4.

First the patient needs to be selected from two available lists, one that holds the patients that were scheduled for the day in question and the other holds a list of all available patients so that any patient can receive treatment even if it is not scheduled. After the Patient Selection window, the selected patient demographics are shown and are available for all the necessary changes that needs to be done (the administrative clerk queries the patient to find possible errors in the demographics data. The next phase involves multiple screens to determine if it is necessary to create a new Episode Of Care or choose an existent one. After that all the necessary information is gathered for the creation of Episodes and Encounters. This is accomplished by using the workflow to follow decisions to the direction on which to follow based on the selections of the administrative clerk. After this step the insurance has to be validated using the website of the selected insurance. Finally the last step presents to the user all the changes that were made in the demographics information, the options that were selected in the Episode of Care and the selected Insurance. If the summary is accepted then the information is submitted to the workflow that saves the changes to the database and sends the necessary HL7 messages.

5.3 Development Methodologies and Tools

Choosing a development methodology implicates that one particular methodology has advantages that compensate its probable flaws in the current project. The choice of the development methodologies were made according to the decisions the project manager. The following methodologies were used.

Scrum

Scrum is an iterative incremental framework for managing complex work. Scrum is a set of process rules that set a group of best practices and predefined roles. There predefined roles in scrum are:

- **ScrumMaster** - Maintains the process;
- **Product Owner** - Represents the stakeholders;
- **Team** - A cross-functional group who do the actual analysis, design, implementation, testing, etc.

During a ”sprint”, typically two to four weeks period, the team creates a potentially shippable product increment. The set of features that go into a sprint come from the product ”backlog”, which is a prioritized set of high level requirements of work to be done. Which backlog items go into the sprint is determined during the sprint planning meeting. During this meeting, the Product Owner informs the team of the items in the product backlog that he wants completed. The team then determines how much of this they can commit to complete during the next sprint[1]. During a sprint, it is note allowed to change the items in the backlog, which means that the requirements are frozen for that sprint. After a sprint is completed, the team demonstrates the use of the software.

The Scrum methodology was adapted to fit the particular needs of this project and the time constrains.

Rational Unified Process

The Rational Unified Process[3] methodology is composed by many disciplines. Only one was applied in this project. The discipline was the Configuration and Change management discipline. This discipline is responsible for documents.

To create the necessary documentation this methodology provided streamlined documentation templates that allowed a correct documentation process.

Tools

To implement the system it was necessary to use a supported language of the used frameworks. The WF and WPF frameworks developed by Microsoft were available in two programming language, Visual Basic (VB) and C Sharp (C#). The chosen language was C# because of the similarities to C++ and Java. C# is easy to learn and a very powerful programming language.

To implement WF and WPF the Microsoft integrated development environment (IDE) was used, the Microsoft Visual Studio 2008 Professional Edition. This allowed the use of all the necessary frameworks in one IDE.

The WPF user interface was developed in two IDE, first in an earlier stage it was developed in Microsoft Visual Studio 2008 because it allowed a fast prototyping with a relatively advanced user interface. In a final stage, to tweak the user interface was used Expression Blend, a software also from Microsoft that is integrated in to Microsoft Visual Studio 2008 that focus on the design and the look and feel of the user interface. This allowed the user interface development to focus on the design of the application.

5.4 Conclusions

This chapter showed the design and implementation details on the developed solution.

It was defined that a workflow runs in the Windows Workflow Foundation engine and connects to a user interface (developed in WPF) through a broker that handles all the communications. The interface supports multiple workflows and pause and resume features.

It was shown the idea of the workflow builder application that creates and deploys workflows to be used in the He-ncounter software. This module was not developed due to time constrains.

The HL7 communication module was described and the integration with the He-ncounter software showed. This modules allows the interoperability of the He-ncounter software with other existent software.

Finally this chapter ended by presenting the development methodologies and the tools used in the project.

Chapter 6

Conclusions and Future Work

Throughout the duration of the project all the software phases were passed. Challenges arose during the development of this project that were identified and surpassed leading to the success of this project.

The selected frameworks that support the software proved to be a right choice. The Pageflow architecture was adapted to the specific needs of the project and the result was great. All the necessary workflows can be modelled and the necessary interfaces can be designed and associated with multiple workflow activities. Making this approach a valid one to enable fast development of the workflows that meets the needs of the clients.

The limitation of the implementation of the Pageflow architecture was successfully resolved and multiple workflows are now supported by the WPF client.

Now all the available workflows can be easily integrated into the existing software to further improve the offering of the Encounter Management.

6.1 Success Evaluation

To evaluate the success of the project the goals defined in section 1.4 have to be evaluated. As such these are the assessments that can be drawn.

1. Web page with authentication system - This objective was fulfilled in the early stages of the project and the required authentications were given to all the necessary entities;

2. Document the state of art on technologies and medical related standards - This report presented relatively complete overview of the different technologies, medical related standards, the implementations of these standards and the frameworks that can support the developed system;

3. Define an architecture that supports - the *Encounter* concept - The architecture was define in chapter 4;

4. Produce a prototype that supports the workflow philosophy - This prototype was successfully implemented and showed the potential of using the Pageflow framework.

5. Design and implement the Encounter Management - The design and implementation of the Encounter Management was done as showed in chapter 5. The implementation was focused on the Inpatient (the outpatient would be similar);

6. Use state of art technology when possible - The developed solution uses components recently released.

Even if not all the planned features were actually implemented, the successful achievement of goals 3,4 and 5 was already exposed in the previous chapters.

Form an overview perspective it is safe to say that while some activities were not carried out, the project succeeded overall. Some features and planned activities had to be dropped. The uncertainty of the type of work that had to be done led to planning more than could be executed.

6.2 Limitations

The persistence and tracking databased have to be in SQL Server. This is a requirement when using the out-of-the-box solution provided by the Windows Workflow Foundation.

The current solution only runs in localhost and was developed as a prof of concept.

6.3 Future Work

Regarding the continuation of this project, there were identified some features and activities that would benefit the application.

Despite designed and and extensively studied, the Workflow Builder module was not implemented, because it was not a high priority feature and the time available was already short. This would be the most interesting feature to implement because it ads an atractive feature that can make the product sell in the health market. This would also allow the implementation that is closest to the ideal architecture that was presented in section 2.6.

There were also another workflows that could have been developed (such as the Inpatient workflows) but for the same reason that was mentioned above it was not considered a priority, thus not being implemented. This can be accomplished with relative easy by creating workflows activities that are necessary and link them to create the necessary workflows and developed the correspondent user interfaces.

The user interface could be changed from WPF to Silverlight by making the necessary adoptions to the communications broker. This would allow the program to be easily deployed in the necessary environment and improve the compatibility to other operating systems. This would solve the problem mentioned above by making the program available to run in a distributed environment.

References

[1] Ken Schwaber. *Agile project management with Scrum*. Microsoft Press, illustrated edition, 2004.

[2] Linda Rising and Norman S. Janoff. The Scrum Software Development Process for Small Teams. *IEEE SOFTWARE*, July/August 2000.

[3] Jochen Krebs Ahmad Shuja. *RUP Reference and Certification Guide*. IBM Press, 2007.

[4] California HealthCare Foundation. California clinical data project: Setting standards. Available in http://www.chcf.org/documents/CCDPProjectOverview.pdf, November 2004.

[5] Health Level Seven. http://www.hl7.org/. Online; accessed June 1, 2008.

[6] Richard Mercille Hervé Verdel Michel Cotten Antoine Geissbühler Stéphane Spahni, Christian Lovis. Implementing a new ADT based on the HL7 version 3 RIM. *International Journal of Medical Informatics*, (76):190–194, 2007.

[7] Health Level Seven, Inc. *HL7 Messaging Standard Version 2.5.1*, 2007.

[8] Matthew MacDonald. *Pro Silverlight 2 in C# 2008*. Apress, 2008.

[9] Microsoft. Welcome to the Silverlight 3 Beta. http://silverlight.net/getstarted/silverlight3/default.aspx. Online; accessed June 5, 2008.

[10] Adobe. Adobe flash cs4. http://www.adobe.com/products/flash/. Online; accessed June 5, 2008.

[11] Adobe. Adobe flex. http://www.adobe.com/products/flex/. Online; accessed June 5, 2008.

[12] Microsoft. Windows presentation foundation. http://msdn.microsoft.com/en-us/library/ms754130.aspx. Online; accessed June 5, 2008.

[13] Chris Bowen Steve Resnick, Richard Crane. *Essential Windows Communication Foundation (WCF): For .NET Framework 3.5*. Addison-Wesley, February 2008.

[14] Bruce Bukovics. *Windows Workflow in .NET 3.5*. Apress, illustrated edition, 2008.

[15] Matt Winkler. Introducing the pageflow sample. http://blogs.msdn.com/mwinkle/archive/2007/06/07/introducing-the-pageflow-sample.aspx, June 2007. Online; accessed June 5, 2008.

[16] René Spronk. HL7 ADT Messages. Technical report, Ringholm GmbH, 08 2008.

[17] Integrating the Healthcare Enterprise. *IHE IT Infrastructure (ITI)*, 5 edition, December 2008.

[18] ACSS. *Serviços de Manutenção Correctiva e Evolutiva das Aplicações da ACSS*, 2009.

[19] S.A. ALERT Life Sciences Computing. Alert saúde na internet. http://portal.alert-online.com/, 2009. Online; accessed June 5, 2008.

[20] Rainer Anzböck and Schahram Dustdar. *Lecture Notes in Computer Science*, chapter Modeling Medical E-services, pages 49–65. Springer Berlin / Heidelberg, June 2004.

[21] Ian Sommerville. *Software engineering*. Pearson Education, 8 edition, 2007.

[22] ISO. Language code list. http://www.loc.gov/standards/iso639-2/php/code_list.php. Online; accessed June 5, 2008.

[23] Pedro Tiago Magalhães Gomes. Master patient index. Master's thesis, Faculdade de Ciências da Universidade do Porto, 2009.

[24] Nuno Miguel Pereira da Silva. Master files management. Master's thesis, Faculdade de Ciências da Universidade do Porto, 2009.

Appendix A

IHE Patient Encounter Management: Options of Triggers

Here is going to be presented the Subset of triggers and respective funcionalities that have been approved by IHE. All the subsets include the explanation and the description of the messages sent trough the HL7 International Health Standard, some include practical examples.

A.1 Basic Subset of Trigger Events

These are the basic subset of transaction of events and related massages. Our system is a Patient Encounter Supplier and Patient Encounter Consumer and in its most basic configuration has to support these 7 triggers and event messages.

A.1.1 A01 – Admit inpatient

The A01 trigger event is used only for Admitted patients only. These patients have to be assigned a bed to be admitted. It begins the patients stay in the healthcare facility. This information is entered in the primary Patient Administration System, and it is broadcasted to the nursing units and ancillary systems. For example and A01 can be used to notify the pharmacy that the patients has been admitted and may be legitimately prescribed drugs, it can be also sent to finance so that the billing period can begin, and it can be also sent to all the interested facilities that may be affected by the admit of a new person (dietary system, laboratory, pathology, radiology, etc).

The message sent is of the type ADT^A01^ADT_A01 and responded with ACK^A01^ACK.

A.1.2 A04 – Register outpatient

An A04 event trigger signals that the patient has arrived or checked in as a one-time, or recurring outpatient, and is not assigned to a bed. The most common use of the A04 is at the beginning of visit to the emergency room.

The message sent is of the type ADT^A04^ADT_A01 and responded with ACK^A04^ACK.

A.1.3 A11 – Cancel Admit inpatient/Register outpatient

For admitted patients, the event A11 is sent when the A01 is canceled. That can be for two reasons, the admit was erroneous or because of a decision to not admit the patient after all. For non-admitted patients the event A11 is sent when the A04 (register a outpatient) event is canceled, either because it was erroneous or because of a decision to not check the patient for the visit after all.

The message sent is of the type ADT^A11^ADT_A09 and responded with ACK^A11^ACK

A.1.4 A03 – Discharge patient/End visit

An A03 event trigger signals the end of the patient's visit to the healthcare facility. The patient's status has changed to discharged and the discharge date has been recorded. This event can be sent to notify all the systems that were advised of the A01 event such as the pharmacy, the nursing system, the finance system, etc. All these systems need the information so that they're sub-systems can handle the necessary changes now that the patient is no longer in the healthcare facility. This can also be used for non admitted patients to signal the end of a visit for a one-time or recurring outpatient who is not assigned to a bed. It can also be used to signal the end of a visit to the Emergency Room. If the patient's dies this event has to have the PID-29 (Patient Death Date and Time) and PID-30 (Patients Death Indicator) filled in.

The message sent is of the type ADT^A03^ADT_A03 and responded with ACK^A03^ACK.

A.1.5 A13 – Cancel Discharge patient/End visit

The A13 event trigger is sent when an A03 event is canceled, either because of erroneous entry of the A03 event or because of a decision not to discharge or end visit the patient after all. The field PV1-3 (Assigned patient location) should reflect the location of the patient after the cancellation has been processed. It can be different from the patient's location prior to erroneous discharge.

The message sent is of the type ADT^A13^ADT_A01 and responded with ACK^A13^ACK.

A.1.6 A08 – Update patient information

The A08 event trigger is sent when any patient information has changed but when no other trigger event has occurred.

The message sent if of the type ADT^A08^ADT_A01 and responded with ACK^A08^ACK.

A.1.7 A40 – Merge patient identifier list

A merge has been done at the patient identifier list level. That is, two PID-3 (Patient identifier list) identifiers have been merged into one. The A40 event is used to signal a merge of records for a patient that was incorrectly filed under two different identifiers.

The message sent if of the type ADT^A40^ADT_A39 and responded with ACK^A40^ACK.

A.2 Inpatient/Outpatient Encounter Management Option of Trigger Events

This option adds support for management for management of patient class and patient location. The following trigger events plus the basic subset of chapter A.1 are required to support the ”Inpatient/Outpatient Encounter Management”.

A.2.1 A05 – Pre-admit patient

An A05 Event is sent when the patient undergoes the pre-admission process. During this process, episode-related data is collected in preparation for a patient’s visit or stay in a healthcare facility.

This message is sent is of the type ADT^A05^ADT_A05 and responded with ACK^A05^ACK.

A.2.2 A38 – Cancel Pre-admit patient

The A38 event is sent when the pre-admit patient is cancelled, either because of erroneous entry of the A05 event or because of a decision not to pre-admit the patient after all.

This message is sent is of the type ADT^A38^ADT_A38 and responded with ACK^A38^ACK.

A.2.3 A06 – Change patient class to inpatient

An A06 event trigger is sent when a patient who was present for a non-admitted visit is being admitted after an evaluation of the seriousness of the patient’s condition. The status is change from non-admitted to admitted. The new patient location should appear in PV1-3 (Assigned patient location).

This message is sent is of the type ADT^A06^ADT_A06 and responded with ACK^A06^ACK.

A.2.4 A07 – Change patient class to outpatient

An A07 event trigger is sent when a patient who was admitted changes the status to “no longer admitted” but it is still being seen for this episode of care. This event changes a patient from an “admitted” to a “non-admitted”

This message is sent is of the type ADT^A07^ADT_A06 and responded with ACK^A07^ACK.

A.2.5 A02 – Transfer patient

An A02 event is issued as a result of the patient changing his or her assigned physical location. The A02 event can be used with admitted and non-admitted patients. If the patient is going to a temporary location (such as O/R, X-RAY, etc.) it is recommended that the A09 (Patient departing – tracking) and A10 (Patient arriving – tracking) events be used instead of A02.

This message is sent is of the type ADT^A02^ADT_A02 and responded with ACK^A02^ACK.

A.2.6 A12 – Cancel transfer patient

The A12 event is sent when the transfer patient is cancelled, either because of erroneous entry of the A02 event or because of a decision not to transfer the patient after all.

This message is sent is of the type ADT^A12^ADT_A12 and responded with ACK^A12^ACK.

A.3 Pending Event Management Option of Trigger Events

This option adds support for management of pending events. The following trigger events plus the basic subset of chapter A.1 and the Inpatient/Outpatient Encounter Management of chapter A.2 are required to support the "Pending Event Management".

A.3.1 A14 – Pending admit

An A14 Event trigger notifies the other systems of a planned admission. This happens when there is a reservation or when a patient admission is to occur imminently. It is similar to the pre-admit but without the implication that an account should be opened for the purposes of tests prior to admission.

This message is sent is of the type ADT^A14^ADT_A05 and responded with ACK^A14^ACK.

A.3.2 A27 – Cancel pending admit

The A27 Event trigger is sent when an A14 event is canceled, either because of erroneous entry of the A14 event or because of a decision not to admit the patient after all.

This message is sent is of the type ADT^A27^ADT_A21 and responded with ACK^A27^ACK.

A.3.3 A15 – Pending transfer

An A15 Event trigger notifies other systems of a planned transfer of a patient to a new location when the patient not yet left the old location. It can be used to warn services that the patient will move to the new location and their services should be redirect to the new location (sucks as meal preparation).

This message is sent is of the type ADT^A15^ADT_A15 and responded with ACK^A15^ACK.

A.3.4 A26 – Cancel pending transfer

The A26 Event trigger is sent when an A15 event is canceled, either because of erroneous entry of the A15 event or because of a decision not to transfer the patient after all.

This message is sent is of the type ADT^A26^ADT_A21 and responded with ACK^A26^ACK

A.3.5 A16 – Pending discharge

An A16 Event trigger notifies other systems of a plan to discharge a patient when a patient has not yet left the healthcare facility. It is used for advanced notification of a discharge in order to prepare for the patient's change in location. For example the pharmacy for the need to prescribe the discharge drugs, or the psychotherapy of the possible need for post-discharge appointments or other services.

This message is sent is of the type ADT^A16^ADT_A16 and responded with ACK^A16^ACK.

A.3.6 A25 – Cancel pending discharge

The A25 Event trigger is sent when an A16 event is canceled, either because of erroneous entry of the A16 event or because of a decision not to discharge the patient after all.

This message is sent is of the type ADT^A25^ADT_A21 and responded with ACK^A25^ACK.

A.4 Advanced Encounter Management Option of Trigger Events

This option provides support to manage changes of attending doctor, leaves of absence, and accounts. The following trigger events plus the basic subset of chapter A.1 are required to support the ”Advanced Encounter Management Option”.

A.4.1 A54 – Change attending doctor

An A54 Event trigger is issued as a result of change in the attending doctor responsible for treatment of a patient. The new attending doctor should appear in the PV1 – 7 – Attending Doctor. This could be used to notify the billing system that doctor’s fees should be billed to the new doctor starting from the timestamp in the message.

This message is sent is of the type ADT^A54^ADT_A54 and responded with ACK^A54^ACK.

A.4.2 A55 – Cancel change attending doctor

The A55 event is sent when an A54 event is canceled, either because of a decision not to change the attending doctor after all.

This message is sent is of the type ADT^A55^ADT_A52 and responded with ACK^A55^ACK.

A.4.3 A21 – Leave of absence

An A21 Event trigger is sent to notify systems that an admitted patient has left the healthcare facility temporarily. It is used for systems in with a bed is assigned to the patient, and it puts the current admitted patient activities on hold. For example the dietary services are informed that the patient is going home for the weekend.

This message is sent is of the type ADT^A21^ADT_A21 and responded with ACK^A21^ACK.

A.4.4 A52 – Cancel leave of absence

The A52 Event trigger is sent when an A21 event is cancelled, either because of erroneous entry of the A22 event or because of a decision not to put the patient on ”leave of absence” after all.

This message is sent is of the type ADT^A52^ADT_A52 and responded with ACK^A52^ACK.

A.4.5 A22 – Return from leave of absence

An A22 Event trigger is sent to notify systems that an admitted patient has returned to the healthcare facility after a temporary ”leave of absence” turning off all the ”hold” status in the various support services.

This message is sent is of the type ADT^A22^ADT_A21 and responded with ACK^A22^ACK.

A.4.6 3.1.4.6 A53 – Cancel return from leave of absence

The A53 Event trigger is sent when an A22 event is cancelled, either because of erroneous entry of the A22 event or because of a decision not to return the patient from de ”leave of absence” after all.

This message is sent is of the type ADT^A53^ADT_A52 and responded with ACK^A53^ACK.

A.4.7 A44 – Move account information

A move has been done at the account identifier level. That is, a PID – 18 – Patient account number associated with one PID – 3 – Patient Identifier list has been moved to another patient identifier list. An A44 Event trigger is used to signal a move of records.

This message is sent is of the type ADT^A44^ADT_A43 and responded with ACK^A44^ACK.

A.5 Temporary Patient Transfers Tracking Option of Trigger Events

This option tracks patient moves to and from temporary locations such as radiotherapy, scanner, EKG, and dialysis. The following trigger events plus the basic subset of chapter A.1 are required to support the ”Temporary Patient Transfers Tracking”.

A.5.1 A09 – Patient departing – Tracking

The A09 Event trigger is used when there is a change in a patient’s physical location (inpatient or outpatient) and when this is not a change in the official census bed location, as in the case of an outpatient setting. There are three situations that qualify as a non-census location changes: Patient Tracking (this could be used when the nursing application sends a transfer before the Patient Administration system issues an A02 Event trigger), the patient is in transit between locations from some time (the patients location between A09 and A10 if defined as in transit, similar to an A02 but with an interval in the middle) or a notification of a temporary location change (The patient is sent to a temporary location such as O/R, X-RAY, LIMBO, or HALLWAY. The patient may or may not return to the same assigned location after occupying the temporary location).

This message is sent is of the type ADT^A09^ADT_A09 and responded with ACK^A09^ACK.

A.5.2 A33 – Cancel patient departing – Tracking

The A33 Event trigger is sent when an A09 event is cancelled, either because of erroneous entry of the A09 event or because of a decision not to send the patient after all.

This message is sent is of the type ADT^A32^ADT_A21 and responded with ACK^A32^ACK.

A.5.3 A10 – Patient arriving – Tracking

The A10 Event trigger is used after the A09 event trigger, when the patient returns to the assigned bed.

This message is sent is of the type ADT^A10^ADT_A09 and responded with ACK^A10^ACK.

A.5.4 A32 – Cancel patient arriving – Tracking

The A32 Event trigger is sent when an A10 event is canceled, either because of erroneous entry of the A10 event or because of a decision not to receive the patient after all.

This message is sent is of the type ADT^A32^ADT_A21 and responded with ACK^A32^ACK.

A.6 Historic Movement Management of Trigger Events

This option adds capability to cancel or update safely any Movement. The movements update can be in the present, future or past. The movements cancel can only be present or future movements. These movements represent every transfer inside the hospital, temporary movements to temporary locations (such as mentioned in chapter A.5). To support this capability it is necessary to add the segment ZBE bellow PV1/PV2. This segment is used in the following trigger events: A01, A02, A03, A04, A05, A06, A07, A11, A12, A13, A14, A15, A16, A21, A22, A25, A26, A27, A38, A52, A53, A54, A55 and Z99.