Institute of Telecommunications. Technische Universität Wien.
Escuela Técnica Superior de Ingeniería Telemática. Universidad Politécnica de Cartagena.

BACHELOR DEGREE THESIS IN TELEMATICS ENGINEERING

UBIQUITOUS COMPUTING PLATFORM FOR ATTENDANCE REGISTERING AT UNIVERSITIES WITH NEAR FIELD COMMUNICATION

By

Alfonso Diego De Gea García

Professor: Markus Rupp
Supervisors: Robert Langwieser, María Victoria Bueno Delgado

Carried out at the Institute of Telecommunications, Technische Universität Wien.

Vienna, February 2011

Resumen.

Las directrices del Espacio Europeo de Educación Superior (EEES) están promoviendo la modernización de las universidades por toda Europa por medio de la implementación de nuevas tecnologías. Dicha modernización permitirá que los servicios que ofrecen las universidades estén más cerca de la sociedad.

Con esta idea en mente, la Universidad Politécnica de Cartagena se encuentra inmersa en la implementación del modelo llamado "Smart University", promoviendo un ambicioso proyecto sobre el estudio de la tecnología Near Field Communication (NFC) y su aplicación en el entorno universitario.

Este proyecto fin de carrera, estando integrado dentro del macro-proyecto descrito, ha desarrollado una plataforma hardware/software que pretende constituir las bases para investigaciones posteriores en la UPCT acerca del impacto de la tecnología NFC sobre el entorno universitario. Así como también ha definido un marco metodológico de aplicación a futuros proyectos en la universidad.

La plataforma resultante de este proyecto, basándose en la tecnología NFC, constituye una plataforma de computación ubicua para el registro de asistencia en universidades.

Los objetivos principales de la plataforma son:

- Servir como herramienta piloto de pruebas para planificar un posible despliegue a gran escala en las instalaciones universitarias.

- Constituir un marco metodológico, así como tecnológico, para su empleo en futuros proyectos de desarrollo que traten con NFC en la universidad.

En primer lugar, se ha llevado a cabo un estudio de la tecnología NFC, discutiendo diferentes aspectos como: características principales, estándares y especificaciones, comparativa con otras tecnologías de comunicaciones de corto alcance, escenarios de aplicación, y consideraciones sobre seguridad.

Desde el punto de vista del proceso de desarrollo llevado a cabo, la funcionalidad global alcanzada por la plataforma se puede concretar como: permitir la monitorización efectiva de la asistencia de los estudiantes a las clases teóricas y prácticas de laboratorio simplemente acercando un dispositivo móvil a un lector NFC colocado en la entrada principal de las dependencias.

La utilización de herramientas de código abierto (o de libre uso) ha sido un requerimiento no-funcional aplicado en el proceso de desarrollo del software. Este requerimiento ha servido para reducir los costes asociados tanto al proceso de desarrollo como a la plataforma resultante desarrollada.

La plataforma hardware/software resultante implica la consecución de diferentes resultados tecnológicos:

- Tecnología Hardware (modelos comerciales):
  - Lector NFC ACS-ACR122U.
  - Teléfono móvil Nokia 6212 Classic con tecnología NFC.
- Tecnología Software (código abierto o de libre uso):
  - Lenguaje de programación Java: Java 2 Standard Edition, Java 2 Micro Edition.
  - Application Programming Interfaces (APIs) y estudio sobre su aplicación: CLDC, MIDP.
  - Librerías Java: nfcip-java, regexp-me, log4j.
  - Integrated Development Environment (IDE): Eclipse.
  - Emulador de dispositivo móvil NFC: Nokia 6212 SDK.
  - Repositorio de código: VisualSVN.
- Herramientas software de servidor para modelar sistemas externos en la universidad (para pruebas de integración):
  - Servidor de Bases de Datos: MySQL.
  - Servidor de Directorio: OpenLDAP.

Este proyecto ha explorado dos posibles versiones de la plataforma:

- La versión "*Record Management System*" (RMS), que almacena datos privados en los registros internos del dispositivo móvil.

- La versión "*Secure Element*" (SE), que emplea la tarjeta segura interna del dispositivo (Java Card) como almacén de datos.

La versión desarrollada por completo ha sido la RMS, y la versión SE ha sido desarrollada parcialmente: desarrollo de un *applet Java Card*, diseño del protocolo de aplicación, lógica de tratamiento del *Push Registry*, e interacción entre el SE y el lector NFC. Sin embargo, la factibilidad completa de esta última versión descansa sobre la firma del código del *MIDlet* suite mediante un certificado de firma de código. El proceso de adquisición del certificado se enfrentó a algunos problemas administrativos por los que no fue posible, y, como resultado, el desarrollo de la versión SE no pudo llevarse a cabo por completo.

El proceso de desarrollo en este proyecto ha seguido un enfoque ingenieril, dividiendo las tareas de desarrollo en varias fases claramente distinguidas: Análisis, Diseño, Implementación y Pruebas. Se ha llevado a cabo la determinación y aplicación de la metodología de desarrollo más apropiada: Extreme Programming (XP). Esta metodología se encuadra dentro de las denominadas Metodologías Ágiles, siendo altamente indicada para proyectos con equipos de trabajo pequeños, requerimientos volátiles, y basados en nuevas tecnologías, de ahí que sea la metodología que ha encajado a la perfección para abordar el proceso de desarrollo en este proyecto. La herramienta utilizada para la elaboración de los modelos necesarios durante el análisis y diseño ha sido *Unified Modeling Language* (UML): diagramas de Casos de Uso, diagramas de Clases, diagramas de Despliegue.

Durante la fase de análisis, los requerimientos del sistema se han recogido utilizando la Metodología de Elicitación de Requisitos para Sistemas Software (REMSS) propuesta por Amador Durán. La definición de los requisitos funcionales ha implicado el diseño de dieciséis casos de uso de pruebas utilizando una técnica de pruebas basada en caja negra (*Black-Box Testing*, BBT). Este conjunto de casos de prueba diseñado se ha usado mediante la técnica *Test-Driven Development* (TDD, desarrollo dirigido por las pruebas) propuesta por XP, para llevar a cabo la implementación del código con el lenguaje de programación Java (Java-SE y Java-ME).

En la fase de implementación, los casos de uso de prueba diseñados han servido para comprobar la correcta ejecución de las funcionalidades, así como de su comportamiento esperado frente a condiciones de error. La ejecución de los casos de prueba ha arrojado un cien por cien de resultados de prueba exitosos. Por tanto, todas las funcionalidades han sido desarrolladas y comprobadas con éxito. Además, se ha tenido en cuenta la aplicación de técnicas de Usabilidad en el desarrollo de la interfaz, diseñando la experiencia de usuario por medio de la aplicación del Modelo de usabilidad propuesto por Francisco Montero.

El desarrollo de este proyecto ha implicado la consecución de otra tarea de forma implícita pero particularmente relevante: la aplicación de los conocimientos teóricos y prácticos adquiridos durante los estudios de Ingeniería Telemática.

Proyectos venideros que también implementarán tecnología NFC en la UPCT utilizarán los resultados alcanzados en este proyecto:

- Control de acceso a dependencias de la universidad (oficinas, clases, laboratorios).

- Gestión de préstamos de recursos didácticos (libros, software, proyectores, medios informáticos).

- Pago de tasas administrativas (emisión de certificados, tasas de matrícula).

El conjunto de tecnologías empleadas constituye un marco de trabajo base para esos futuros proyectos, estableciendo la configuración básica hardware así como las herramientas software requeridas para abordar los procesos de desarrollo.

Las metodologías y técnicas de desarrollo utilizadas en este proyecto son:

- La metodología ágil Extreme Programming.

- Metodología para la Elicitación de Requisitos de Sistemas Software propuesta por Amador Durán.

- Test-Driven Development.

- Black-Box Testing.

- Modelo de usabilidad propuesto por Francisco Montero.

Su determinación y aplicación en este proyecto conforman pautas para su aplicación en trabajos futuros con características similares: individuales, basados en nuevas tecnologías, requisitos cambiantes, que incluyan un proceso de desarrollo de software. Por tanto, los resultados metodológicos alcanzados en este proyecto constituyen un marco de trabajo para este tipo de proyectos futuros.

Además, la plataforma desarrollada servirá como una herramienta para pruebas piloto para la planificación de un posible despliegue a gran escala en instalaciones de la universidad. Como trabajo futuro, durante la fase de pruebas de puesta en marcha, diferentes parámetros serán estudiados como el tiempo de respuesta, fiabilidad de la

tecnología, ganancia en términos de tiempo y monetaria, e impacto en los miembros de la universidad.

Respecto a la seguridad, el estándar NFC proporciona algunas características ideadas para prevenir algunos ataques, como *Man-In-The-Middle*, y *sniffing* de datos. Sin embargo, la seguridad debe ser mejorada empleando criptografía dedicada a establecer un canal seguro entre la comunicación de los dispositivos. Este canal seguro jugará un papel muy relevante especialmente en futuros desarrollos relacionados a la venta de entradas o a la banca móvil, ya que estas aplicaciones tratan con datos de usuario extremadamente sensibles.

Como resumen, este proyecto es el primero de una serie de trabajos de investigación y desarrollo en la UPCT relacionados con la tecnología NFC, constituyendo un primer paso para alcanzar la implementación satisfactoria del modelo "Smart Universitiy".

Abstract.

The universities all over Europe are deeply involved in significant changes to adapt themselves to current European directives in educational matters. The European Higher Education Area, as the main objective of the Bologna Process, was meant to create more comparable, compatible and coherent systems of higher education in Europe, providing guidelines for these purposes, which involve the modernization of the universities through the full implementation of new technologies in all their areas.

The Technical University of Cartagena has promoted an ambitious project on the study of Near Field Communication technology and its application at the university environment. This thesis, being a part of this project, develops a hardware/software platform intended to constitute a basis for further scientific research at the Technical University of Cartagena dealing with the impact on the university society.

The platform resulting from this thesis, using as basis Near Field Communication, constitutes a ubiquitous computing platform for attendance registering at universities. The main objectives of this platform are: to serve as a pilot testing tool intended for the planning of a possible large-scale deployment at university facilities, and to constitute a methodological and technological base framework for future development processes dealing with Near Field Communication at the university.

From the development process viewpoint, the global functionality achieved by the platform can be stated as follows: allowing the effective monitoring of student attendance at lectures or laboratory practices just by bringing a mobile device close to a Near Field Communication reader placed at the main entrance of the dependencies.

During the testing and commissioning phase, several parameters will be studied, such as response time, technology performance, gains in terms of time and money saving, and impact on the university members.

Upcoming projects at the university will be able to use the set of technologies selected to address the development process in this thesis, such as: the application of Extreme Programming development methodology, the application of a usability model, the Java programming resources (Application Programming Interfaces, libraries, development environment, device emulators), the commercial hardware devices used (mobile phone and reader), and the server software tools to develop external systems at the university (database and directory server).

To my family

Acknowledgments.

First of all and foremost, I have to be deeply grateful to my parents for the early bet that they did on me, which has allowed me to achieve this milestone in my life: the conclusion of the Bachelor Degree in Telematics Engineering.

I acknowledge the valuable help, the confidence shown and the ever-present guidance provided by my thesis supervisors: Robert Langwieser and Maria Victoria Bueno Delgado (in the near and in the far field). Thanks also to Professor Markus Rupp.

I appreciate the helpful pieces of advice about Usability provided by Francisco Montero, as well as the guidance about Extreme Programming implementation issues, by José Eduardo Córcoles. Thanks so much!

Thanks to all my university mates for their support and company, especially to Amalia Lorca and to the "Tiger-centred" fellowship.

I am glad to count on such good friends and co-workers who encouraged me when I was a bit lost. Special thanks to the "Little Moments" gang, to Zollo, and to Estrella de Levante. Thanks also to Pronaia and her thousand stories that gently prodded me into finding a foreign "opportunity".

Last but not least, I would like to expressly emphasize my acknowledgement to Natalia Arias. Thanks a lot for your inexhaustible help!

Vienna, February 2011.

Table of Contents.

List of Figures.

List of Tables.

Chapter 1. INTRODUCTION.

"What would life be if we had no courage *to attempt anything?"*

Vincent Van Gogh (1853-1890)
Dutch painter

This chapter is intended to serve as an introduction for the thesis. Its background, motivation and objectives are presented here, focusing on the development requirements within the environment in which the thesis has been carried out.

1.1. BACKGROUND.

The universities all over Europe are deeply involved in significant changes to adapt themselves to current European directives in educational matters. The European Higher Education Area (EHEA) [WEHEA], as the main objective of the Bologna Process, was meant to create more comparable, compatible and coherent systems of higher education in Europe.

Besides the obligatory curricula adaptations, the compliance with the EHEA guidelines implies rebuilding a "university of quality" by establishing drastic changes of the teaching models at the universities, which involves the introduction of new management and organization measures, control and quality assurance of education, and modernization of the universities through the full implementation of new technologies in all their areas, that is, the implementation of a brand-new model.

Therefore, the study of new technologies and their correct implementation are mandatory and challenging tasks to let the European universities become "smart". This new university model, the so-called "Smart University", brings university services closer to society, enhances their quality and promotes students and teachers' mobility. It goes without saying that this changes are going to directly impact on the whole university society: teachers, students, university managers, and administrative staff.

Ubiquitous computing (also known as pervasive computing or "ambient intelligence") is a computational model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities and spread all over the environment. Thus, in a ubiquitous computing system, relationship between people, practice, and technology happen as part of the "natural" or "touching"

interaction paradigm [BD2006]. In the touching paradigm the interaction in the environment is realized by bringing the mobile device into contact or very close to a smart object [JTSM2007].

Near Field Communication (NFC) is an interesting emerging technology which is becoming a possible candidate to fit the requirements of pervasive systems projects. NFC combines a wireless proximity communication technology with mobile phones. This technology allows the users to simply and easily interact with the system, fulfilling the "touching" paradigm. In other words, users are able to interact with the smart computing elements of their surroundings by simply "touching" them with their NFC-enabled devices, like mobile phones. Thereby, NFC turns these devices into more valuable tools by providing them with additional applications.

NFC technology, as discussed in Chapter 2, is based on Radio Frequency (RF) Identification (RFID) technology, and allows data communication over a distance up to 20 cm. The major advantage of NFC over other wireless communication technologies is its simplicity: transactions are initialised automatically, simply by touching an NFC element, like a reader, another NFC-enabled device or an NFC compliant transponder (see Chapter 2, Near Field Communication Technology).

Fortunately, the massive incorporation of NFC on mobile devices and the availability of them to final users is a matter of a short time, since Nokia announced that all the new Nokia smartphones from 2011 would come with NFC technology [WMF10].

1.2. MOTIVATION.

In anticipation to the global NFC introduction into the market, the Technical University of Cartagena (*Universidad Politécnica de Cartagena*, UPCT) has promoted an ambitious project on the study of this technology and its application at the UPCT facilities by means of several development sub-projects [BPDG2011**]**.

The project, led by the Telematics Engineering Group (*Grupo de Investigación Telemática*, GIT) of the UPCT, is aimed at implementing the "Smart University" model at the UPCT to meet the guidelines promoted by the EHEA (see Section 1.1).

A study on the impact of NFC technology on society is being performed. This research sub-project is aimed at finding out the impact of this technology in Spain: degree of penetration and acceptance by users of mobile communications, hardware/software requirements, economic cost, concerns about security and privacy, and technological impact [BPDG2011**]**.

Other sub-projects for the implementation of NFC technology at the UPCT facilities are being carried out: access control to UPCT dependencies (offices, classrooms, and laboratories), loan of didactic material, payment of administrative fees, and registering attendance at lectures and laboratory practices [BPDG2011**]**.

- **Access control to UPCT dependencies with NFC technology.** This system will allow the university staff to open the doors of the dependencies to which they have granted access, by simply touching with their mobile phones an NFC reader on the door. The reader, connected to the university network, will handle the queries to the database that contains all the information about users, dependencies and permissions granted. The authorization will be obtained when the mobile user has access to that dependency. If so, the door will open using a relay circuit.

- **Management of loans of didactic resources at UPCT libraries with NFC technology.** This system will allow all the university members to perform the loan of didactic resources (books, software, and magazines) in two simple steps: *i)* a user, touching an NFC tag placed on a book, captures the book data, and then *ii)* the user touches an NFC reader of the library as a final step to perform the loan. The reader will be connected to a database that stores all the information about loans, users, and expiration dates of loans.

- **Payment of administrative fees at the UPCT with NFC technology**. The purpose of this project is to allow students to perform administrative payments immediately, just by touching an NFC reader placed in the secretariat concerned. There are several kinds of administrative fees, such as matriculation charges, transcripts of records issuing, certified documents, or extra-curricular activities, to name but a few. This application requires the reader to have direct access to the database of the secretariat to determine the price of the fee to pay. The user, just by bringing the phone close to the reader, automatically sends the total amount to pay to the bank. At this point, the reader acts as an intermediary in the communication between the user and the bank. Finally, the bank transaction is committed.

- **Attendance registering at laboratory practices with NFC technology**. The objectives of this project, as the actual work of this thesis, are discussed in Section 1.3.

The main motivation of this thesis, more than to obtain a software product as it is, is to constitute a basis for further scientific research dealing with the impact on the university society: acceptance, benefits, money and time saving, and technology reliability. What is more, the resulting work will serve as a methodological and technological base framework for future development projects which deal with NFC technology.

Summing it up, the thesis work, ideated and supported by the GIT, constitutes one more step to achieve the satisfactory implementation of the "Smart University" model at the UPCT.

1.3. OBJECTIVES.

The hardware/software platform resulting from this thesis will serve as a pilot testing tool intended for the planning of a possible large-scale deployment at the UPCT. During the testing and commissioning phase, several parameters will be studied such as response time, technology performance, gains in terms of time and money saving, and impact on the university members. Obviously, this testing and commissioning phase will be part of future work, out of the scope of this thesis.

The pioneer work of the current thesis, since it carries out the first of a series of projects related to NFC technology in a developing approach at the UPCT, is intended to serve as a methodological and technological base framework for future projects which deal with NFC technology. Above all, the issues related to development methodology application, programming tools, software libraries, and programmatic approaches used or produced as a result of the current thesis, will serve as basis of those projects dealing with the development of NFC applications at the UPCT.

Having said this, the two main objectives of the platform can be stated as follows:

- *To constitute a platform for pilot testing intended to the planning of a possible large-scale deployment at the UPCT.*
- *To serve as a methodological and technological base framework for future projects which deal with NFC technology.*

However, despite the fact that the main objectives of the thesis have already been stated, from the development process viewpoint, and using as basis NFC technology, the global functionality to be achieved by the hardware/software platform can be stated as follows:

- *"The platform will allow the effective monitoring of student attendance at lectures and laboratory practices just by bringing a mobile device close to an NFC reader placed at the main entrance of the dependencies".*

When working with mobile devices, several considerations have to be taken into account: *i)* the efficiency, since mobile phones are resource-constrained devices, *ii)* the

definition of the most appropriate functionalities to be offered to these devices, and *iii)* the manner of presenting and accessing to the information from these devices: the User Interface (UI).

These considerations raise a significant non-functional restriction when programming a mobile device, due to the reduced space on the screen as well as the low performance of this kind of devices: The UI has to be accessible, with screens that provide a set of concise information and menus that provide access to the whole functionality. This highlights the need to reformulate the user interaction with the computer platform and the contemplation of Usability during the development process.

The usability model applied as a non-functional requirement in the development of this thesis is the proposal by Francisco Montero [MF2005], which advocates mixing international standards with ergonomic criteria. The main characteristics of this proposal are:

- Based on internationals standards like International Organization for Standardization (ISO) 9126 Standard.
- It is an open proposal, in which other usability criteria can be added and linked to the existing ones.
- It uses ergonomic criteria.
- It allows the Usability criteria to be put into practice.

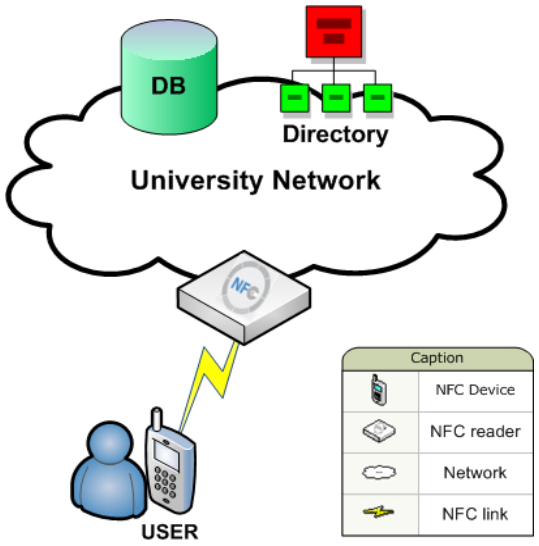


Figure 1.1: First approach to the system architecture.

Figure 1.1 depicts a first approach to the system architecture of the platform. The actors and elements involved are: *i)* a user with an NFC-enabled mobile device; *ii)* an NFC reader which interacts with the user; *iii)* a database containing information on subjects, practices scheduling, teachers and students; and *iv)* a directory server to host the users' credentials.

The resulting hardware/software platform of this thesis will constitute a tool for the monitoring of student attendance at lectures and laboratory practices which correspond to subjects they are enrolled in, by using NFC devices. Thus, in a real environment, the user establishes a communication with the NFC reader (by using a UI that fits well to mobile devices). This communication launches a query to the directory server, which validates the user. Finally, the user's attendance is recorded in the database.

Several key points are obtained from the architectural scheme outlined in Figure 1.1. These remarks have to be considered and treated as tasks within the methodology used in the development process:

- Detailed study of NFC technology.
- Use of the most appropriate development methodology.
- Selection of the set of hardware and software tools necessary to address the development: hardware devices, emulators (if needed), programming tools, software libraries, and device drivers.
- Design of the UI of the mobile devices, according to the usability model.
- And finally, assessment and analysis of work produced.

Furthermore, the development of these tasks involves the completion of another task implicitly but particularly relevant within the scope of this thesis: the proper application of those theoretical and practical concepts acquired during the studies of Telematics Engineering.

1.4. PRIOR CONSIDERATIONS.

As stated in previous sections, the main motivation of this thesis lies in the idea of improving the services offered by the UPCT through the application of technological advances promoted by the EHEA [WEHEA]. The development process attempts to integrate the knowledge acquired during the author's studies with the practical experience of a university environment.

The thesis addresses some of the issues associated with information management at a university environment, specifically, at the Technical University of Cartagena.

Universities are an integral part of the current society model, the Information Society. They produce and consume increasingly amounts of digital information. In consequence, they have to constantly increase their capacity to manage this information. Moreover, universities often do not have sufficient funding, so that the most feasible and suitable strategy to address the implementation of new technological solutions has to take into account the use of non-proprietary software that does not involve the expenditure of large sums of money in concept of user licenses. In other words, open-source based tools must be preferably used. Besides, the adopted strategy must avoid the utilization of technologies of low acceptance by the community or which involves a disproportionate learning curve.

Thus, during the execution of this thesis, the adopted strategy for the software development has been to use open-source tools, meeting this way the non-functional requirement of minimizing economic costs.

The use of Agile Methodologies perfectly fits to this project due to several factors: *i)* the novelty of the NFC technology usage and the new programming issues related to mobile phones, *ii)* the presence of unstable software requirements, *iii)* the reduced development staff, and *iv)* the tight delivery deadline.

The most popular methodology nowadays within this group is Extreme Programming (XP), which constitutes an agile methodology focussed on promoting interpersonal relations as the key to success in software development. XP is especially aimed at those projects with small development teams, with short deadlines, volatile requirements, and/or based on new technologies.

That is why XP arises as the most suitable methodology to address the development process of this thesis, and, in consequence, it is the selected methodology for that purpose.

In this sense, within the framework of the development project, the thesis supervisors play the role of customers that determine the requirements of the platform to be developed. The author, for his side, plays the actual roles of analyst/developer/tester responsible for translating the models needed to reach a valid solution which meets the established requirements.

1.5. CONTENT STRUCTURE.

The thesis is structured in the following chapters:

The current chapter, named ***Introduction***, outlines the background of the thesis; its motivation, focusing on the development needs within the environment in which it has been carried out; as well as the objectives pursued with it.

In Chapter 2, ***Near Field Communication Technology***, a description of NFC technology is presented. Several aspects are discussed, from its historical evolution and its main characteristics and standards, to its specifications and operational properties. Finally, the chapter ends comparing NFC to other short-range communication technologies.

Chapter 3, ***Methodology and System Analysis***, details the methodology used in the project development and performs the analysis of the requirements that the developed platform has to fulfil.

In accordance with the Software Engineering process, Chapter 4, ***System Design and Implementation***, discusses the detailed design of the system. This chapter defines the design of the UI, the software architecture, and how the functionality offered by the system is accessed.

Finally, in Chapter 5, ***Conclusions and Further Works***, the conclusions from the results achieved in the thesis are detailed, as well as future works that can be addressed in relation to these results.

Chapter 2. NEAR FIELD COMMUNICATION TECHNOLOGY.

"Freedom in general may be defined as the absence of obstacles to the realization of desires"

Bertrand Russell (1872-1970)
English logician and philosopher

This chapter presents a description of NFC. Several aspects are discussed, such as: historical evolution and main characteristics, standards and specifications, comparison with other short-range communication technologies, application scenarios, and security issues.

2.1. INTRODUCTION.

NFC is an emerging technology which is becoming a possible candidate to fit the requirements of pervasive system projects, combining a wireless proximity communication technology with mobile phones.

NFC technology allows phone users to simply and easily interact with the system, fulfilling the "touching" paradigm [BD2006]: NFC users are able to interact with the smart computing elements of their surroundings by simply "touching" them with their NFC-enabled devices, like mobile phones.

Once into proximity, NFC devices can set up a Peer-to-Peer (P2P) connection and exchange configuration and authentication data. The devices could engage in transactions using any of the compatible protocols or set up a connection using faster and longer range protocols like Bluetooth or Wireless Ethernet (Wi-Fi) [WNFCF]. Thereby, NFC turns mobile devices into more valuable tools by providing them with additional applications (see Section 2.9).

NFC is both a "read" and "write" technology based on RFID [BVEG2009]. NFC constitutes a short-range contactless smart card technology which operates in the 13.56 MHz frequency band, allowing data communication over a distance up to 20 cm and data rates up to 424 kbps [BPDG2011].

NFC technology is backward compatible with current standards for contactless communication and it supports two protocols on its own, NFC Interface and Protocol (NFCIP) versions 1 and 2 (NFCIP-1 and NFCIP-2 respectively), described in Section 2.3.

A built-in NFC chip can operate both as a contactless card and as a contactless reader, making the standard very suitable for device identification and communication initialization. Because the transmission range is so short, NFC-enabled transactions are inherently secure. Also, physical proximity of the device to the reader gives users the reassurance of being in control of the process [WNFCF].

NFC can be used with a variety of devices, from mobile phones that enable payment or transfer information to digital cameras that send their photos to a TV set with just a touch. The possibilities are enormous, and NFC is intended to take the complexities out of today's increasingly sophisticated consumer devices and make them simpler to use [WNFCF].

2.2. HISTORICAL DEVELOPMENT.

The inception of NFC technology in 2002 was originally motivated by a joint venture between Royal Philips Electronics, Nokia Corporation, and Sony Corporation to develop an open standard technology which made connectivity between close coupled devices easier.

NFCIP was designed and submitted for its adoption as a standard by the European Computer Manufacturers Association (ECMA), being approved in 2003 under the names NFCIP-1 and NFCIP-2. Then, the ISO and the European Telecommunications Standards Institute (ETSI) also approved both of these standards.

In 2004, the joint venture created the NFC Forum to promote the NFC technology implementation and standardization, thus ensuring interoperability between devices and services. As stated in [WNFCF], the goals of the NFC Forum are to:

- Develop standards-based NFC specifications that define a modular architecture and interoperability parameters for NFC devices and protocols.
- Encourage the development of products using NFC Forum specifications.
- Work to ensure that products claiming NFC capabilities comply with NFC Forum specifications.
- Educate consumers and enterprises globally about NFC.

In June 2006, only eighteen months after its founding, the Forum formally outlined the architecture for NFC technology. As of December 2010, the Forum has released fifteen specifications. These specifications provide a "road map" that enables all interested parties to create new consumer-driven products. In February 2011, the NFC Forum has one hundred and forty members [WNFCF].

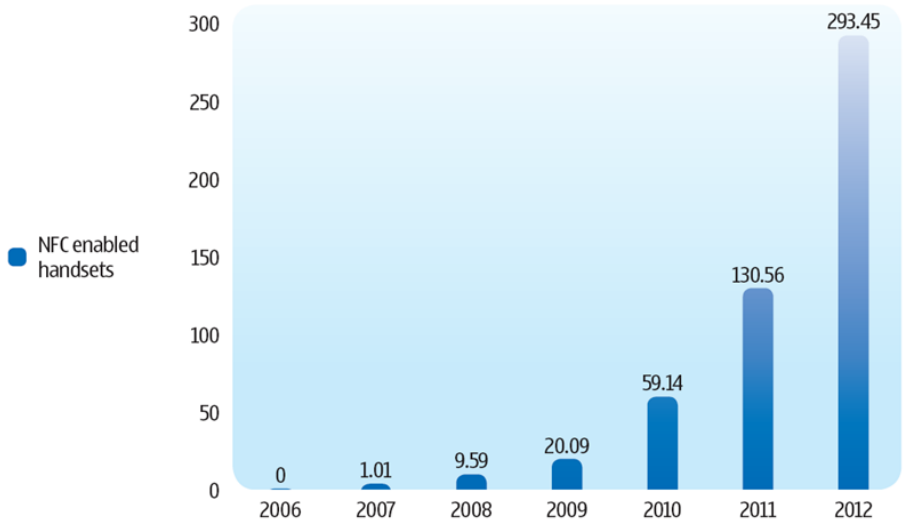


Figure 2.1: NFC-enabled handset shipments (millions), from [WABIR].

ABI Research estimates that "NFC-enabled handsets will approach 300 million shipments in 2012" [WABIR] (see Figure 2.1).

According to the report [WI2010] and the ABI Research estimates, nowadays, the number of operational mobile devices in the world is about $5 \cdot 10^9$; hence, the NFC-enabled devices represent only the 1.18 per cent of this quantity. [WI2010] also estimates that the number of operational mobile devices in 2012 will be about $6 \cdot 10^9$. In consequence, almost a 5 per cent of them will be NFC-enabled, an amount more than interesting to mobile operators to bet on NFC technology in the coming years [WTMOV].

The massive incorporation of NFC on mobile devices and the availability of them to final users is a matter of a short time, since Nokia announced that all new Nokia smartphones from 2011 would come with NFC technology [WMF10].

2.3. STANDARDS.

Currently there are various standards and specifications for NFC, defined by the ISO/IEC [WISO], ETSI [WETSI] and ECMA:

- NFCIP-1 is specified by ECMA-340 Standard [ECMA340] (ISO/IEC 18092, ETSI TS 102 190).

- NFCIP-2 is specified by ECMA-352 Standard [ECMA352] (ISO/IEC 21481, ETSI TS 102 312).

- ECMA-356 Standard defines the NFCIP-1 RF interface test methods [ECMA356] (ISO/IEC 22536, ETSI TS 102 346).

- ECMA-362 Standard defines the NFCIP-1 protocol test methods (ISO/IEC DIS 23917, ETSI TS 102 394).

NFCIP-1 specifies the interface and protocol for simple wireless communication between NFC devices, with data rates of 106, 212, and 424 kbps.

As stated in [ECMA352], "The ECMA-340, ISO/IEC 14443 and ISO/IEC 15693 standards specify the RF signal interface, the initialisation, the anti-collision mechanism, and the protocols for the wireless interconnection of closely coupled devices and access to contactless integrated circuit cards operating at 13.56 MHz".

[ECMA352] (NFCIP-2) specifies: "the communication mode selection mechanism, designed to not disturb any ongoing communication at 13.56 MHz", for devices implementing: *i)* ECMA-340: NFCIP-1; *ii)* ISO/IEC 14443: Proximity Coupling Device (PDC); or *iii)* ISO/IEC 15693: Vicinity Coupling Device (VCD). The specification of NFC, PCD, and VCD communication modes are outside the scope of NFCIP-2 Standard. Devices implementing these communication modes have to meet their respective standards.

In addition to the standards, the NFC Forum performs several specifications (see Figure 2.2):

- Logical Link Protocol (LLCP): NFCIP-1 data exchange protocol (P2P).

- Record Type Definition (RTD) and NFC Data Exchange Format (NDEF): data formats for communication with external cards.

- Card emulation mode: NFC-enabled devices internal smart card, using the Secure Element (SE).

GlobalPlatform has proposed a specification of a multi-application architecture of the Secure Element [GPSEMM].

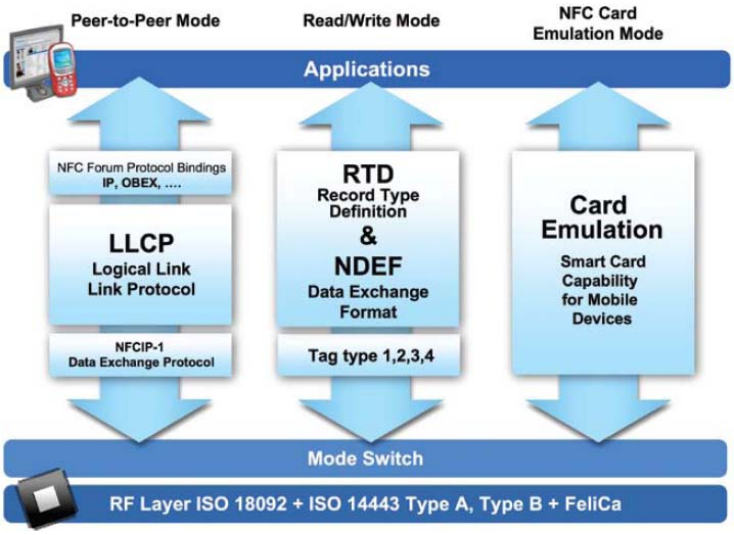


Figure 2.2: NFC specifications, from [WNFCF].

2.4. BACKWARD COMPATIBILITY.

NFC is backward compatible with some widely adopted contactless smart card standards as depicted by Figure 2.3, such as ISO/IEC 14443 (RFID) Type A (MiFare, Philips) and Type B; as well as FeliCa cards (Sony). This compatibility enables NFC to be used with already existing contactless infrastructure.

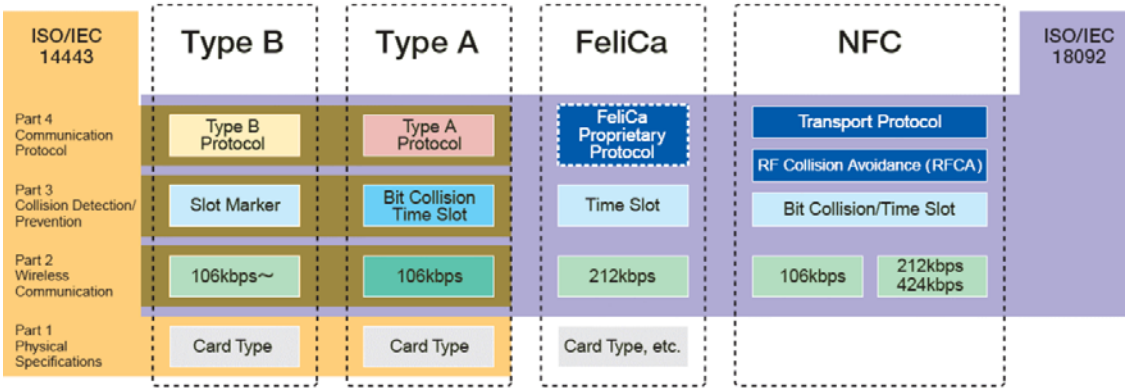


Figure 2.3: NFC backward compatibility, from [WSONYF].

However, this backward compatibility offers NFC several communication scenarios; hence, the suitable protocol and interface have to be chosen: the so-called "communication modes". NFCIP-2 defines the selection mechanism of the communication mode. The following section describes the communication modes.

2.5. COMMUNICATION MODES.

ISO/IEC 21481 Standard specifies NFCIP-2, describing the mechanism to detect and select one of the available operation modes in the 13.56 MHz frequency band. NFCIP-2 is designed to not interfere with any ongoing communication (see Section 2.3).

An NFCIP-2 compliant device has to implement the following operation modes:

- **NFC**: is specified in ISO/IEC 18092. This operation mode uses the NFCIP-1 protocol (P2P), including "active" and "passive" NFC communication modes.

- **PCD**: is specified in ISO/IEC 14443. This operation mode is used when the contactless card acts as an RFID card.

- **VCD**: is used when the contactless communication occurs in accordance with ISO/IEC 15693, a widely adopted standard for item level RFID tracking.

Focusing on the NFC operation mode, all NFCIP-1 compliant devices must support 106, 212, and 424 kbps data rates, in both active and passive NFC communication modes.

Devices containing a power source are called "active" devices, such as: mobile devices or NFC readers. Devices without any available power source are called "passive" devices, such as RFID or FeliCa tags, acting as a transponder. Some active devices, like the NFC-enabled mobile devices, can act as passive devices using the "card emulation mode" (see Section 2.3).

The NFCIP-1 compliant devices can act as "Initiator" or as "Target", as described in [ECMA340]:

1. "Target" is the default mode for a device in NFC operation mode.

2. Target has to silently wait for an incoming command from Initiator, without generating an RF field.

3. If any device-related application requires initiating the communication, the device must change to "Initiator" mode.

4. The Initiator's application shall decide the NFC communication mode (active or passive) and the transfer data rate: 106, 212, or 424 kbps.

ISO/IEC 18092 Standard defines the NFC communication modes for NFC operation mode (NFCIP-1): active and passive NFC communication modes:

- **NFC active communication mode**: in this mode both active devices Initiator and Target use their own RF field to communicate. Initiator starts the communication and turns off its RF field to allow Target to respond by setting up its self generated RF field. Figure 2.4 depicts the NFC active communication mode.

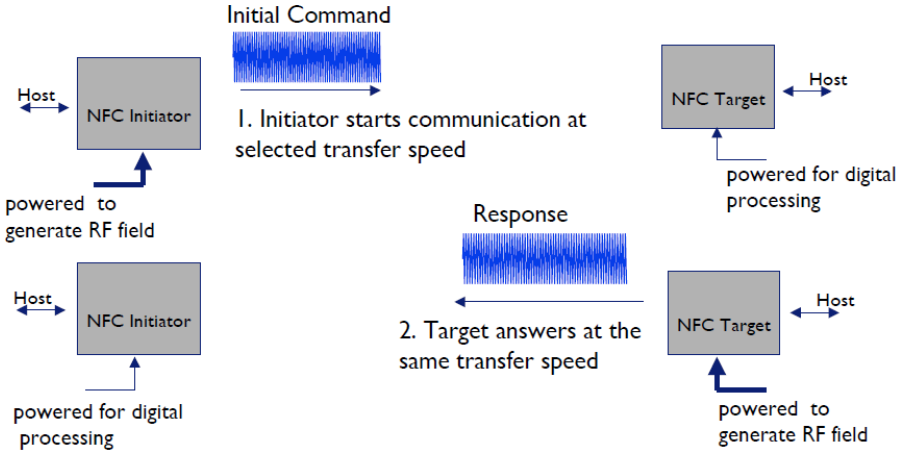


Figure 2.4: NFC active communication mode, from [WNXPP].

- **NFC passive communication mode**: in this mode Initiator starts the communication the same way as in the NFC active communication mode, but it does not turn off its RF field. Target, usually a passive device, answers using load modulation on the Initiator's RF field (see Section 2.6.1). The coupling between the devices is called "inductive coupling". Figure 2.5 depicts the NFC passive communication mode.

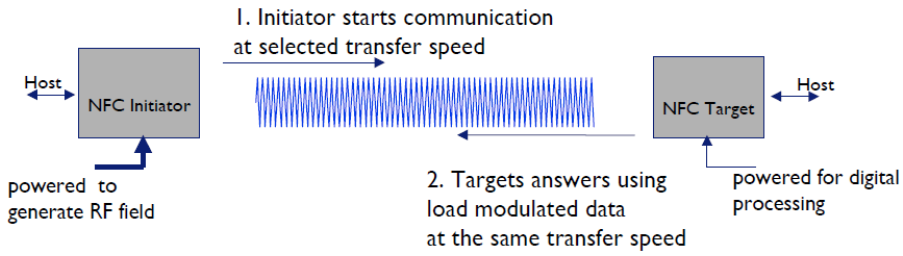


Figure 2.5: NFC passive communication mode, from [WNXPP].

2.6. OPERATIONAL PROPERTIES.

NFC interfaces operate in the RF band centred at $f_c = 13.56$ MHz: the ISM (industrial, scientific and medical) radio band. This means that no licenses are required for the use of this RF band. Nevertheless, each country can impose certain limitations on the electromagnetic emissions in this RF band. The bandwidth is $f_c \pm 7$ kHz, and the allowed magnetic field value (H) lies in the range: 1.5 A/m = H = 7.5 A/m.

The exclusive use of the magnetic field limits the maximum operating distance between two NFC-enabled devices: up to 20 centimetres [BPDG2011].

NFC employs two different coding schemes to transfer data. If an active device transfers data at 106 kbps, a modified Miller coding with 100% modulation depth is used. In all other cases Manchester coding is used with a modulation ratio of 10%.

NFC devices are able to receive and transmit data at the same time. Thus, they can check the RF field and detect a collision if the received signal does not match the transmitted signal [BPDG2011].

The modulation scheme used is Amplitude Shift Keying (ASK).

The possible NFC data rates are: 106, 212, and 424 kbps. Initiator establishes the used bit rate [LSG2009].

2.6.1. Inductive coupling.

The communication between two devices in NFC is performed by magnetic field induction for data transmission/reception. This kind of communication is known as "inductive coupling", where two loop antennas (coils) are located within each other's "near field": the area from the antenna to the point where the electromagnetic field forms is called the "near field" (less than one wavelength) of the antenna [FK2003].

Initiator device (also known as "the reader") provides a carrier field and Target device ("the transponder") answers by modulating the provided field. Target device may draw its operating power from Initiator-provided electromagnetic field, thus making Target device a transponder.

The magnetic field of the reader plays a fundamental role, since it is essential for inductive coupling to take place. The reader must generate an electromagnetic wave strong enough to allow the transponder to feed its circuit and to generate the response signal. This response signal contains all the data stored in the memory of the transponder [BPDG2011].

For a detailed description of the "inductive coupling" and "load modulation" concepts see [FK2003].

It is necessary to briefly review some of the concepts on electromagnetism to understand all the factors that allow a satisfactory inductive coupling.

2.6.1.1. Magnetic field.

The magnetic field generated by an NFC device depends on the used type of antenna. The most common antennas are square or cylindrical spirals.

The magnetic field (H) generated by a cylindrical antenna is given by the equation:

$$H = \frac{I \cdot N \cdot r^2}{2(r^2 + d^2)^{3/2}} \quad (1)$$

Where I is the current flowing along the spiral, N is the number of turns of the spiral, r the spiral radius, and d is the distance from the centre of the coil in the x-axis. The near field occurs provided that $d \ll \frac{\lambda}{2\pi}$ since this is the limit from which the far field starts.

The magnetic field generated by a square spiral whose sides are a and b in length is given by:

$$H = \frac{N \cdot I \cdot a \cdot b}{4 \cdot \pi \sqrt{\left(\frac{a}{2}\right)^2 + \left(\frac{b}{2}\right)^2 + d^2}} \left(\frac{1}{\left(\frac{a}{2}\right)^2 + d^2} + \frac{1}{\left(\frac{b}{2}\right)^2 + d^2} \right) \quad (2)$$

2.6.1.2. Mutual inductance.

When a second spiral is located next to the spiral that emits the magnetic field, this second spiral is affected by the generated magnetic flow. The magnetic flow generated by a spiral antenna is given by the following expression:

$$\Psi = N \cdot \Phi = N \cdot \mu_0 \cdot \mu_r \cdot H \cdot A \quad (3)$$

Where Φ is the sum of the magnetic flow that occurs in a plane A, $\mu_0 = 4\pi\ 10^{-7}$ Vs/Am and μ_r the relative permeability, whose value depends on the magnetic properties of the antenna's material.

The inductance (L) of the spiral antenna is the ratio between the generated magnetic flow (Ψ) and the electric current (I):

$$L = \frac{\Psi}{I} \quad (4)$$

Therefore, the mutual inductance that occurs between a spiral of an active device (A) and a passive one (P) is denoted as:

$$M_{P,A} = N_P \cdot \frac{\Phi_{P,A}}{I_A} = N_P \oint_{A_P} \frac{B_P}{I_A} dA_P \quad (5)$$

That is, the ratio of the magnetic flow that runs through the passive spiral ($\Phi_{P,A}$) and the current of the active spiral (I_A). B is the magnetic flow density, denoted as:

$$B = \mu_0 \cdot \mu_r \cdot H \quad (6)$$

The magnetic inductance has the same value for $M_{P,A}$ and $M_{A,P}$.

Expressions from 1 to 6 are stated in [FK2003].

2.7. ANTI-COLLISION MECHANISM.

NFC comprises two types of anti-collision mechanisms: *Carrier Sense* (CS) and *Binary Tree Protocol* (BTP), depending on the used NFC communication mode [BPDG2011].

NFC passive communication mode: Initiator applies the CS mechanism to minimize possible collisions with signals from other devices. Before starting the communication, Initiator listens to the channel during a certain time period (T). If any detected magnetic field reaches a threshold (H_t), the device does not start the communication and waits for a random time (T_w) before listening to the channel again. Otherwise, if no detected signal is greater than H_t during T, the channel is assumed to be free and Initiator starts the transmission after a guard time (T_g) > 5. T is calculated as:

$$T = \frac{P}{f_c} + n\frac{512}{f_c} \quad (7)$$

P is the initial delay time, which should be $P > 4096$, and n is a random number between $0 \leq n \leq 3$.

Another collision type can be produced in the NFC passive communication mode: when two or more passive devices are fed by the Initiator's magnetic field at the same time. In this case, after the first signal of the active device, all the involved passive devices will respond simultaneously, causing a collision. Initiator solves this problem by using a BTP anti-collision protocol [BVG2009].

NFC active communication mode: once received the signal from Initiator, Target must wait a guard time T_g before sending the response. This guard time must meet:

$$\frac{768}{f_c} \leq T_g \leq \frac{2559}{f_c} + n \cdot \frac{512}{f_c}. \quad (8)$$

During the time T_g, Target executes the CS to assure that no other devices are using the channel. Should the channel be engaged, Target will wait a guard time before attempting to send its response:

$$\frac{768}{f_c} \leq T_g > \frac{1024}{f_c} \quad (9)$$

Expressions from 7 to 9 are stated in [ECMA340].

2.8. COMPARISON WITH OTHER TECHNOLOGIES.

NFC is based on integrating existing RFID technology to portable consumer devices, such as mobile phones. NFC technology has its own position in contrast to other wireless technologies. Figure 2.6 illustrates how NFC positions to close proximity, and lower data rates.

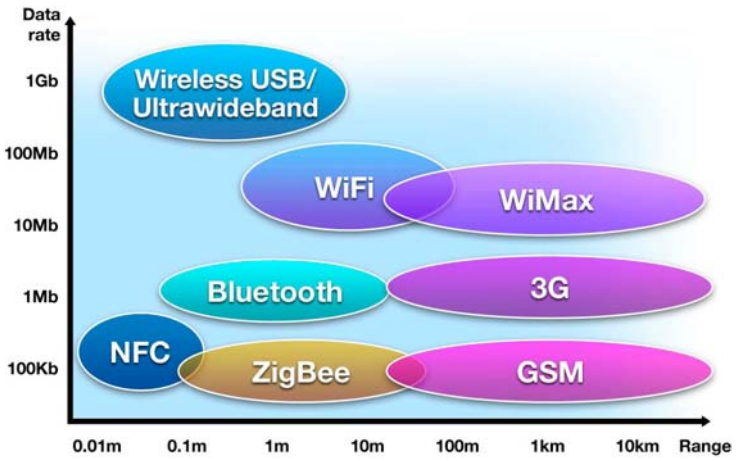


Figure 2.6: NFC compared to other wireless technologies, from [WNFCF].

The competence might arise with other short-range technologies, such as IrDa, Bluetooth, or ZigBee.

Table 2.1 depicts an NFC comparison with other short-range technologies.

NFC fulfils the "touching" paradigm, allowing user transactions to start automatically simply by bringing an NFC-enabled phone into proximity with another NFC device (a reader, a transponder, or another NFC-enabled phone). The simplicity constitutes

the foremost pro over other short-range communication technologies, such as IrDa, Bluetooth or ZigBee.

Features	NFC	IrDa	Bluetooth	ZigBee
Set-up time	< 0.1·ms	0.5 s	6 s	>15 ms
Range	Up to 20 cm	Up to 5 m	Up to 30 m	Up to 50 m
Data rate	Up to 424 kbps	115 kbps	2.1 Mbps	Up to 250 kbps
Operation mode	Active-Active Active-Passive	Active-Active	Active-Active	Active-Active Active-Passive
Connectivity	Point-to- point	Point-to-point	Point-to-multipoint	Point-to-multipoint
Frequency	13.56 MHz	38 kHz (NEC specification)	2.45 GHz	868 MHz (EU) 915MHz (USA) 2.45 GHz (rest)
RFID compatibility	Yes	No	No	No
Usability	Easy, intuitive, fast	Data centric, easy	Data centric, medium	Data centric, easy
Selectivity	High, given, security	Line of sight	Who are you?	Wake up and answer
Use cases	Pay, get access, share, initiate service, easy set up	Control and exchange data	Network for data exchange, peripherals	Industrial control, sensors networks, domotics
Consumer experience	Touch, wave, simply connect	Easy	Configuration needed	Configuration needed
Cost	Low	Low	Medium	Medium

Table 2.1: NFC compared to other short-range technologies, from [BPDG2011].

In addition, the NFC backward compatibility enables NFC to be used with already existing contactless infrastructure (see Section 2.4).

The main NFC limitation compared to other short-range technologies, such as Bluetooth or ZigBee, is the very short communication range. Depending on the application, it can bring along several disadvantages. As stated in [SHY2007]:

- "NFC does not suit to portable devices requiring online connectivity to another portable device or to a fixed access point".

- "Lower data rates together with the short communication range can make the touch-based transfer of longer data blocks unpleasant".

- "The placement of the antenna is more critical. The location of the antenna has to be indicated to the user".

However, the NFC-enable devices can set up a connection using faster and longer range protocols like Bluetooth or Wi-Fi [WNFCF], meaning that the disadvantages exposed earlier can be partly overcome.

2.9. APPLICATION SCENARIOS.

NFC can be built in a variety of devices, such as mobile phones, printers, or digital cameras sending photos to a TV by just touching it. ABI Research estimates that "Mobile handsets remain the key market for NFC but, increasingly, the potential of the technology is driving NFC into other devices and form factors" [WABIR].

However, the main current applications of NFC technology are those in which mobile phones are involved. Mobile phones already outgrew their original communication purpose, evolving into portable multimedia systems. Furthermore, these devices enable NFC technology to be cheaply combined with a display, a keyboard, and several types of communication channels like the Internet or Bluetooth [WNFCF].

Figure 2.7 depicts some possible NFC application scenarios.

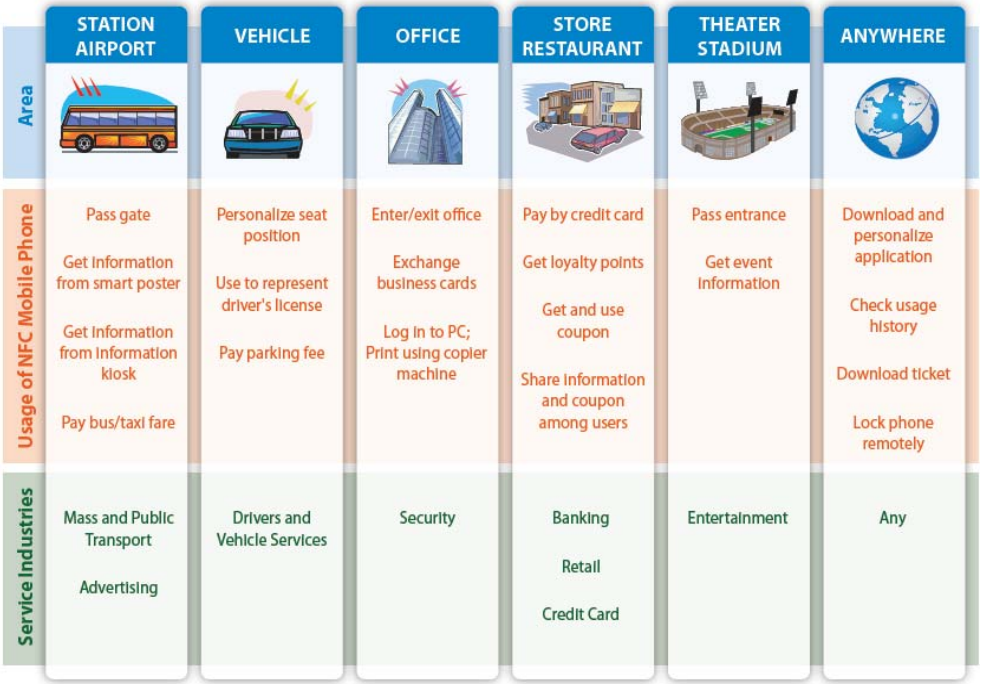


Figure 2.7: NFC application scenarios, from [WNFCF].

Being able to run multiple applications and fulfilling the "touching" paradigm (see Chapter 1), NFC-enabled phones can constitute an alternative to contemporary physical wallets by means of providing mobile ticketing and mobile payment.

In addition, the NFC-enabled phones can provide end users with a wide range of "touching" applications intended to:

- Read embedded information in smart posters about: news, weather, or travel timetables.

- Share information with others NFC users: notes, visit cards, contacts, events.
- Browse the Internet, make phone calls, or send text messages.
- Manage personal documents: travel cards, or identity documents.
- Manage electronic keys: car, house, office, hotel room.

2.10. ENERGY CONSIDERATIONS.

Mobile devices run on the limited energy available in a battery and thus the energy consumed by the system determines the length of the battery life.

The power consumption of a mobile device is determined by its hardware. Major hardware components of a mobile device include the processor, memory, storage, network interface, display and other interfacing devices. The instantaneous power consumption is determined by all hardware activities occurring at the same time. Power efficiency of individual hardware components can be achieved by means of better material and of better mechanical, circuit, and architecture designs [CB1998] [YG1997].

During the wording of this thesis, there is not any public available study on the specific associated power consumption with the application of NFC technology to mobile devices. Obviously, NFC increases the power consumption of mobile devices, depending also on the application.

For further discussion on general mobile power consumption see [LL2008].

2.11. SECURITY ISSUES.

As stated in [HB2006], several kinds of security attacks can be performed on the communication between two NFCIP-1 compliant devices:

Eavesdropping: or "data sniffing" is a security attack where, "using an antenna and analysis equipment, an attacker tries to listen to any data being sent between two devices", as stated in [HG2008]. Sniffing the data interchanged between two NFCIP-1 compliant devices requires roughly ten meters for active devices and one meter for passive devices. However, sniffing passive devices is more difficult because the communication is made by inductive coupling [HB2006].

Data Modification: an attacker tries to modify the data being sent out. As stated in [HB2006], data modification in an NFCIP-1 communication is possible for bit rates higher than 106 kbps and possible only for some small extend on the 106 kbps bit rate.

Man-in-the-middle: an attacker tries to interfere the communication between two devices, altering the information interchanged. The attack is as successful as the capability of the attacker to be unnoticed by both devices is. To be unnoticed, the attacker blocks the signal from the transmitter device and sends its own data out to the receiver device. However, the NFCIP-1 anti-collision mechanism allows NFC devices to detect jamming or incoherent signals (see Section 2.7). In addition, the short range of NFC (up to 20 cm) implies this attack is practically impossible to be performed (see Section 2.6).

As a conclusion, although the NFC standard provides some features intended to prevent some attacks, the NFC security has to be improved by using dedicated cryptography to establish a secure channel between communicating devices.

Chapter 3. METHODOLOGY AND SYSTEM ANALYSIS.

"Divide each difficulty into as many parts as is feasible and necessary to resolve it"

René Descartes (1596-1650)
French philosopher and mathematician

This chapter presents the development methodology used in this thesis, together with the analysis of the system. The methodology is applied to the entirety of the project, from the early stages concerned with the problem domain, to the achievements related to the solution domain. The following sections explain how the methodology applies to the development process. Then, the results of the system analysis are presented. Finally, the technology chosen to address the development is detailed.

3.1. INTRODUCTION.

A development methodology has to manage all the phases of a software development process, carrying out tasks which range from the study of the "problem domain": Requirements Elicitation (RE), system analysis; to the "solution domain": design, implementation, testing.

As stated in [WC2W], "problem domain is a Software Engineering term referring to all information that defines the problem and constrains the solution. The constraints are part of the problem as well". This term is also known as "scope of analysis" in Software Engineering.

"A problem domain includes the goals that a customer (or the problem owner) wishes to achieve, the context within which the problem exists, and all rules that define essential functions or other aspects of any solution achieved. It represents the environment in which a solution will have to operate, as well as the problem itself" [WC2W].

It should be noted that a customer usually identifies the existence of a good chance to develop some solution instead of a real "problem". From an engineer's viewpoint, a "problem domain" constitutes a collection of conditions that demands the engineer to develop a solution.

The problem domain is described by means of user stories, functional and non-functional requirements, and/or use cases (see Appendix 3).

While the “problem domain” defines the environment where the solution will come to work, the “solution domain” defines the abstract environment where the solution is developed. The differences between those two domains are the cause for possible errors when the solution is planted into the problem domain.

“In Software Engineering, architecture and development of software, hardware, and networking constitute the solution domain. These are the tools with which a solution to a set of user-requirements is achieved”, as stated in [WC2W]. Chapter 4 deals with the solution domain.

As stated by Gamma in [GHJV1995]: “a good software architect focuses as much on the problem to be solved and the various forces on the problem, as he does on the solution to the problem. The Information Technologies (IT) industry has a tendency to focus on the solution”. Focusing only on the solution implies an important risk in a development project: its results might constitute a very good solution to an undetermined problem that is not the problem that the expected solution users face.

“In software development, both of these domains constrain the developer. They might also overlap if, for instance, the user happens to require that the solution be achieved for certain commodity hardware and operating system”, as stated in [WC2W]. Anyway, this overlap does not make the concepts “problem domain” and “solution domain” indistinct.

This chapter addresses the description of the “problem domain” as an opportunity to develop a hardware/software platform aimed at: (see Chapter 1)

- *Constitute a platform for pilot testing intended to the planning of a possible large-scale deployment at the UPCT.*

- *Serve as a base framework for future projects which deal with NFC technology.*

The global functionality of the platform is, as stated in Chapter 1, “the effective monitoring of student attendance”, using NFC technology as basis. It is necessary to design a UI which enables the users of the system to properly interact with it. The UI Usability features are undoubtedly one of the essential requirements for finding a viable solution.

A crucial factor when carrying out a project is the choice of the development methodology to be used. In this thesis the selected methodology is an adaptation of XP, which falls into the group known as “Agile Methodologies”. Such methodologies are especially aimed at small projects, and they provide a customized methodological solution

in these environments which represents a great simplification comparing to traditional methods. Nevertheless, Agile Methodologies do not waive the essential practices to ensure the product quality.

Agile Methodologies, though recent, are very popular for their industrial vision about software development. Besides being very suitable for small and medium sized projects, Agile Methodologies are particularly interesting in those projects where development teams are small, with short deadlines and the requirements are volatile and/or based on new technologies. These are the main reasons why XP has been chosen to carry out the development of this project, since this methodology meets every one of them.

Agile Methodologies are based on values compiled in The Agile Manifesto [WAGILMAN] by The Agile Alliance [WAGILEA], a non-profit organization that promotes concepts related to the agile software development as well as to help other organizations to adopt these concepts.

The Agile Manifesto gathers twelve principles that present characteristics that distinguish an agile process from a classical or traditional one. The first two principles are general and summarize a great deal of the agile spirit. The rest are related to the procedure to be followed and the development teamwork, as far as aims and organization are concerned. These principles are:

i. Priority is given to customer satisfaction by rapid and continuous delivery of useful software which represents a business value.

ii. Welcome changing requirements. Changes are captured for the customer to have competitive advantage.

iii. Working software is delivered frequently with short time intervals between deliveries (from two weeks to two months).

iv. Close daily cooperation between business people and developers throughout the project.

v. Projects are built around motivated individuals who should be provided with the environment and support that they need and be trusted to accomplish their tasks.

vi. Face-to-face conversation is the most effective and efficient form of communicating information within a development team.

vii. Working software is the principal measure of progress.

viii. Agile processes promote sustainable development. Promoters, developers and customers should be able to get on well with each other.

ix. Continuous attention, technical quality and good design improve agility.

x. Simplicity is essential.

xi. The best architectures, requirements and designs emerge from self-organizing teams.

xii. In regular intervals the team reflects upon their effectiveness and adapts to changing circumstances accordingly.

Appendix 1 shows an overview of Agile Methodologies. In particular XP, as one of the most widespread, is introduced.

Table A1.1 in Appendix 1 summarizes the comparison between agile and classical methodologies (also known as traditional or heavyweight methodologies). The differences between them do not only affect the development process but also the context of the team together with its organization.

The following sections are structured as follows:

Section 3.2 explains the adaptation of XP to this project.

Section 3.3 is devoted to the usability model proposed by Francisco Montero, as it is the model applied to the development process in this thesis.

In Section 3.4 the analysis of system functionality is performed, with particular emphasis on the RE (also known as Exploration phase). For this purpose Requirements Elicitation Methodology for Software Systems (REMSS) proposed by Amador Durán [DB2002] is used. It is important to note that during the wording of this section only the end result of requirements and analysis deliverables have been considered, assuming that the process followed to achieve them has been iterative and incremental.

Section 3.5 discusses the system architecture where the developed platform is integrated, thus allowing identifying new requirements related to the development technologies to be used such as the programming language, or the tools to carry out the development. The system architecture also defines those externals systems interacting with the system under development.

Finally, Section 3.6 gathers the selected development technology set, discussing the reasons to its usage.

This latter two sections deal with the "solution domain", identifying new requirements related to the development process. Hence, they are part of the analysis phase.

3.2. DEVELOPMENT METHODOLOGY USED.

As stated in Section 3.1, XP is the selected methodology for carrying out the development process during this thesis.

XP defines several roles (see Section A1.3.2, in Appendix 1). To determine the responsibilities of all the participants in the development process, they have to be assigned to one or more XP roles. The assigned participants to each XP role in this thesis are gathered in Table 3.1.

XP Role	Assigned Person
Programmer	Thesis author
Client	Thesis supervisors
Tester	Thesis author
Tracker	Thesis supervisors
Coach	Thesis author
Consultant	Thesis supervisors / Thesis author
Big-Boss	Thesis supervisors

Table 3.1: XP roles assignment.

The life-cycle of the XP process consists, in outline, of the following steps:

1. Customer defines the value of the business to be implemented.

2. Programmer estimates the necessary effort for its implementation.

3. Customer selects what to build, according to their own priorities and time restrictions.

4. Programmer builds the selected business value, driven by the unit tests.

5. The acceptance tests are run against the user requirements.

6. Restart on step 1.

Both Customer and Programmer learn in each iteration of this cycle. No more working pressure than the estimated (in the steps 2 and 3) must be put on Programmer since software quality may be lowered or deadlines may be impossible to meet. Likewise, Customer has an obligation to manage the product delivery to make sure that the business has the highest possible value as a result of each iteration.

The ideal XP life-cycle consists in six phases [BK2000]: *i)* Exploration, *ii)* Release, *iii)* Iterations, *iv)* Production, *v)* Maintenance, and *vi)* Project death.

This thesis has addressed the XP process phases ranging from first to fourth: Exploration, Release, Iteration and Production. Due to the time limitation of this thesis (four months) the stages of Maintenance and Project-Death have not been feasible to address:

- **Exploration**: the system requirements has been analysed using REMSS. After the analysis of each requirement, and once validated by the customer, it is documented in detail (see Appendix 3).

- **Release**: the release planning has also been performed for each iteration. The deadlines have been set according to estimates of the complexity for each delivery. The degree of complexity has been calculated based on two factors: *i)* the novelty and difficulty-to-apply ratio of the technology used, and *ii)* Programmer's objective experience. The values of novelty and difficulty of the technology have been estimated by the thesis supervisors, playing the Tracker XP role in this case. The thesis author's objective experience, playing the Programmer XP role, has been calculated according to his transcript of records and previous experience.

- **Iterations**: Three iterations have been defined. Each of them has performed the following tasks: RE, design, and testing. The releases of each iteration are as follows: *i)* prototype of mobile device application, *ii)* prototype of server software, and *iii)* version for commissioning tests. The duration planned for each iteration has been a month.

- **Production**: has not been approached strictly due to the unavailability of a real end user who explores the result. However, this phase has served as a testing phase for the development in each iteration.

The following paragraphs define how the XP practices have been addressed in the project.

- **The planning game**: *there is frequent communication between Customer and the development team. The development team estimates the required effort to implement user stories and Customer decides on the scope and delivery times of each iteration*.

  Customer XP role has been played by the thesis supervisors. Thus, a formal and strict planning has been made, since they are the most qualified to evaluate the effort of each task and, what is more, they are responsible for establishing delivery deadlines.

- **Small deliveries**: *the aim is to quickly produce working versions of the system, even if they do not include all the functional requirements. A version constitutes a valuable result for the business and its delivery time should not take more than three months*.

  The project development has been performed by defining and implementing incremental cycles developing basic features, which have been increased by subsequent iterations.

- **Metaphor**: *the system is defined by means of a metaphor or a group of metaphors shared by Customer and the development team. A metaphor is a shared story which describes the way the system should operate (set of names that act as vocabulary to tackle a system domain in order to help the nomenclature of classes and system methodologies).*

  There are two metaphors of the system for this project: *i)* NFC technology, and *ii)* the monitoring of student attendance.

- **Simple design**: *the simplest solution must be designed in a way that it can be implemented at a certain point of the project.*

  This has led to different implementation solutions in a short space of time (see Section 3.6.5.1).

- **Test-driven development (TDD)**: *the production of the code is driven by unitary tests. These are established by Customer before the code is written and they are constantly rebuilt after each modification of the system.*

  The unit tests have been performed during the implementation of each of the system functionalities (see Section 4.6 in Chapter 4). Besides, the acceptance test cases have been inherently established in RE (see Appendix 3).

- **Refactoring**: *it is a constant activity of code reconstruction with the aim of removing code duplication, improving its legibility, simplifying it and making it more flexible to facilitate subsequent changes. The code internal structure is improved without altering its external behaviour.*

  During each iteration, the internal code structure has been improved without changing its external behaviour.

- **Pair programming**: *every code production must be carried out through Programmers' pair work. This entails implicit benefits (lower error rate, higher design quality, and higher satisfaction among programmers.).*

  This practice, the flagship of XP, has been impossible to apply in this project, because the software coding process has been performed just by the thesis author.

- **Code collective property**: *all Programmers are authorised to change any part of the code at any time.*

  In being an individual thesis, this practice has been strictly applied and in an unavoidable way.

- **Continuous integration**: *each part of the code is integrated into the system once it is ready. In doing so, the system may be integrated and rebuilt several times on the same day.*

  The implementation of each functionality into a different and loosely-coupled module greatly facilitates the integration of each individual module (see Section 4.1 in Chapter 4).

- **Forty hours per week**: *people must work a maximum of forty hours per week. Overtime is not allowed in two consecutive weeks. Should this happen, a problem is occurring and should be settled. Working overtime discourages the team.*

  No doubt, this practice has been particularly useful in this thesis.

- **On-site Customer**: *Customer must be present and available to the team at any moment. This is one of the main success factors of an XP project. Customer constantly leads work towards what contributes more value to the business and programmers are able to solve any related doubt immediately. Oral communication is more effective than written communication.*

The communication between Programmer and Customer (role played by the thesis supervisors) has been direct and fluid, having frequent meetings and almost ever-present availability.

- **Coding standards**: *XP emphasizes that Programmers' communication takes place through the code. Therefore it is essential to follow programming standards to assure code legibility.*

  Different coding standards have been followed (see Section 3.6): Structured Query Language (SQL), Object Oriented (OO) design patterns, Mobile Information Device Profile (MIDP), and Connected Limited Device Configuration (CLDC). In addition, the coding process has followed the code conventions for Java programming [WCCJP] in defining classes, interfaces, methods, variables, and constants.

This thesis deeply explores the user requirements, thus discovering new ones outside the system functionality to be added to the development process to obtain more optimal results. The development considers, in an empirical way, a non-functional system requirement related to Usability. Thus, the following section discusses the usability model applied to the software development of this thesis.

3.3. USABILITY MODEL.

The usability model applied as a non-functional requirement to the development of the thesis is the proposal by Francisco Montero [MF2005], which advocates mixing the international standards with ergonomic criteria, using a user-centred viewpoint.

In this usability model, summarized in Table A2.1 in Appendix 2, the ergonomic criteria are associated to the criteria of ISO-9126 usability standard. These resulting criteria are grouped with the factors of *understandability*, *learnability*, and *operability*. In consequence, the criteria are directly linked to Usability.

The main characteristics of the model are:

- Based on internationals standards like ISO-9126.
- It is an open proposal, in which other usability criteria can be added and linked to the existing ones.
- It uses ergonomic criteria.

- It allows the techniques of Usability to be put into practice.

This section has introduced the usability model applied to the software development of the thesis. In Chapter 4 it is explained how the model has been applied to the development.

Section A2.4 in Appendix 2 gives a more detailed overview of Usability.

The following section deals with the information, functional and non-functional requirements of the system.

3.4. SYSTEM REQUIREMENTS ELICITATION.

RE in Exploration Phase of this project has been carried out using REMSS.

This methodology establishes a set of templates to represent the functional, non-functional and the information requirements, and uses language patterns (L-patterns) for better information interchange between the project developer staff and the customer. In addition, it also supports the Unified Modeling Language (UML) use case diagrams to enhance understanding [WUML].

UML was originally defined by Booch, Rumbaugh and Jacobson [BRJ1999], and it is currently maintained by the non-profit organization Object Management Group [WUML]. UML is nowadays the mainstay of OO Analysis (OOA) and Design (OOD).

UML use case diagrams overview the usage requirements for a system and, in outline, they depict the use cases themselves, the actors and the associations between them [AS2004].

It should be noted that the main objective in this thesis is not to strictly follow the entire REMSS methodology, so it has been tackled with some flexibility.

3.4.1. Definition of the information requirements.

Information Requirements (IRQ) reflect the information involved in the system domain. Every used information unit is defined and its items are detailed, along with the Restriction Checks (CRQ) that they have to accomplish.

Table 3.2 gathers an example of the definition of an information requirement. Most fields are self-described by means of their names.

IRQ–01	**Information About Users.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• UC–01 Configure user information. • UC–02 Register attendance. • UC–03 Activate group. • UC–04 List attendees. • UC–06 Authenticate user. • UC–09 Check system logs. • UC–06.01 Check user type. • UC–06.02 Authenticate password. • UC–06.03 Validate group. • UC–06.04 Acquire student password. • UC–06.05 Acquire teacher password. • UC–06.06 Acquire student group. • UC–06.07 Acquire teacher group.	
Description	The system must store the information related to the users, in specific:	
Specific Data	• User full-name. • User login-name. • User password. • User type.	
Time to Life	**Average**	**Max**
	5 years	10 years
Concurrent Instances	**Average**	**Max**
	1	1
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The access of every user could only be in a sequential way. That is why the number of concurrent instances in the system is only one. The student preserves the same information during the whole duration of his studies.	

Table 3.2: Information requirement's definition example.

The definition of each information requirement also identifies the functionalities related to it, allowing the programmers to keep the trace of the information usage.

The IRQ's definitions complete list can be found in Appendix 3.

3.4.2. Definition of actors.

An actor represents a role played by persons or external systems interacting with the system under development.

ACT–02	**Student.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Description	This actor represents a student attending at lectures or laboratory practices.
Comments	None.

Table 3.3: Actor's definition example.

Table 3.3 shows an example of a system actor's definition. The fields are self-described by their names, focussing the interest on the actor description.

The complete list of actors' definitions can be found in Appendix 3.

3.4.3. Definition of functional requirements.

A functional requirement defines the function of a software component, enumerating the required steps to accomplish the function, the related requirements, and other considerations like possible exceptions raised and/or preconditions.

The functional requirements are equivalent to the so-called "fully-dressed" use cases. A functional requirement comprises more detailed and formally a functionality description than a "user story", since a user story does not enumerate required steps or exceptions, acting just as a first approach to a user requirement (see Section A3.3.1 in Appendix 3).

Table 3.4 shows an example of a functional requirement definition.

The functional requirements definitions complete list can be found in Appendix 3.

UC–02	**Register Attendance.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.	
Start event	The system should behave as described in this use case when a student decides to record his attendance.	
Precondition	The student has to configure his own data in advance (UC–01 Configure user information).	
Normal Sequence	**Step**	**Action**
	1	The student asks the system for starting the register attendance process.
	2	The system automatically obtains the current system date and time.
	3	The use case UC–05 (Check group activation) is run, using the date and time retrieved at step 2.
	4	The use case UC–06 (Authenticate user) is run, using the current date and time retrieved at step 2, and the group code retrieved at step 3 (IRQ–02 Information about groups).
	5	The system asks the DBMS for storing the attendance record of the student: the current date and time at step 2, the retrieved group code at step 3 (IRQ–02 Information about groups), and the user login-name at step 4 (IRQ–01 Information about users).
	6	The DBMS stores the attendance record of the student.
	7	The system writes a new record in the system log with the following information: current date and time of creation; the "Attendance" literal; student login-name (IRQ–01 Information about users); and group code (IRQ–02 Information about groups).
	8	The system informs the student that the process has successfully finished.
Post-condition	The DBMS has stored the information corresponding to the student attendance.	
Exceptions	**Step**	**Action**
	3	If there is no information about groups scheduling stored in the system, the system communicates this situation to the student and then, the use case finishes with no effect.
	3	If the current scheduled group has not been activated yet, the system communicates this situation to the student and then, the use case finishes with no effect.
	3	If the current scheduled group has been cancelled, the system communicates this situation to the student and then, the use case finishes with no effect.
	4	If the student is not authenticated, the system communicates this situation to him explaining the reason, and then, the use case finishes with no effect.
	4	If the user type is not a student, the system communicates this situation to him and then, the use case finishes with no effect.
	5	If the attendance has already been registered, the system communicates this situation to the student and then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	2	1 second.
	7	2 seconds.
Frequency	20 times / minute.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The expected frequency of usage is only reached during the ten previous minutes to the start of a lecture or a laboratory practices lesson (when the students register their attendance). The developed platform is able to attend up to 50 incoming user's commands per minute (not having into account the user's response time).	

Table 3.4: Functional requirement definition example.

3.4.4. Definition of non-functional requirements.

A non-functional requirement represents a demanded feature of the system, the development process itself, the production service or any other aspect of development, which usually indicates a restriction in them.

Table 3.5 gathers an example of a non-functional requirement definition. The fields are self-described by their names. The most important fields are the requirement description and the comments, serving as start point to address the requirement.

NFR–02	**Portability.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	–
Related Requirements	–
Description	The system should be easily ported into another modern operating systems of Microsoft Windows family (like Vista or Windows 7) or Linux distributions (Ubuntu, Suse, Debian, Red Hat, etc). Similarly, the system must be prepared for changes in the operating system of the mobile clients (handsets).
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	To address the first question, portability at the server side, Java free software-based products may be chosen, taking advantage of the portability of the Java architecture. To solve the portability issues at the clients, programmatic standards for mobiles devices must be used (such as MIDP, CLID or JSR-257) easily portables between different handset platforms.

Table 3.5: Non-functional requirement definition example.

The non-functional requirements definitions complete list can be found in Appendix 3.

3.5. SYSTEM ARCHITECTURE.

This section shows the design of the system architecture in which the developed platform of this thesis has to be integrated later on. Furthermore, the study of the system architecture allows selecting the software tool set which constitutes the selected technology to address the development process.

Figure 3.1 depicts the designed system architecture. There are three types of users that can be distinguished: *i)* Student, *ii)* Teacher, and *iii)* Administrator. The users i) and ii) are part of the group "NFC users", since they are equipped with NFC-enabled devices. The user iii) is in direct connection to the NFC server host by means of the server console or using a remote connection.

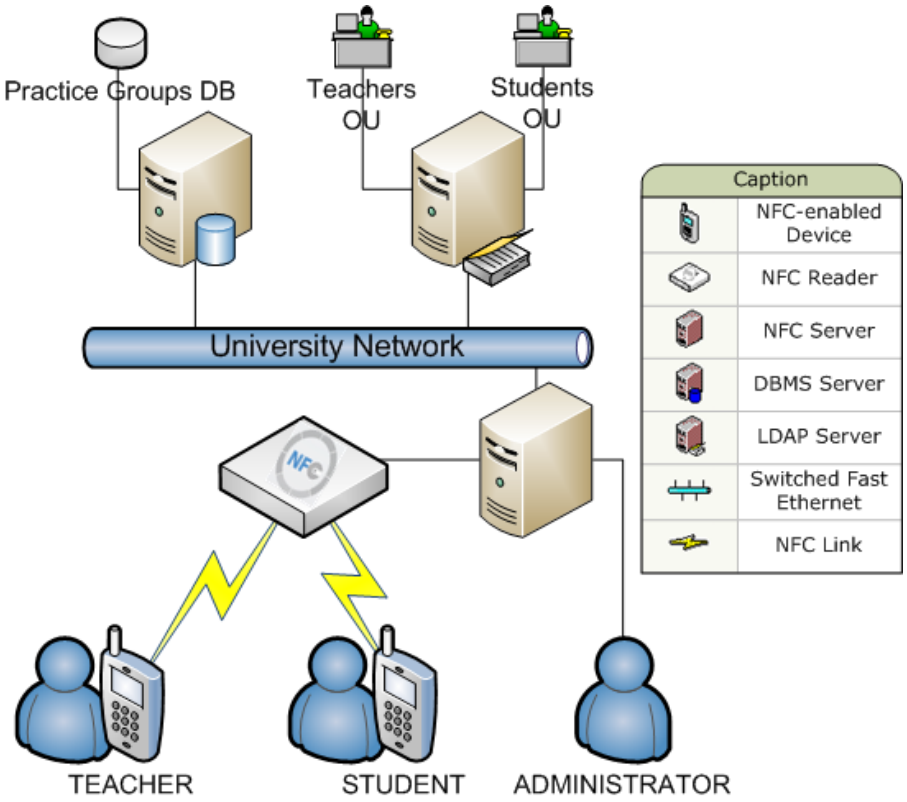


Figure 3.1: System architecture.

The main hardware components of the system architecture are:

i) **NFC-enabled devices**: which run the client software application and store the user information. The UPCT owns a Nokia 6212 NFC Classic for testing purposes (see Section 3.6).

ii) **NFC reader**: is the hardware device which allows the communication between the NFC users and the server. The reader is the mediator which allows the NFC users to interact with the system, that is, this device acts as a middleware of the system. This device is in direct connection with the NFC server host. The GIT owns an ACR-122U NFC reader for testing purposes (see Section 3.6).

iii) **NFC server host**: is attached to the university network and handles the communication link with the NFC reader. This host executes the server software application and provides access to the reader. The application receives and executes users' commands. Performing the service, the application will open connections to the DBMS and LDAP servers.

iv) **DBMS server**: DBMS is an acronym which stands for *Database Management System*. The used server to store the practices database scheme

at the UPCT is a version of MySQL server [WMYSQL]. To develop the platform in this thesis it is necessary to design and implement a practices database scheme for testing purposes, using MySQL server and according to the real scheme at the UPCT (see Section 4.4.1). Current MySQL version is 5.5.8.

v) **LDAP server**: LDAP is an acronym which stands for *Lightweight Directory Access Server*. It is an application protocol defined by the Internet Engineering Task Force as "an Internet protocol for accessing distributed directory services" [RFC4510]. The directory is structured in different Organizational Units (OU), containing user objects. An OU constitutes a method of categorizing stored items in directories, usually used either to distinguish items with the same name, or to distribute rights to manage items. The used directory server to handle users and credentials at the UPCT is a version of OpenLDAP server [WOLDAP] [WOLDAPW]. To develop the platform in this thesis it is necessary to design and implement an LDAP scheme for testing purposes, using OpenLDAP server and according to the real scheme at the UPCT (see Section 4.4.2). Current OpenLDAP version is 2.4.23.

3.6. DEVELOPMENT TECHNOLOGY.

According to the XP methodology, once the RE is performed, the technology to implement the software must be selected.

The observation of the non-functional requirements is an ever-present issue in the study of the "solution domain" (see Section 3.1). In the present system, another crucial consideration is the NFC usage.

The contemplation of these issues, together with the study of the presented system architecture in Section 3.5, allows to select the appropriate technology set to address the development. The technological choice focuses on those wide-accepted tools by the open-source community, as stated in Chapter 1.

NFR–01 Production environment, NFR–02 Portability, and NFR–04 Economic cost reduction, are the non-functional requirements with more impact on this choice (see Section A3.4 in Appendix 3).

Table 3.6 shows the required server software tools to design and implement the external systems for testing purposes (see Section 3.5): OpenLDAP and MySQL.

System Element	Software Tool
LDAP server	OpenLDAP 2.4.23
Database server	MySQL 5.5.8

Table 3.6: External systems tools.

3.6.1. Programming language.

The programming language expertise is a determinant factor to address the implementation phase of a software development. As stated by Williams, the team may incur significant learning costs when using an unfamiliar programming language. While it may be possible to switch between languages of a similar type (such as OO languages) with a relatively small learning curve, adjusting from the OO paradigm to functional languages may prove more difficult and thus slow the development process. Besides, an OO programming language emphasizes code reusability [WKL2005].

This paragraph raises some considerations:

- XP follows an OO paradigm.
- XP demands code reusability as a common practice.
- The UML notation is close related to OOA and OOD. It is used in this thesis as the tool to generate the analysis and design models.
- The author's programming expertise is mainly based on Java technologies.

This considerations drive to a simple and easy choice: Java is the selected programming language.

Java is distributed free of charge, under an open-source license [WJAVA], and is the OO programming language with the more widespread acceptance in the programmers' community. In consequence, using Java allows (almost) any development team to substantially reduce the learning curve. Thus, Java language meets the non-functional requirements mentioned earlier in this section.

For the proper execution of a Java application, the prior installation of the Java Virtual Machine (JVM) is required. JVM constitutes the runtime environment where the Java applications run.

- For the NFC server host, the Java 2 Standard Edition (J2SE) JVM is the right choice, as this is the more widely adopted by the software community [WJ2SE]. Current version is 6.23.

- In the case of the NFC-enabled devices, the Java 2 Micro Edition (J2ME) JVM [WJ2ME] is the only possible choice since it is factory-integrated inside them. Current version is 3.0.

Table 3.7 gathers the system elements with a required JVM.

System Element	Software Tool
NFC server host	J2SE JVM 6.23
NFC-enabled device	J2ME JVM 3.0

Table 3.7: System elements with JVM.

Appendix 6 summarizes the available Java-related technologies dealing with the mobile phones programming. Since these technologies constitute a wide range, only those with direct application on this thesis are presented, such as: MIDP, and CLDC.

3.6.2. Development environment.

The selected Integrated Development Environment (IDE) is Eclipse [WECLIPSE].

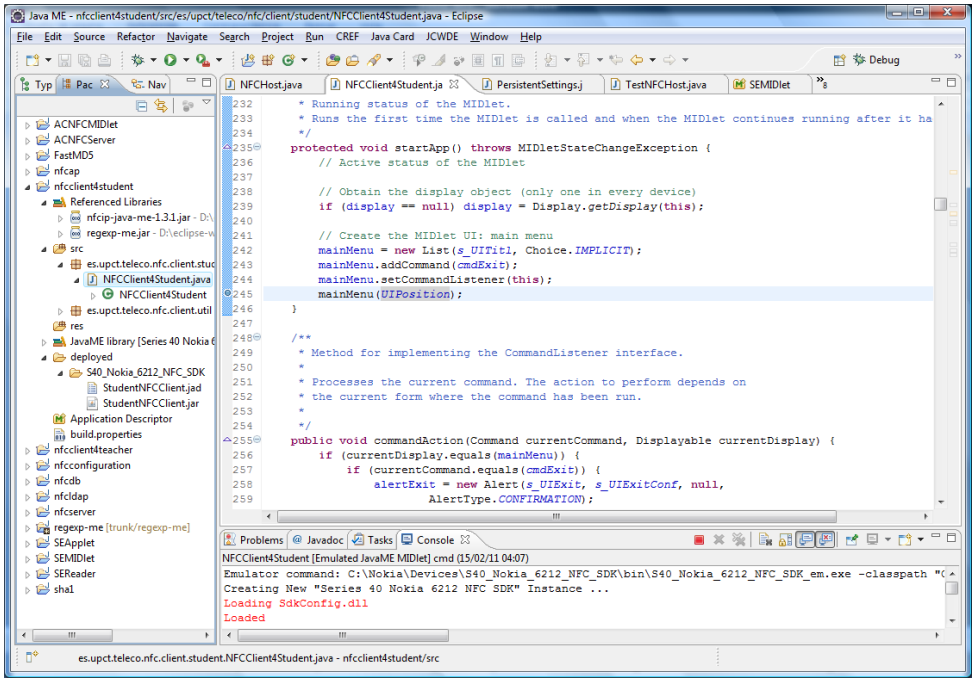


Figure 3.2: Eclipse IDE during the development.

The current version of Eclipse IDE, corresponding to 3.6, is named “Helios” and is released under the terms of an open-source license.

Figure 3.2 depicts an Eclipse IDE screen shot during the development of a MIDlet suite in this thesis (the “student” version).

Eclipse is a multi-language IDE with an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and in other programming languages including Ada, C, C++, COBOL, Perl, PHP, Python, and Ruby, to name but a few.

3.6.3. Hardware devices.

As stated in Chapter 1, the GIT is involved in an ambitious project of study and application of NFC technology, aimed at implementing the “Smart University” model at the UPCT. In this sense, the GIT acquired an NFC kit containing NFC-enabled devices for testing purposes: a reader, and a mobile phone; hence, the software development in this thesis has to meet, as a non-functional requirement, the usage of these devices.

3.6.3.1. Reader.

The reader, acquired from Advanced Card Systems (ACS) Limited, is an ACR122U NFC contactless smart card reader [WACR122U].

Figure 3.3 depicts an image of this reader.

Figure 3.3: ACS-ACR122U NFC reader.

Table 3.8 summarizes some relevant technical specifications of the ACS-ACR122U NFC reader.

ACS-ACR122U NFC Reader Relevant Technical Specifications Summary.		
USB interface	Power source	From USB
	Speed	12 Mbps (USB Full Speed)
	Supply voltage	Regulated 5 V DC
	Supply current	200 mA (maximum); 50 mA (standby); 100 mA (normal)
Contactless smart card interface	Standards	ISO/IEC 18092 NFC, ISO 14443 A (MiFare) and B, FeliCa
	Protocols	FeliCa protocol, T=CL protocol
	Operating frequency	13.56 MHz
	Smart card read/write speed	106 kbps, 212 kbps, 424 kbps
Operation	Antenna size	50 mm x 40 mm
	Operating distance	Up to 50 mm (depending on tag type)
Certifications/compliance	PC/SC, CCID, USB Full Speed.	
Device driver operating system support	Windows 98, ME, 2000, Server 2003, XP, Vista, Server 2008, 2008 R2, 7, Windows CE 5.0, Linux, Mac	

Table 3.8: ACS-ACR122U NFC reader relevant technical specifications summary.

A more complete list of the reader technical specifications is gathered in Appendix 7.

3.6.3.2. Mobile phone.

The NFC mobile phone, acquired from the European division of Nokia Corporation, is a Nokia 6212 Classic mobile phone [WNE6212] (from here on "the N6212").

Figure 3.4 depicts an image of this mobile phone.

Figure 3.4: Nokia 6212 Classic.

Figure 3.5 depicts the communication capabilities that the N6212 NFC controller chip allows.

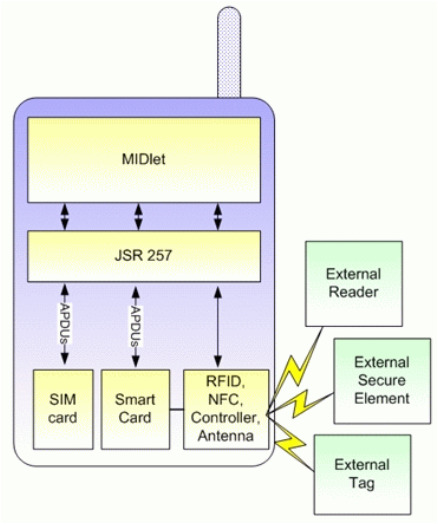


Figure 3.5: NFC phones communication capabilities, from [OG2001].

The N6212 SE contains a tamper resistant NFC controller chip, allowing the phone to act as:

1. An NFC smart card reader.

2. An NFC smart card: using the so-called "card emulation mode" which acts as a MiFare smart card (see Appendix 5).

3. An NFC P2P device: using the NFCIP protocol (see Chapter 2).

The N6212 relevant technical specifications are summarized in Table 3.9.

Nokia 6212 Classic Relevant Technical Specifications Summary.		
Operating frequency	GSM/EDGE	850/900/1800/1900
	WCDMA (UMTS)	850/2100
Communications interfaces	GPRS multi-slot class 10	4+1/3+2 slots, 32-48 kbps
	EDGE multi-slot class 10	236.8 kbps
	3G (UMTS)	384 kbps
	WAP	
	Near Field Communication	read/write/sharing
	Bluetooth	2.0
	USB	2.0
Java compatibility	J2ME CLDC	1.1
	J2ME MIDP	2.0, 2.1
Display	Type	QVGA 16 M colours, TFT
	Size	2.00 inch
	Resolution	240 x 320 pixels
Memory	Internal shared memory	22 MB
	Memory card (optional)	microSD (TransFlash), up to 4 GB
Browsing	XHTML browser over TCP/IP	
	WAP 2.0	
	Opera Mini browser	Pre-installed
Miscellaneous features	Predictive text T9	

Table 3.9: Nokia 6212 Classic relevant technical specifications summary.

A more complete list of the phone technical specifications is gathered in Appendix 8.

3.6.4. Device emulator.

Nokia provides the NFC developers with a Software Development Kit (SDK) aimed at prototyping and application testing [WN6212SDK] free of charge.

Figure 3.6 depicts an image of Nokia 6212 NFC SDK.

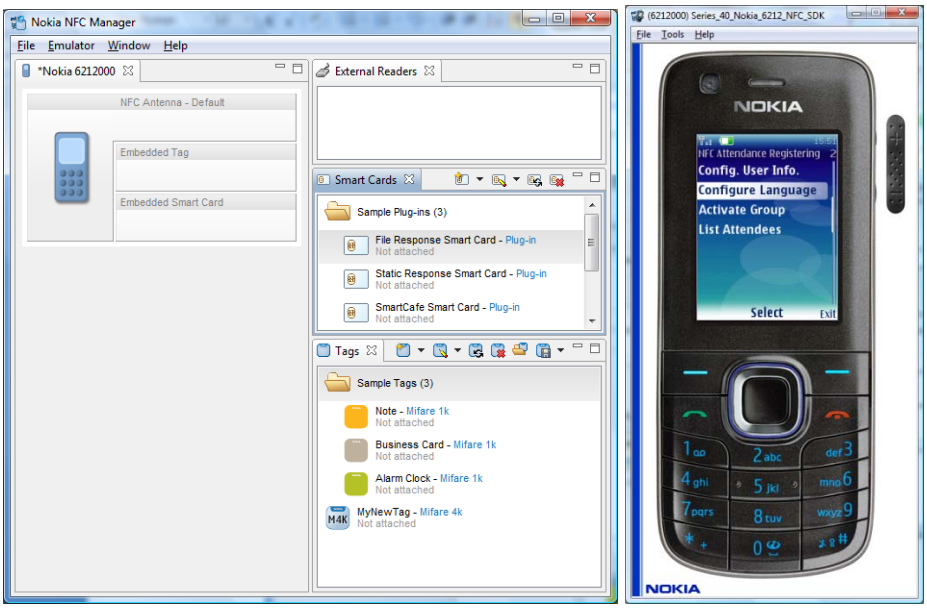


Figure 3.6: Nokia 6212 NFC SDK.

The selection of this SDK is obviously driven by the total compatibility with the mobile phone used in the development process. Furthermore, the Eclipse IDE is able to interact with this SDK: deploying and running (or even debugging) applications automatically into the SDK.

3.6.5. Application Programming Interfaces.

An Application Program Interface (API) is a software library which offers access to services provided by others software components or hardware devices; hence an API serves as an access interface to these devices or components.

This section details the APIs used to develop the hardware/software platform, focusing on the application to the N6212.

3.6.5.1. Java 2 Micro Edition.

To create and deploy applications on the N6212 the CLDC and MIDP APIs are mandatory, since they constitute the minimum required API set to develop J2ME applications (see Appendix 5). Furthermore, the N6212 is compatible with MIDP 2.1 and CLCD 1.1, as indicated by the N6212 technical specifications (see Appendix 8).

An application written for J2ME compatible devices is called a "MIDlet" and is bundled in a "MIDlet suite", which can contain one or more MIDlets. The used J2ME APIs for implementing MIDlets for the N6212 in this thesis are:

- CLDC 1.1.
- MIDP 2.1.

Data on the phone can be stored in the following locations:

- The Record Management Store (RMS): that can be used by means of the MIDP API (see Appendix 5).
- The SE: the access to the SE requires the MIDlet suite a special permission, and the MIDlet suite has to be signed by a code signing certificate (see Appendix 5).

Regarding these two storing possibilities, two application versions can be implemented in this thesis:

1. An RMS version, storing the private user data in a record store: feasible.
2. A SE version, storing the data in the internal and most secure JCP smart card, implying several considerations:
    - Design of an appropriate application protocol, since the user's credentials can be automatically retrieved from the SE by the NFC reader. Thus, the presence of these data inside the payload field of the application protocol is not required anymore (see Section 4.5 in Chapter 4).
    - Automatically start the client applications, using the phone Push Registry.
    - Sign the MIDlet suite, since a special permission is required in order to access the SE.

The SE version has been partially developed in this thesis: JCP Applet, redesign of the application protocol, Push Registry issues, and interaction between the SE and the NFC reader. However, the feasibility of this version lies in the MIDlet suite signature by a code signing certificate. The acquirement process of this certificate has faced some administrative issues and, as a result, the SE version has not been completely developed.

The RMS version is the final version resulting of this thesis.

MIDP, CLDC, and the MIDlet construction steps are detailed in Appendix 5.

3.6.5.2. Communication.

Nfcip-java project [WNFCIPJ] consists of two Java libraries: J2SE and a J2ME-compliant respectively. These libraries facilitate the communication over NFCIP (Peer-to-peer) using ACS-ACR122U NFC readers and Nokia NFC-enabled phones. Version 1.3.1 is distributed under an open-source license.

The compatibility with the hardware devices as well as the selected programming language is the definitive factor to its choice.

3.6.5.3. Regular expressions.

Regexp-me project [WREGEXPME] offers a J2ME-compliant library aimed at checking regular expressions. It is distributed under an open-source license.

J2ME does not integrate any function to regular expressions checking. However, the UI has to deal with data input which has to meet determined patterns. For instance, the user login name follows a determined pattern (see Table A3.3 in Appendix 3).

3.6.5.4. Systems logs.

The system generates a log file with records about its activity. The system administrator inspects this log file to find the reason of any unexpected system behaviour (see Table A3.22 in Appendix 3). Besides, checking these logs is also a valuable tool in the development process to follow the system operation during the tests.

Log4j is an open-source Java library which allows the developer to control which log statements are output with arbitrary granularity, which is configurable at runtime using

external configuration files. Log4j is the more widespread logging Java API, with a gentle learning curve. The current library version, 1.2, is hosted as an Apache project [WLOG4J].

3.6.6. Code repository.

Apache Subversion (SVN) is a software versioning and a revision control system. It is widely used by the open-source community to maintain current and historical versions of files such as source code, web pages, and documentation. The SVN project is lead by the Apache Software Foundation [WASVN].

The development process in this thesis uses a SVN server to store the code repository. The selected server is VisualSVN Standard [WVSVNS] version 2.1.5, distributed free of charge.

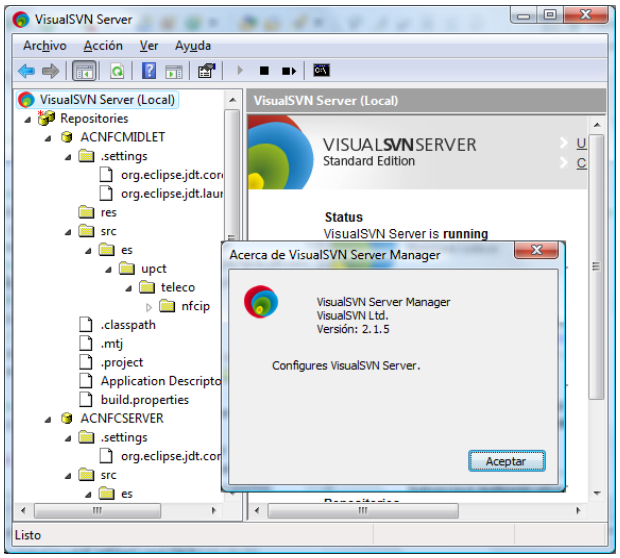


Figure 3.7: VisualSVN server.

Figure 3.7 depicts an image of the VisualSVN Standard server.

The selection of VisualSVN is based on two factors: *i)* the preferable usage of the standardised and widespread SVN technology, and *ii)* its compatibility with the operating system installed in the development computer: Windows Vista.

3.6.7. Selected technology.

Table 3.10 summarizes the complete list of software tools to be incorporated in the development process.

Selected Development Technology		
Type	**Purpose**	**HW Device / SW Tool**
Hardware	NFC reader	ACS-ACR122U
	NFC mobile phone	Nokia 6212 Classic
Software	LDAP server	OpenLDAP 2.4.23
	Database server	MySQL Standard Edition 5.5.8
	NFC server host	J2SE 6.23
	NFC-enabled device	J2ME 3.0 (factory-integrated)
	IDE	Eclipse Helios 3.6
	Phone emulator	Nokia 6212 NFC SDK
	APIS	J2ME: CLDC 1.1, MIDP 2.1 Nfcip-java 1.31 Regexp-me 1.0 Log4j 1.2
	Code repository	VisualSVN Standard 2.1.5

Table 3.10: Selected development technology.

Since the selected software tools are multi-platform, they can be installed and run on virtually any operating system. In consequence, they meet both non-functional requirements NFR–01 Production environment and NFR–02 Portability. Similarly, they meet the requirement NFR–04 Economic cost reduction, since all of them are distributed free of charge and/or under open-software licenses (except Nokia 6212 NFC SDK, copyrighted by Nokia). Both server and client software applications are coded with the Java programming language.

Chapter 4. SYSTEM DESIGN AND IMPLEMENTATION.

"Virtue not only signifies theoretical knowledge, besides, it is the ability of applying this knowledge into practical matters"

Socrates (469 BC – 399 BC)
Greek philosopher

This chapter addresses the solution domain of the platform, carrying out the design and implementation phases. The information generated by the analysis phase is used to perform several tasks of the design phase: UI, software architecture, external systems, application protocol, and test cases. Finally, the implementation phase is addressed using the results of the design phase and the developed platform is shown.

4.1. INTRODUCTION.

The traditional software development methodologies directly carry out the software architecture design after completion of the analysis phase, using its gathered software requirements. However, in those applications where the UI plays a crucial role, the software architecture design has to be postponed after the UI design [CJ2002].

The user-centred UI design is aimed at precisely defining how users interact with the application to perform their tasks [CJ2002]. User-centred UI design intentionally ignores the software functionalities design, taking into account only how the UI be perceived by end users. In this chapter, this user-centred UI design approach is used.

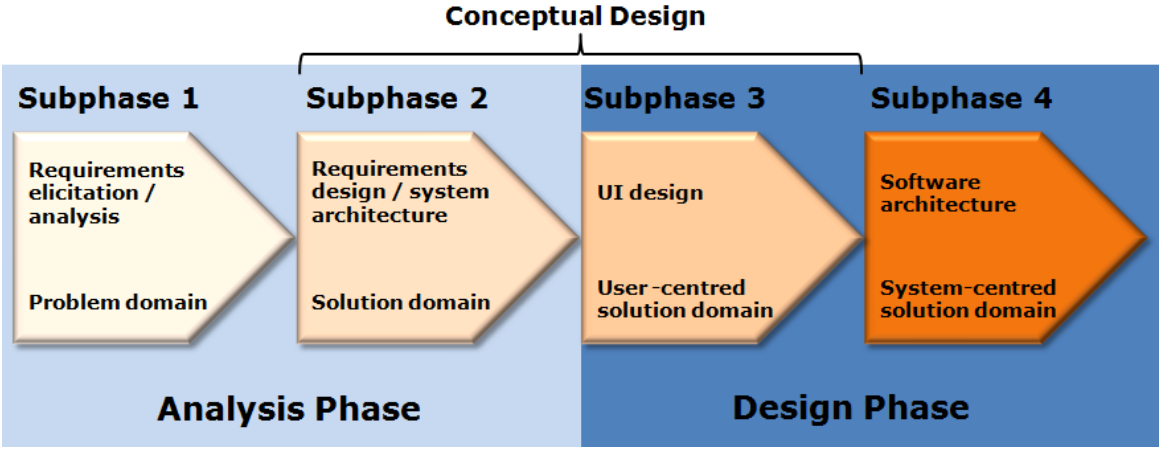


Figure 4.1: UI design location, from [CJ2002].

Figure 4.1 shows the correct UI design subphase location in the software development process.

The first task that must be carried out after the analysis phase is the UI design. The previous phase gathers a set of functional requirements that the system must meet, as well as a series of usability criteria used to get a more usable interface (see Chapter 3).

4.2. USER INTERFACE.

The UI design has to meet the non-functional requirement NFR–03 Usability (see Table A3.32), since the key feature in the NFC user access lies in a usable UI.

The usability model applied to address this requirement is based on ergonomic criteria, linking them with international standards (see Table 3.3 in Chapter 3). The following paragraphs detail the application of these criteria:

- Criteria related to ***Understandability***:
  - ***Legibility***: the UI favours the legibility by scalable type fonts. Besides, the UI is multilingual-ready: Spanish, English, and German. This feature is particularly important since the foreign teachers and students at the UPCT are possible users.
  - ***Prompting***: the UI helps its users to know which available actions they are able to perform, showing text descriptions with the different options they can choose. Besides, the UI shows the title of each application screen, so that the users know in which screen they are.
  - ***Significance of codes and behaviours***: means the understandability of codes showed by the UI. The designed UI standardises the use of representative icons which stand for unexpected behaviours: “warning” or “error”, as well as successful execution of actions: “info”. The UI, responding to actions performed by the user request, shows text codes which also follow homogeneous patterns.
- Criteria related to ***Learnability***:
  - ***Grouping***: options offered to the user are cohesively grouped regarding the task they perform. This allows the user to easily find and access to the required functionality. The UI menus group the functionalities in accordance to the tasks they performs.

- ***Minimal Action***: the UI reduces the number of steps that the user has to follow to perform a task. This feature is a common issue when using mobile devices where the keypad is the main input system, requiring the reduction of keystrokes. For instance, the UI stores the user's credentials and then it is able to automatically provide the server with them.

- ***Consistency***: the way information is presented to the user has to be consistent to allow the user to be quickly ready to use it. The UI screen design is homogeneous with a clear separation between action and information elements, allowing the user to easily learn and remember the application operation. The designed UI comprises the following screen patterns:

  - **"Action menu"**: shows the different actions that the user can perform, grouping the access to the different functionalities. The common actions for both the teacher and the student version are: *i)* "Configure User Information", and *ii)* "Configure Language". The specific student functionality is: "Register Attendance". The specific teacher functionalities are: *i)* "Activate Group", and *ii)* "List Attendees". The design of the "Main Menu" screen uses this pattern.

  - **"Information"**: shows related information to the specific user functionalities. This information comprises: *i)* status: "waiting", "running", "finished"; execution result and explanation: "success", "error"; and retrieved information. The design of the "Register Attendance", "Activate Group", and "List Attendees" screens use this pattern.

  - **"Edition"**: shows an edit formulary for storing and/or updating UI values. The design of the "Configure User Data" screen uses this pattern.

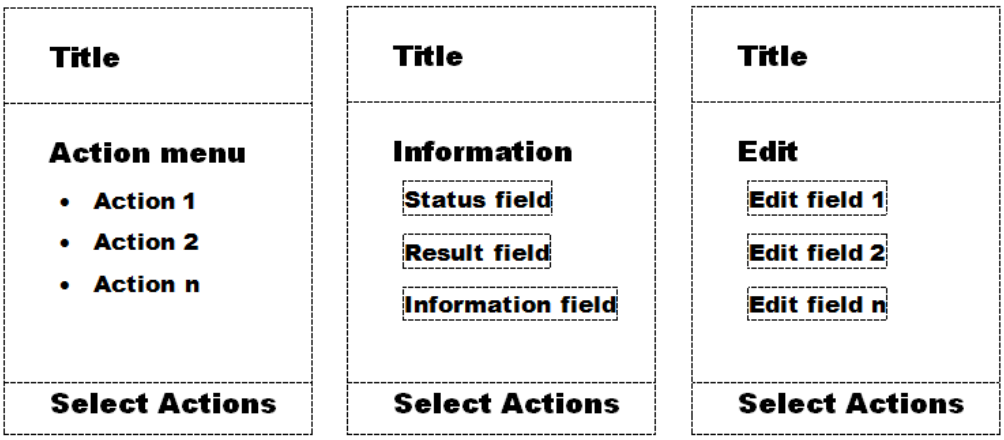


Figure 4.2: UI screen patterns design: action, information, and edition.

Figure 4.2 shows the schematic structure of the designed UI. The three screen patterns show common areas: *i)* the title, and *ii)* the selection area which gathers several auxiliary actions the user can select, such as "accept", "cancel", "select", "back", and "exit".

- ***Information Density***: one of the typical features of a mobile device is the reduced screen size. Thus, not giving the user a big amount of information is an important aspect to consider. In this sense, the UI shows concise information on those screens where the processing of a user request has been committed (registering the attendance, activating a group, or retrieving the attendance list).

- Criteria related to ***Operability***:

  - ***Flexibility***: means the UI ability to adapt itself to different execution environments, that is, different devices. The UI implemented can operate in different NFC-enabled devices, since no proprietary API is used (like Nokia's UI APIs).

  - ***Error protection***: the functionality requested is executed in a separate process (actually an execution thread), protecting the UI from undesired malfunctions. When this execution suffers any error, the UI shows a description of this. Besides, the UI tolerates external events which can occur during the normal operation, for instance, when the phone receives an incoming call. The UI operation pauses and recovers after the external event is attended by the phone user.

  - ***Quality of error messages***: error messages are checked during the unit test cases execution to guarantee the correct error description presentation.

  - ***Privacy policies***: access to the different functionalities is ruled by privacy policies. The student users are not able to access to the teacher functionalities (and vice-versa).

  - ***Accessibility***: guarantees access to the information and functionalities offered by a software application without any limitation regarding mental or physical disability. In this sense, the multilingual UI and the scalable type fonts are helpful features. Besides, the NFC "touching" paradigm allows a broad range of users to interact with the system by simply bringing the phone close to the reader.

4.3. SOFTWARE ARCHITECTURE.

XP, as well as Agile Methodologies in general, puts less emphasis on the software architecture than the classical methods (see Table A1.1 in Appendix 1). The XP development is driven by unit tests, and it is not focused on generating a big amount of deliverables, producing just the unavoidable ones.

Nevertheless, as Nord and Tomayko state [NT2006]: "including architecture-centric design and analysis methods in XP can help software developers address quality attributes in an explicit, methodical, engineering-principled way. Properly managed, architecture-centric methods can be a cost-effective addition to the software development process and will increase system and product quality".

Hofmeister separates software architecture into four views: *i)* **Conceptual**, *ii)* **Module**, *iii)* **Execution**, and *iv)* **Code** [HNS1999]:

- "The **conceptual** view describes the architecture in terms of domain elements. Here, the functional features of the system are designed".

- "The **module** view describes the decomposition of the software and its organization into layers. An important consideration here is limiting the impact of a change in external software or hardware".

- "The **execution** view is the run-time view of the system: it is the mapping of modules to run-time images, defining the communication among them, and assigning them to physical resources".

- "The **code** view captures how modules and interfaces in the module view are mapped to source files, and run-time images in the execution view are mapped to executable files".

The UML is used as a tool for modelling the software architecture in this thesis.

The UML class diagrams addresses the modelling of the conceptual view and the UML deployment diagrams fit well on modelling the module view of the software architecture. Execution and code views of the software architecture are models intended to improve the performance on large and complex systems, so are no needed in this project.

4.3.1. Conceptual view.

As stated by Ambler in [AS2004], the UML class diagrams are the main building blocks in OO modelling, and they show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and the attributes of the classes. The classes represent both the main objects in the application and the objects to be programmed.

In the UML class diagram the classes are represented with boxes which contain three parts, as stated in [AS2004]:

- "The upper part holds the name of the class".
- "The middle part contains the attributes of the class".
- "The bottom part gives the methods or operations the class can take or undertake".

During the design phase, a number of classes are identified and grouped together in a class diagram which helps to determine the relations between those objects.

Figure 4.3 depicts the detailed class diagram which serves as the software architecture conceptual view of the developed software platform.

An inheritance hierarchy with the class "User" is established, since there will be two distinct types of users in the system: "Student" and "Teacher".

The "User" class hierarchy implements the interfaces "GroupCheckable" and "Authenticable", meaning that every instance of the class hierarchy contains the methods defined in these interfaces.

A class containing a collection of others classes constitutes an aggregation association. For instance, a "Group" comprises one or more "Timetable" objects depending on the "laboratory" property.

The "Singleton" design pattern is used to design the class called "PracticesDB", indicated by the stereotype <<singleton>>. This will guarantee the existence of a single class instance in the system. When a database is queried, the most time-consuming task is opening connections. The use of this design pattern allows maintaining a single database connection opened at runtime which can be reused by successive queries. In general, the use of this design pattern on classes that manage database connections is a very good practice.

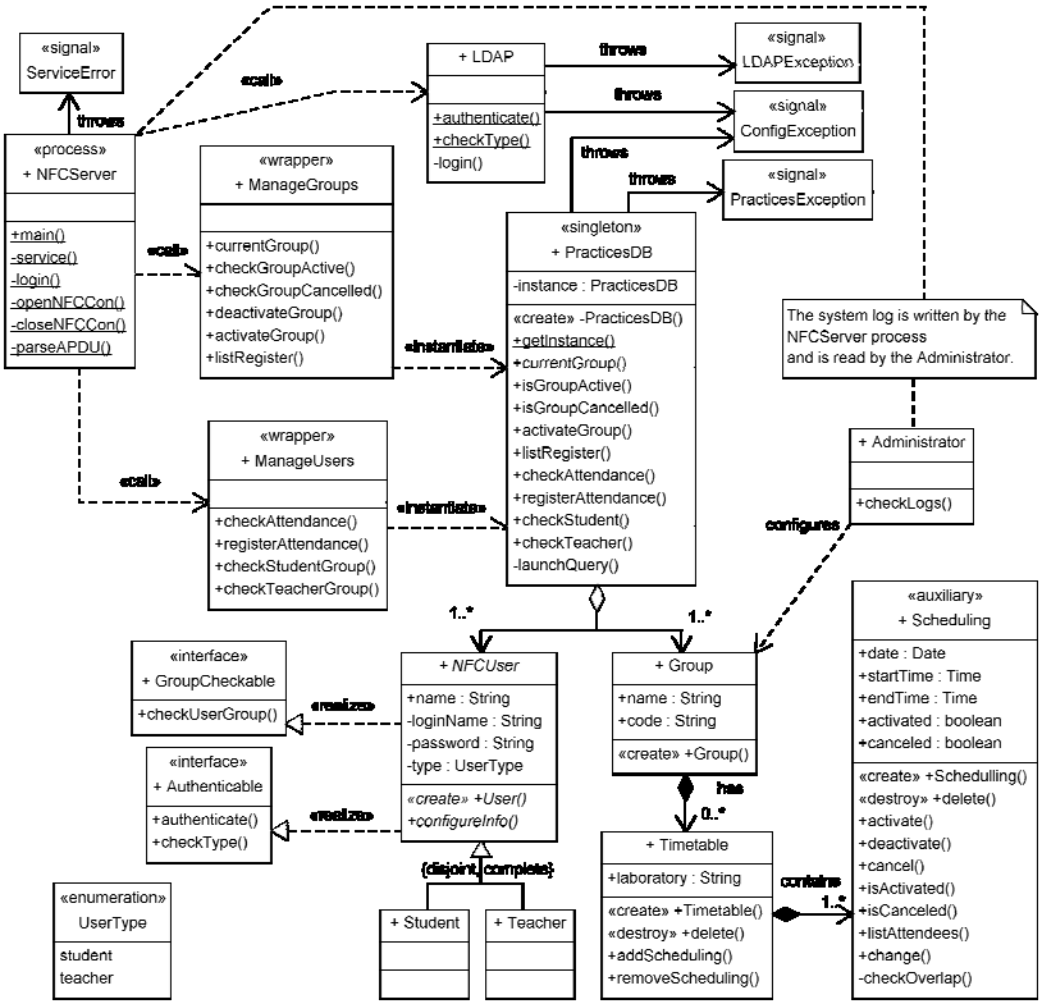


Figure 4.3: Software architecture conceptual view.

The diagram depicts several exception signals which can be thrown by the system when an error condition is found:

- "ConfigException" is thrown when the system finds any problem acquiring the proper configuration file or any of its parameters.

- "LDAPException" is thrown when the system finds any recoverable error condition dealing with the LDAP.

- "PracticesException" is thrown when the system finds any recoverable error condition dealing with the DBMS.

- "ServiceError" is thrown when the system finds an unrecoverable state, a fatal error.

4.3.2. Module view.

As stated by Ambler in [AS2005], "a deployment diagram depicts a static view of the run-time configuration of hardware nodes and the software components that run on those nodes". It can be glanced the required nodes and the software that is installed on it, as well as the middleware used to connect the nodes to one another.

A deployment diagram is useful to:

- "Explore the issues involved with installing the system into production".
- "Explore the dependencies that the system has with other systems that are currently in, or planned for, the production environment".
- "Depict a major deployment configuration of a business application".
- "Design the hardware and software configuration of an embedded system".
- "Depict the hardware/network infrastructure of an organization".

Figure 4.4 depicts the UML deployment diagram which serves as software architecture module view. There are five nodes involved with the system, in accordance with the designed system architecture (see Section 3.6 in Chapter 3):

1. The NFC device (or the mobile phone): it runs the client software and stores the user information.
2. The NFC reader: the hardware device which allows the communication between the client and the server.
3. The server: responsible for receiving commands by the clients and providing them with service.
4. The Database Management System (DBMS) server: stores the database with information about practices, timetables, teachers, and users.
5. The Lightweight Directory Access Protocol (LDAP) server: stores users' credentials and Organizational Units (OU).

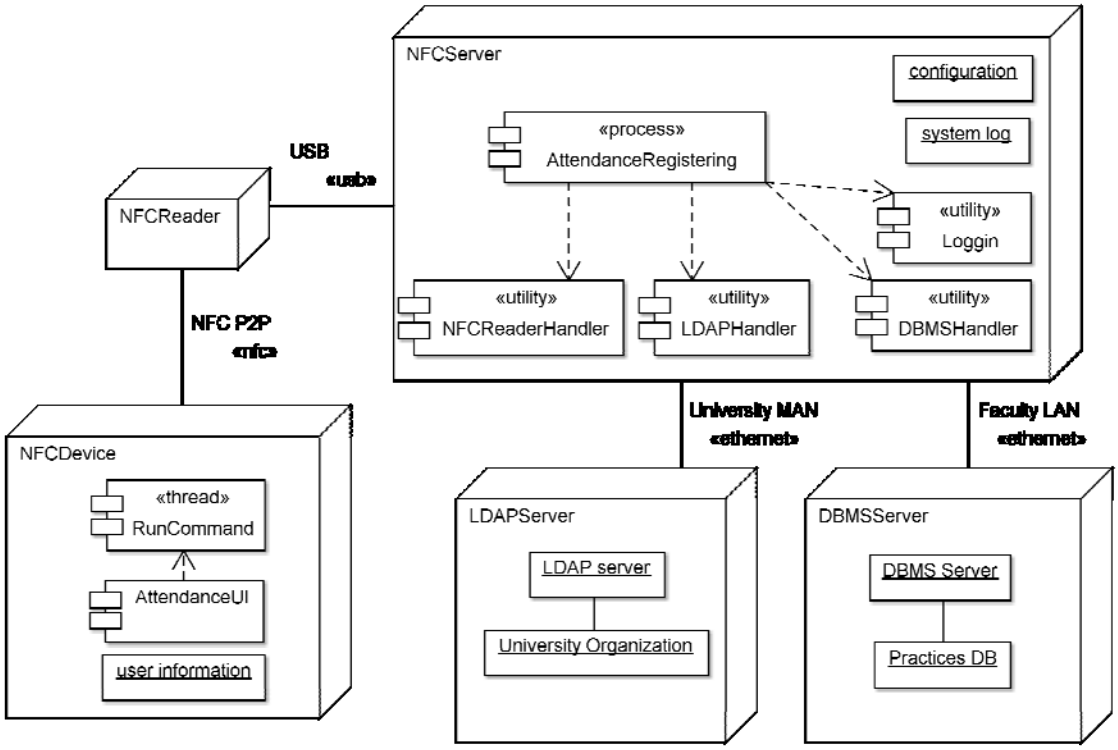


Figure 4.4: Software architecture module view.

The deployment diagram also shows the different existent associations between nodes, representing the network connections and hardware links established between them. For instance, every faculty at the UPCT has its own database dealing with teaching practices, so the network link is established at the faculty Local Area Network (LAN). However, the LDAP server is centralized and shared between the UPCT faculties, so eventually, the connection is established through the network backbone or Metropolitan Area Network (MAN) of the UPCT.

4.4. EXTERNAL SYSTEMS.

This section carries out the design of the external systems for testing purposes identified by the system architecture (see Section 3.6).

The two external systems present in the system architecture are: *i)* the database server which stores the practices database schema, and *ii)* the directory server which is responsible for managing the users' credentials at the UPCT.

4.4.1. Practices database design.

The Relational Model (RM) is used to design the practices database, since the RM is the most widely used model to database design [DC2003]. The RM generates a tabular representation of data in a simple and intuitive way. However, the RM theory is based on structuring the logical view of data around two mathematical concepts [WC2W]: *i)* domains, and *ii)* relations. "The name relational comes from 'relation' as known and widely used in mathematics, although in database theory the definition of relation is slightly extended".

- A domain is simply a set of values, together with its associated operators. It is equivalent to the notion of a programming data type.

- A relation over several domains is simply a subset of the Cartesian product. An element of the Cartesian set is called a tuple.

- A database is a collection of relations, together with the set of integrity constraints that the data must satisfy.

Figure 4.5 depicts the relational design for the practices database schema. The design is performed by means of MySQL Workbench software tool [WMYSQL], which is the most suitable tool for this purpose in accordance with the MySQL database.

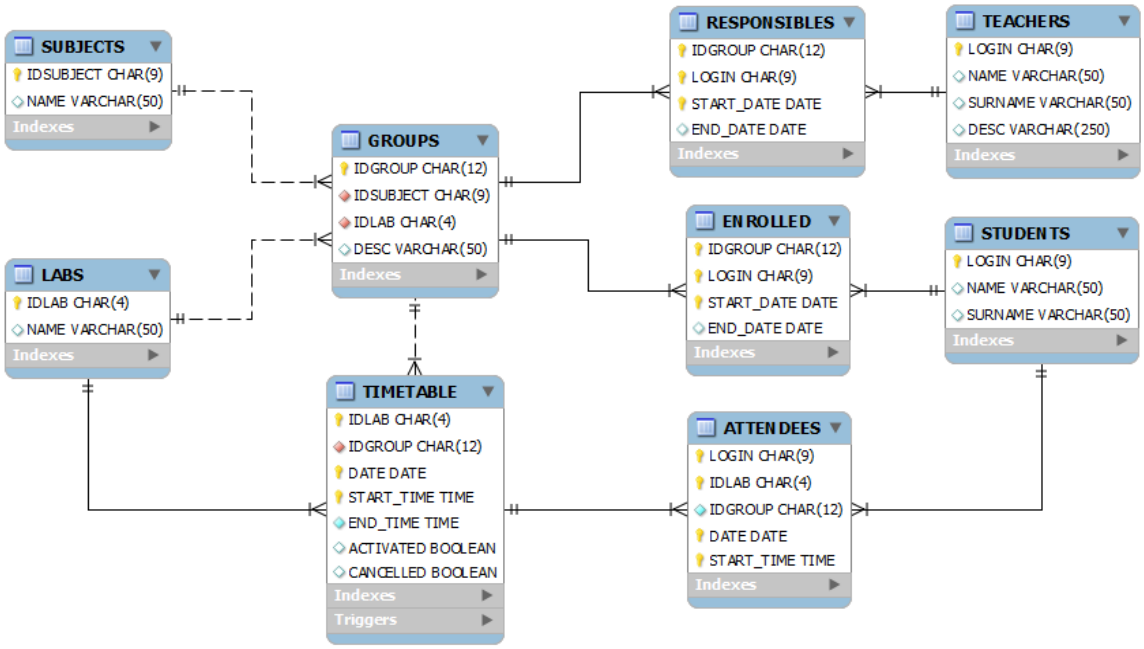


Figure 4.5: Designed practices database schema for testing purposes.

The designed database schema is rather comprehensible, as it is not very complex. It includes relations between subjects, students enrolled in them, and teachers responsible for lectures and practices sessions. Some points should be clarified:

- "Groups": classrooms and practices laboratories can have a reduced capacity compared to the number of students enrolled in a subject. Hence, the students should be assigned to different groups by the responsible teacher, thus improving the teaching performance. The group could constitute a class attending at lectures, or it can represent a practices group which attends at laboratory practical sessions.

- "Timetable": contains the planned sessions of each group, that is, its scheduling.

- "Lab": despite the name, it actually represents the set of laboratories or classrooms at the UPCT. A group is associated with a laboratory or classroom.

- "Attendees": contains the students who have attended at specific sessions defined in the timetable.

The practices database schema accurately corresponds to the information and functional requirements gathered during the analysis phase (see Appendix 3).

This database design for testing purposes is implemented using MySQL, as stated in the system architecture section (see Section 3.6 in Chapter 3).

It should be noted that this database schema design is aimed at define the actual configuration at the UPCT, but it is general enough to be adapted to other environments with no major efforts.

4.4.2. Directory schema.

The directory service of an organization is a shared information infrastructure which stores and manages information about common objects: printers, network elements, users' credentials, groups, devices, phone numbers, and other objects.

The directory is hierarchically structured in OUs which provides a way of classifying these objects.

Figure 4.6 shows a simplification of the UPCT directory. The domain "upct.es" is the root node which is structured in two OUs:

- “alumnos”: containing objects representing students.

- “usuarios”: containing objects defining teachers, university managers, and administrative staff.

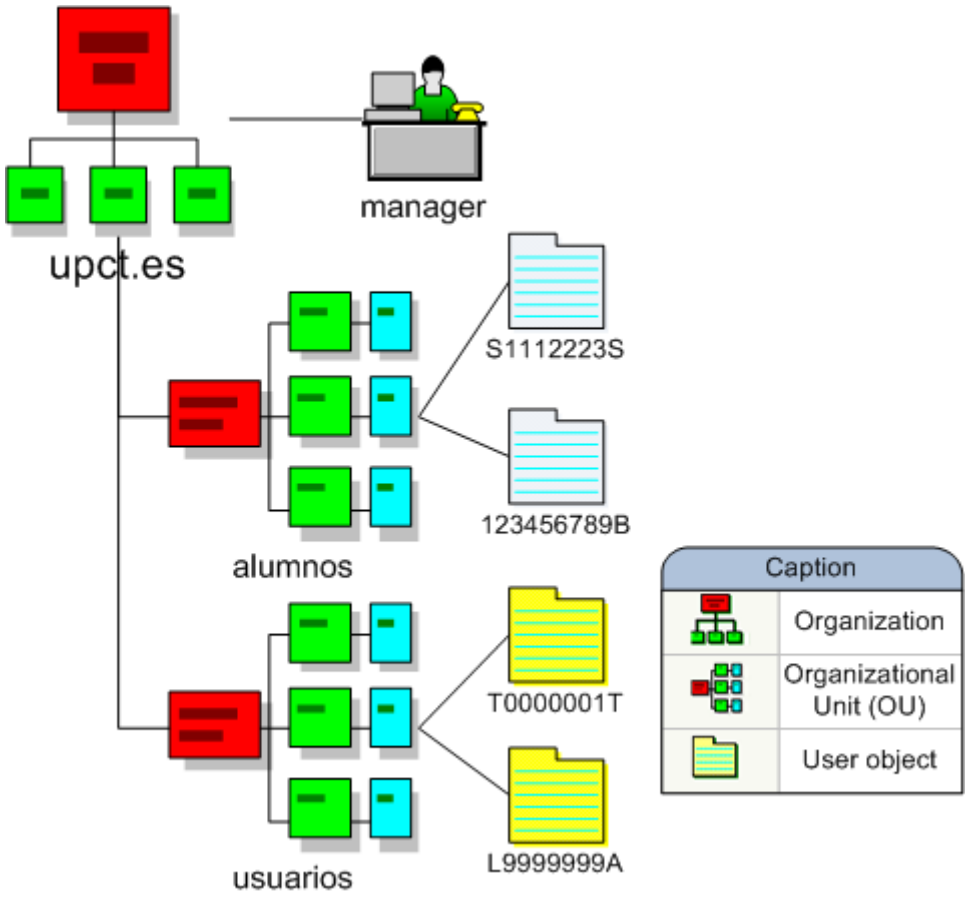


Figure 4.6: Designed LDAP schema for testing purposes.

The Seeded Signature Hash Algorithm (SSHA) [RFC2307] is used to store the users’ passwords, thus enhancing the security of the directory to meet the NFR–05 Security of private user data (see Section A3.4 in Appendix 3).

The directory administrator plays the role of “manager”.

This directory design for testing purposes is implemented using OpenLDAP, as stated in the system architecture section (see Section 3.6 in Chapter 3).

4.5. APPLICATION PROTOCOL.

This section shows the Application Protocol Data Unit (APDU) designed set for command interchange between client application and server. The Backus–Naür Form (BNF) notation is used to describe the communication protocol.

In IT, BNF is a notation technique for context-free grammars, often used to describe the syntax of languages used in computing, such as computer programming languages, document formats, instruction sets, and communication protocols. It is applied wherever exact descriptions of languages are needed [ML1986].

It is noteworthy that the proposed application protocol meets the functional requirements resulting of the analysis phase (see Appendix 3).

The two kinds of APDU defined are: *i)* Command APDU (CAPDU), sent from the client to the server; and *ii)* Response APDU (RAPDU), sent from the server to the client.

Command APDU
<CAPDU> ::= <COMMAND> <PAYLOAD> <COMMAND> ::= <CMD_STUDENT> | <CMD_TEACHER> | <CMD_ERROR> <PAYLOAD> ::= <USER> <PASSWORD> <CMD_STUDENT> ::= “ATTEND” <CMD_TEACHER> ::= “ACTIVA” | “LIST_A” <CMD_ERROR> ::= error ; empty or bad formed command <USER> ::= BYTE {9} <PASSWORD> ::= BYTE {4}

Table 4.1: CAPDU BNF specification.

Table 4.1 shows the CAPDU BNF specification. Each CAPDU contains user and password as parameters in the payload field.

- The student can only send the “ATTEND” command, intended to record the attendance.

- The teacher can activate a group, by means of the “ACTIVA” command, or requesting the attendee list using the “LIST_A” command.

Table 4.2 gathers the RAPDU BNF definition. The RAPDU is composed of three mandatory fields: RCODE (response code), RSUBCODE (response sub-code), and GROUP; plus an optional field only used when the attendance list is requested: LIST.

Response APDU
<RAPDU> ::= <RCODE> <RSUBCODE> <GROUP> [<LIST>] <RCODE> ::= <RCOD_STUDENT> | <RCOD_TEACHER> | <RCOD_ERRCMD> | <RCOD_SFAULT> | <RCOD_UNKERR> <RSUBCODE> ::= "LOGIN_OK_" | "ERROR_N_A" | "ERROR_N_G" | "ERROR_B_C" | "ERROR_G_W" | "ERROR_G_D" | "ERROR_G_C" | "ERROR_A_R" | "ERROR_A_A" | "SYS_ERROR" <RCOD_STUDENT> ::= "ATTEND_OK" | "ERR_ATTEN" <RCOD_TEACHER> ::= <RCOD_ACTIVA> | <RCOD_LIST_A> <RCOD_ERRCMD> ::= "ERROR_CMD" <RCOD_SFAULT> ::= "SYS_FAULT" <RCOD_UNKERR> ::= "UNK_ERROR" <RCOD_ACTIVA> ::= "ACTIVA_OK" | "ERR_ACTIV" <RCOD_LIST_A> ::= "LIST_OK__" | "ERR_LIST_" <GROUP> ::= BYTE {12} <LIST> ::= BYTE {512}

Table 4.2: RAPDU BNF specification.

APDUs Interchange Between Student Client and Server				
CAPDU (Client) →		← RAPDU (Server)		
COMMAND	**PAYLOAD**	**CODE**	**SUBCODE**	**GROUP**
"ATTEND" (attendance registering)	user and password	"ATTEND_OK" (attendance registering successful)	"LOGIN_OK" (login successful)	code of the current scheduled group
		"ERR_ATTEN" (error registering the attendance)	"USER NOT ALLOWED" | "BAD CREDENTIALS" | "WRONG GROUP" | "GROUP NOT ACTIVATED" | "GROUP CANCELLED" | "ALREADY REGISTERED"	
			"NO SCHEDULED GROUP" | "SYSTEM ERROR"	(*empty*)

Table 4.3: APDUs interchange between student client and server.

Each RAPDU is associated to the server processing of a CAPDU. Hence, there are two RCODE types direct associated to the user commands: RCOD_STUDENT and RCOD_TEACHER. The RCOD_ERRCMD is used when the incoming CAPDU contains a no recognized command. The RCOD_SFAULT indicates an unrecoverable system fault.

The RSUBCODE field adds information related to the RCODE field. The following tables define the possible RAPDU fields depending on the received CAPDU processing.

APDUs Interchange Between Teacher Client and Server				
CAPDU (Client) →		**← RAPDU (Server)**		
COMMAND	**PAYLOAD**	**CODE**	**SUBCODE**	**GROUP**
"ACTIVA" (activate group)	user and password	"ACTIVA_OK" (group activation successful)	"LOGIN_OK" (login successful)	code of the current scheduled group
		"ERR_ACTIV" (error activating the group)	"USER NOT ALLOWED" | "BAD CREDENTIALS" | "WRONG GROUP" | "GROUP CANCELLED" | "ALREADY ACTIVATED" |	
			"NO SCHEDULED GROUP" | "SYSTEM ERROR"	(*empty*)
"LIST_A" (list attendees)		"LIST_OK__" (attendance list retrieved successfully)	"LOGIN_OK" (login successful)	code of the current scheduled group
		"ERR_LIST_" (error retrieving the attendance list)	"USER NOT ALLOWED" | "BAD CREDENTIALS" | "WRONG GROUP" | "GROUP NOT ACTIVATED" | "GROUP CANCELLED" |	
			"NO SCHEDULED GROUP" | "SYSTEM ERROR"	(*empty*)

Table 4.4: APDUs interchange between teacher client and server.

Table 4.3 shows the APDUs interchange between the student client and the server. There are two possible RAPDU field configurations, depending on the processing success. If there is an error or no scheduled group, the GROUP field is empty.

Table 4.4 details the APDUs interchange between the teacher client and the server. There are several possible RAPDU field configurations, depending on the processing success. If there is an error or no scheduled group, the GROUP field is empty.

APDUs Interchange for System Errors				
CAPDU (Client) →		**← RAPDU (Server)**		
COMMAND	**PAYLOAD**	**CODE**	**SUBCODE**	**GROUP**
(*bad formed command*)	(*discarded*)	"ERROR_CMD" (incorrect or bad-formed command)	"SYSTEM ERROR"	(*empty*)
(*discarded*)	(*discarded*)	"SYS_FAULT" (system failure, unrecoverable)	"SYSTEM ERROR"	(*empty*)
(*discarded*)	(*discarded*)	"UNK_ERROR" (unexpected error, recoverable)	"SYSTEM ERROR"	(*empty*)

Table 4.5: APDUs interchange for system errors.

Table 4.5 shows the APDUs interchange between the server and a client when any error occurs. This interchange takes place when the server finds an error condition and sends the corresponding CODE to the client: *i)* the server receives an incorrect command, answering with "ERR_CMD"; *ii)* the server encounters an unrecoverable error, indicating it with "SYS_FAULT", or *iii)* the server encounters an unknown (not expected) error condition.

4.6. UNIT TEST CASES DESIGN.

This section details the unit test cases design which uses an adapted Black-Box Testing (BTT) technique. The test cases are aimed at driving the software development process, using the TDD technique in accordance to XP. A set of unit test cases has been designed, coded, and run for each functionality implemented.

As stated by Kolawa in [KH2007], "the goal of unit testing is to isolate each part of the program and to show that the individual parts are correct. A unit test provides a strict,

written contract that the piece of code must satisfy". The major benefit provided by unit testing is to help to discover problems in early stages of the development process, thus allowing the development team to fix them with less effort.

Using unit testing, the developers are able to refactor code at any time and, subsequently, to check whether a unit still works correctly or not. The procedure consists in designing and implementing test cases for all the functional requirements and, when a change causes a unit failure, is easy to identify it and fix the code [KH2007].

In addition, using unit testing perfectly fits the bottom-up testing approach, where the modules are tested separately in advance to be tested as a whole integrated system (integration tests) [KH2007].

The design and development of the unit test cases in this thesis has been performed using an adapted and simplified BBT technique, raising and running a battery of test cases that covers all system use cases or system functionalities.

4.6.1. Black-Box Testing technique.

The BBT technique treats the software as a "black box", with no knowledge about its internal implementation. BBT focuses on trying to find where the module does not conform to its specification. Thus, they are also called functional tests, where the tester is limited to provide data as input and to study the output without worrying about what could be done inside the module. This technique is particularly suitable for those modules that will serve as UI, but it does not lose its usefulness for any module of the system.

BBT is based on the requirement specification of the software modules. In fact, it speaks of "specification coverage" to give a measure of the number of requirements that have been tested. It is easy to achieve a hundred percent coverage with internal modules, but it can be more laborious with the outside interface modules. In any case, it is advisable to get high coverage with the outside modules.

The major issue with BBT does not often lie in the number of functions provided by the module (which is always a very limited number in reasonable designs), but in the data that are passed to these functions. In other words, the potential data set is usually very large (e.g.: an integer).

With the requirements of a module in mind, BBT follows an algebraic technique known as "equivalence classes". This technique treats each parameter as an algebraic model where some data are equivalent to others. If successful, from an excessively wide range of possible actual values, a small set of equivalence classes is obtained. Then it is

sufficient to prove a case of each class, since the other data in the same class are equivalent.

Hence, the work consists in identifying equivalence classes, a task for which there is no rule of universal application, but some recommendations for most practical cases can be followed:

- If an entry requires a specific data type, there are two equivalence classes: correct and incorrect data type.

- If an input parameter must fall within a certain range, there are three equivalence classes: below, at and above the range.

- If an entry requires a value within a set of values, there are two equivalence classes: inside and outside the set.

- If an entry is logical, there are two classes: 'yes' or 'no'.

- The same criteria apply to the expected outputs: producing results in every equivalence class must be tried.

- Having identified relevant equivalence classes in a module, one representative "normal" value of each class is chosen. A normal value cannot be a class boundary, but a random inside value acting as a common value which might appear at runtime.

However, experience shows that a great number of errors occur around the turning points of equivalence classes. There are a number of values called "frontier" (or "bounds") that should be proved in addition to those indicated in the previous paragraph. Usually, two frontier values have to be proved for each class: just below and above the class boundaries.

It should be noted that this thesis is not aimed at strictly applying of the BBT technique exposed in this section. Thus, the use of BBT has been done with some flexibility. For instance, all the data types or value ranges have not been checked, but the goal has been the monitoring of the proper run of the required functionalities and the observation of the exceptions or special cases when running them.

4.6.2. Extreme Programming unit testing.

TDD is the keystone of XP and uses unit testing, writing tests aimed at proving whether a software requirement is correctly implemented or not. A unit test might not pass

when it is deliberately aimed at finding a defect in a module, or if the functionality is not implemented yet. The more straightforward code has to be implemented to pass the test, thus lowering the complexity of the solution.

XP mandates a "test everything that can possibly break" strategy, over the traditional "test every execution path" method, developing of fewer tests than in classical methods [KH2007].

XP considers the test code as a first class project artifact which has to be maintained at the same quality as the implementation code: developers upload the unit testing code into the code repository in conjunction with the code it tests [KH2007].

With XP, the customer requirements are captured as user stories (descriptions of functionality) and development occurs in short cycles, called iterations (see Figure A1.2 in Appendix 1). Each iteration involves the building, testing, and delivery of customer-prioritised stories, followed by an evaluation discussion. Here, the customer can provide immediate feedback on desired features and estimates of their required effort. This entire means that the customer has more control over the process and, in consequence, the product itself.

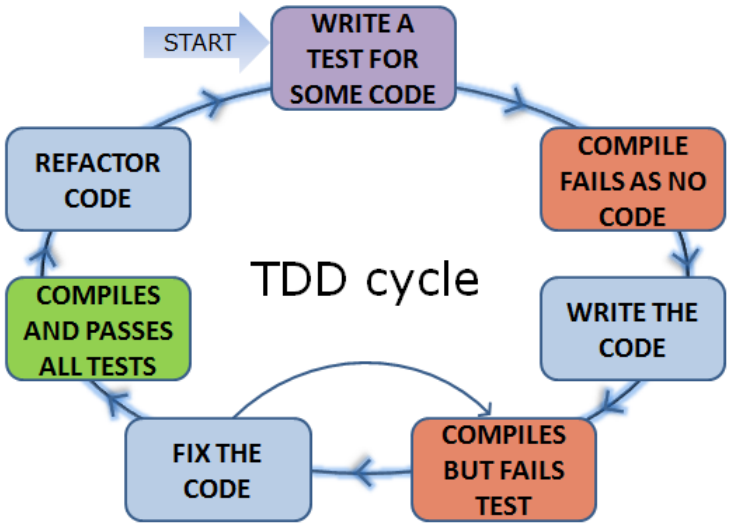


Figure 4.7: TDD cycle, from [SW2003].

XP uses the creation of unit tests for driving the development, the so-called TDD technique depicted in Figure 4.7. TDD forces developers to write the tests first, encouraging customers to think about how they want the software to work from the start. This will increase the quality and lowers the risk of spurious programming, the so-called "spaghetti code". Running the tests provides customers with feedback about the code in a short time.

4.6.3. Designed test suites.

The rule of thumb is to design a test case suite for each functional requirement, also known as use case (see Appendix 3). This allows the tester to check all the system functionalities in an isolated way before integrating them to constitute the whole software system.

As stated in Section 4.6.1, the selected design technique for unit testing is based on BBT, where the tester checks the behaviour of each unit by just providing data entries to it.

TC–01	**Unit Test Cases for UC–01 (Configure User Information).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–01 Configure user information.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Login name="T0000002T", password="0002" (teacher).
		Output:	Message showing the successful user configuration.
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1, with login name="".
		Reason:	Login name cannot be empty (CRQ–01).
		Output:	Error message showing the exception reason.
		Result:	Success.
	3	**Type:**	Exception.
		Values:	Same as test 1, with password="002".
		Reason:	Password length has to be at least four digits (CRQ–02).
		Output:	Error message showing the exception reason.
		Result:	Success.
	4	**Type:**	Exception.
		Values:	Same as test 1, with retyped password="0004".
		Reason:	Password and retyped password have to be equals.
		Output:	Error message showing the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	None.		

Table 4.6: Test case definition example.

The tester attempts to prove the correct execution of the unit under test, as well as the expected behaviour under the presence of errors, providing correct and incorrect entries. Thus, the designed test case suites are intended to check the proper implementation of the functionality (normal case) and to raise all the possible exceptions that the functionality could throw (exception cases).

Table 4.6 shows a test suite designed for a determined functional requirement of the system. The example test suite is composed by four test cases. First of all, the entries for the normal test case are designed to prove the proper execution of the functionality.

Following the normal test case, there are some other cases called as exception, where their values are aimed at making the functionality to fail in an expected way: *i)* throwing a determined exception, or *ii)* showing an error message or any other prearranged error condition. An exception test case is designed for each known exception condition. If another unexpected exception arises when running the test suite, a new exception test case is designed.

The full set of the designed battery of test cases is shown in Appendix 4.

4.6.4. Testing results.

Less-experienced software engineers tend to conclude that a successful test is that which has not found any bug or error. On the contrary, "*Absence of evidence is not evidence of absence*", as stated by the North American astronomer Carl Sagan [SC1997]. Hence, a successful test is that which has discovered more bugs and errors. The whole purpose of testing process is to discover as many bugs and errors as possible. The test considered more successful is that which covers more functionality and finds more errors in the software. In summary, "a successful test is one that finds a bug", as stated by Meyers [MG1979].

Due to the TDD technique (see Figure 4.7) followed by XP, all designed tests have been run successfully as long as all the functional requirements have been successfully developed. That is, during the earliest iterations, the tests have found a big amount of errors. This amount has been decreasing while the development, driven by the tests, has been progressing.

Designing tests on the functionality prior to the functionality itself, provides several benefits: *i)* customers always know what the most suitable business value is, focusing the development team efforts on it, and *ii)* developers are limited to successfully complete the implementation of tests, checking whether the functionality runs properly or not, and avoiding duplicity in the resulting code (see Figure 4.7).

4.7. DEVELOPED PLATFORM.

Once the implementation phase has been carried out, the developed platform is ready to further tasks stated in Chapter 1: *i)* commissioning testing, and *ii)* performing of some research about the application of NFC (see Chapter 1).

Figure 4.8 depicts the developed platform.

Figure 4.8: Developed hardware/software platform.

Figure 4.9: Registering the attendance.

The user's interaction with the platform is performed by just touching the NFC reader with an NFC-enabled mobile phone. The server software manages the reader and executes the incoming user's commands, opening connections with external servers, such as the LDAP server, or the DBMS server. Figure 4.9 depicts the most common use case of the platform: the attendance registering.

The NFC reader is an ACS-ACR122U, and the NFC-enabled phone is a Nokia 6212 Classic. Both devices are commercial models that can be directly purchased from their manufacturers (see Section 3.6). The implementation of the server software is carried out by means of Java programming language and java libraries.

The following section describes the UI and the functionalities of the developed software in the thesis. Screenshots of the implemented application are used to enhance the understandability of the explanations. Moreover, in order to guarantee traceability of the system functional requirements, listed in Appendix 3, the screenshots will be associated with their related requirements.

4.7.1. Developed User Interface.

The system interacts with three human actors: student, teacher, and administrator (see Section A3.3.2 in Appendix 3). Both teacher and student interaction with the system is based on NFC mobile phones, as stated in Section 3.6 in Chapter 3, but these actors carry out different functionalities. Hence, the usable UI design presents three different versions, depending on the user they are aimed at (see Section 4.1).

Figure 4.10: UI Main menu.

Figure 4.10 depicts the UI main menu, which follows an "action menu" screen pattern (see Section 4.1). Student and teacher versions present common functionalities:

- "Configure user information": this UI functionality implements the functional requirement UC–01 Configure user information (see Table A3.14).

- "Configure language": this functionality is aimed at improving the UI Usability (see Section 4.1).

The following sections present the three UI versions.

4.7.1.1. Student version.

Figure 4.11 depicts the student UI main menu. In addition to the common functionalities, the UI shows another: "Register Attendance", which implements the functional requirement UC–02 Register attendance (see Table A3.15 in Appendix 3).

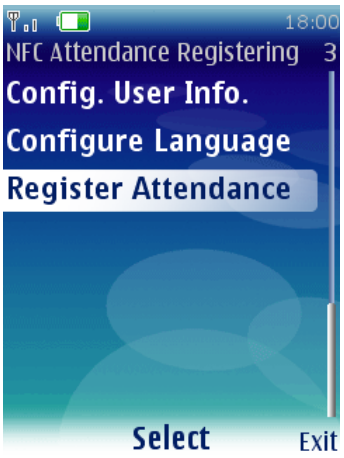


Figure 4.11: Student version.

4.7.1.2. Teacher version.

Figure 4.12 depicts the teacher UI main menu.

In the teacher version two new UI functionalities are added to the common ones:

- "Activate Group" implements the functional requirement UC–03 Activate group (see Table A3.16 in Appendix 3).

- "List attendees" implements the functional requirement UC–04 List attendees (see Table A3.17 in Appendix 3).

Figure 4.12: Teacher version.

4.7.1.3. Administrator version.

The administrator UI functionalities are provided by means of MySQL Workbench [WMYSQL].

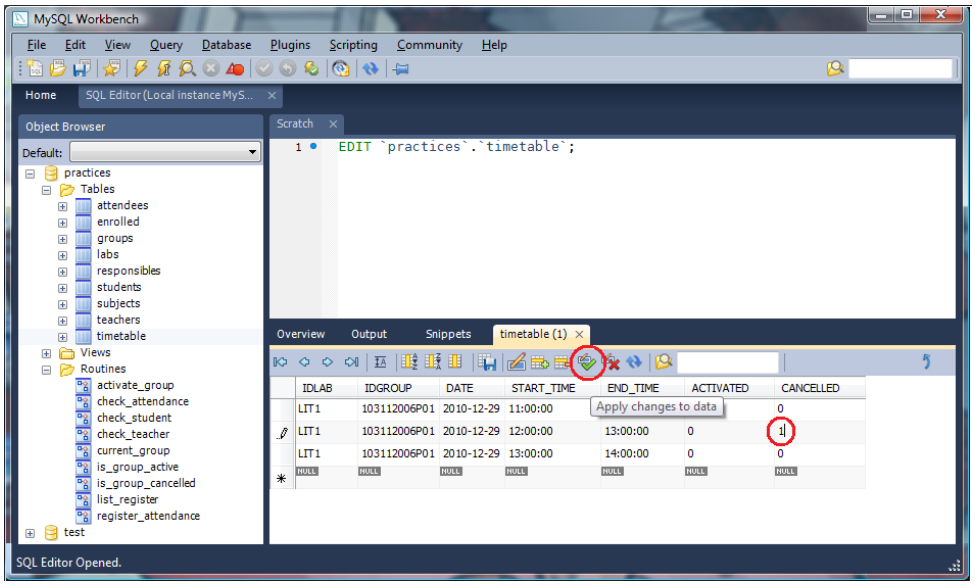


Figure 4.13: Administrator UI.

Figure 4.13 depicts an example of the administrator UI. Here, the functionality provided by the UI corresponds to the cancellation of a group session scheduled in advance, by means of updating the "cancelled" field to 1 (boolean "true"). Red circles highlight the important areas. The functional requirement implemented is UC–08 Cancel group (see Table A3.21 in Appendix 3).

The complete list of the UI functionalities is described in Appendix 6.

4.7.2. Server configuration.

Table 4.7 gathers the NFC server configuration file, which parameters have to be properly set while deploying the server software. Once the configuration is done, and the communication with the external systems is checked (LDAP and DBMS servers), the NFC server is ready to be tested or to be put into production.

NFC Server Host Configuration File

```
########################################
#  NFC SERVER HOST CONFIGURATION FILE  #
########################################
# Author: Alfonso De Gea. Version: 1.0
# File name: host-config.properties
########################################
#     Practices database parameters    #
########################################
# Database driver
dbpractices.driver=com.mysql.jdbc.Driver
# MySQL JDBC connection string: protocol + "://" + host + ":" + port + "/" + db
dbpractices.connectionString=jdbc:mysql://localhost:3306/practices
# Database user
dbpractices.dbuser=practices
# Database password
dbpractices.dbpass=practices
########################################
#     LDAP parameters                  #
########################################
# LDAP server URL: protocol + "://" + host + ":" + port
ldap.providerURL=ldap://localhost:389
# LDAP organization
ldap.organization=dc=upct,dc=es
# Teachers OU
ldap.usersOU=ou=Usuarios
# Students OU
ldap.studentsOU=ou=Alumnos
########################################
#     Server parameters                #
########################################
# NFC terminal number (reader). Allowed values: 0..10
server.nfc.terminal=0
# Dependency (laboratory, classroom) where the server is deployed.
# Cannot be missed, since it produces an unrecoverable system error.
server.nfc.dependency=LIT1
# Maximum user request attempts when an external system is not responding.
# When this number is reached, the server aborts its operation.
# Allowed values: 1..500
server.nfc.maxuserattempts=50
# Maximum server attempts to connect to the reader.
# When this number is reached, the server aborts its operation.
# Allowed values: 1..100
server.nfc.maxconnectreader=10
```

Table 4.7: NFC server host configuration file.

The configuration file is auto-documented and structured in three sections: *i)* Practices database parameters, *ii)* LDAP parameters, and *iii)* Server parameters. The parameters of sections i) and ii) cannot be missed, since this might produce an unrecoverable system error, aborting the server operation. The dependency in which the system is deployed is also a mandatory parameter. When any of the rest of the

configuration parameters is missed or incorrectly assigned, the system assumes default values and continues the operation.

Chapter 5. CONCLUSIONS AND FURTHER WORKS.

"Only those who contribute to the future have the right to judge the past"

Friedrich Nietzsche (1844-1900)
German philosopher

The present chapter details the conclusions from the results achieved in the thesis, as well as future works that can be addressed in relation to these results.

The European Higher Education Area guidelines are promoting the modernization of the universities all over Europe by means of the implementation of new technologies. This modernization will allow the services offered by universities to be closer to society.

With this idea in mind, the Technical University of Cartagena is deeply involved in the implementation of the so-called "Smart University" model, promoting an ambitious project on the study of Near Field Communication technology and its application at the university environment.

This thesis, being a part of this project, has developed a hardware/software platform intended to constitute a basis for further scientific research at the Technical University of Cartagena dealing with the impact on the university society.

The platform resulting from this thesis, using as basis Near Field Communication, constitutes a ubiquitous computing platform for attendance registering at universities.

The main objectives of this platform are:

- To serve as a pilot testing tool intended for the planning of a possible large-scale deployment at university facilities.

- To constitute a methodological and technological base framework for future development processes dealing with Near Field Communication at the university.

First of all, a study of Near Field Communication technology has been performed in this thesis, discussing several aspects such as: main characteristics, standards and specifications, comparison with other short-range communication technologies, application scenarios, and security issues.

From the development process viewpoint, the global functionality achieved by the platform can be stated as follows: allowing the effective monitoring of student attendance at lectures or laboratory practices just by bringing a mobile device close to a Near Field Communication reader placed at the main entrance of the dependencies.

The usage of open-source (or free of charge) tools has been a non-functional requirement applied in the software development in this thesis. This requirement has served to reduce the economic cost associated to the development process.

The resulting hardware/software platform developed in this thesis implies the achievement of several technological results:

- Hardware technology (commercial models):
    - ACS-ACR122U NFC reader.
    - Nokia 6212 Classic NFC-enabled mobile phone.
- Software technology (open-source or free of charge):
    - Java programming language: Java 2 Standard Edition, Java 2 Micro Edition.
    - Applications Programming Interfaces and study of their application: CLDC, MIDP.
    - Java libraries: nfcip-java, regexp-me, log4j.
    - Integrated development environment: Eclipse Helios.
    - Device emulator: Nokia 6212 SDK.
    - Code repository: VisualSVN.
- Server software tools for modeling external systems at the university (for testing purposes):
    - Database server: MySQL.
    - Directory server: OpenLDAP.

This thesis has explored two possible versions of the platform:

- The Record Management System version, storing private data in phone internal records.

- The Secure Element version that uses the phone internal smart card as data store.

The Record Management System version has been carried out, and the Secure Element version has been partially developed: Java Card Applet, redesign of the application protocol, Push Registry issues, and interaction between the Secure Element and the Near Field Communication reader. However, the feasibility of this latter version lies in the MIDlet suite signature by a code signing certificate. The acquirement process of this certificate has faced some administrative issues and, as a result, the Secure Element version has not been completely developed.

The development process in this thesis has been addressed using a software engineering approach, splitting up the development tasks in several distinguished phases: analysis, design, implementation, and testing. Unified Modeling Language has been used to produce the required models during the analysis and design phases: use cases diagrams, class diagrams, deployment diagrams.

In this thesis, the determination and the application of the most suitable software development methodology has been performed: Extreme Programming. This methodology is aimed at those projects with small development teams, volatile requirements, and based on new technologies; hence, this methodology has fitted perfectly to address the development process in this thesis.

During the analysis phase, the systems requirements have been obtained using the requirements elicitation methodology proposed by Amador Durán. The definition of the functional requirements has implied the design of sixteen test use cases using a black-box testing technique. Then, these test use cases have been used by the Test-Driven Development technique proposed by Extreme Programming to address the Java code implementation.

During the implementation phase, the designed test use cases have serve to check the proper execution of the functionalities, as well as their expected behaviour in presence of error conditions. The execution of the test cases has thrown a hundred percent of successful test results. Thus, all the functionalities has been developed and successfully checked.

In addition, this thesis has taken into account the application of Usability during the development process to achieve a suitable user experience by means of the application of a

usability model. The usability model proposed by Francisco Montero has been applied to the development.

Furthermore, the development of this thesis has involved the completion of another task implicitly but particularly relevant: the proper application of those theoretical and practical concepts acquired during the studies of Telematics Engineering.

The completion of the Secure Element version constitutes a future task to be performed. This completion has been procrastinated and will be achieved once that a code signing signature be acquired.

Upcoming projects dealing with the Near Field Communication technology implementation at the university will use the results achieved in this thesis:

- Access control to university dependencies (offices, classrooms, and laboratories).
- Management of loans of didactic resources (books, software).
- Payment of administrative fees (certificate issuing, matriculation charges).

The selected technology set constitutes a technological base framework to those future projects, establishing the basic hardware devices and software tools required to address the development process.

The methodologies and development techniques used in this thesis are: Extreme Programming, Test-Driven Development, black-box testing, and a usability model. Their determination and application in this thesis constitute a guideline for future theses with similar characteristics: individual, based on new technologies, and including a software development process. Hence, the methodological results achieved in this thesis constitute a base framework for this kind of projects.

Besides, the developed platform in this thesis will serve as a pilot testing tool intended to the planning of a possible large-scale deployment at university facilities. As a future work, during the testing and commissioning phase, several parameters will be studied such as response time, technology performance, gains in terms of time and money saving, and impact on the university members.

Regarding the security, the Near Field Communication standard provides some features intended to prevent some attacks, such as Man-in-the-middle, and data sniffing. However, the security has to be improved by using dedicated cryptography to establish a

secure channel between communicating devices. This secure channel will play a paramount role especially in future developments related to ticketing or mobile banking, since these applications deal with extremely sensitive user data.

In summary, this thesis is the pioneer project of a series of scientific research and development projects at the Technical University of Cartagena, constituting one first step to achieve the satisfactory implementation of the “Smart University” model.

Appendix 1. AGILE DEVELOPMENT AND EXTREME PROGRAMMING.

The goal of this appendix is to support the selection of the most suitable methodology to be used during the development process of this thesis. This appendix discusses, in a summarized way, the context in which Agile Methodologies have emerged, their values, and principles. Furthermore, a comparison with classical methodologies is presented. In addition, a brief description of the Extreme Programming proposal is attached, which is the most popular agile methodology nowadays.

A1.1. INTRODUCTION TO AGILE METHODOLOGIES.

Software development is not an easy task. A proof is the existence of numerous methodological proposals that affect different dimensions in the development process.

There are classical proposals that especially focus on the process control, by rigorously establishing the involved activities, the artifacts that must be produced, and the tools and notes that will be used. These proposals have proved to be effective and useful for a great number of projects, yet for many others some problems have arisen.

A possible methodological improvement can be made by including more activities, artifacts and restrictions to the developing processes on the grounds of the weak points that have been detected. Nevertheless, the final result would turn into a more complex development process that may even limit the ability of a development team to carry out a project.

Another approach would be to focus on other dimensions, such as the human factor or the software product. This is the philosophy of Agile Methodologies, which give more importance to the individual, the collaboration with the customer and the incremental development of the software with short iterations. This approach has been proved effective in projects with fast-changing requirements or when it is necessary to drastically reduce the development times while keeping up-to-standard quality [CLP2005].

During the last decades of the twentieth century, Software Engineering performed several modelling notations and software tools as the “silver bullet” to guarantee success in software development [BF1995]. However, this expectation was never reached. This was due to the deferral of another important element: the development methodology itself. Good notations and tools are useless without guidelines on how to apply them [CLP2005].

Consequently, the beginning of the twenty first century brought with it a growing interest in development methodologies. Until recently, development processes emphasized on the process itself by means of: a thorough definition of roles, activities and artifacts; and a great, detailed production of models and deliverables. This classical outline to tackle the software development proved to be effective and necessary for projects of great magnitude (in terms of time and resources), where a greatly elaborated process is required. Nonetheless, this approach is not the most appropriate one for many current projects, where system environments are very changeable or when it is necessary to drastically reduce the development iterations while keeping up-to-standard quality [CLP2005].

In view of the difficulties to use classical methodologies with these time and flexibility restrictions, many development teams resign themselves to dispense with the efficient working of Software Engineering, assuming the risk which it entails. In this scenario, Agile Methodologies emerge as a possible answer to fill this methodological gap.

The environments for which Agile Methodologies have been especially aimed at are in keeping with a great range of industrial projects on software development: those with small development teams, with short deadlines, volatile requirements, and/or based on new technologies. Agile Methodologies provide a simplification which does not lower the standards of the essential practises of the software development [WKL2005].

This appendix is organised as follows: in Section A1.2 the main characteristics of Agile Methodologies, included in the "Agile Manifesto" [WAGILMAN], are introduced. Likewise, a comparison between these ones and classical methodologies is established. Section A1.3 focuses on the Extreme Programming methodology, introducing its main features, the procedure to be followed and the practices suggested. Lastly, in Section A1.4, some final considerations on Agile Methodologies are presented, focusing on Extreme Programming.

A1.2. AGILE METHODOLOGIES.

In a meeting held in Esset (USA) in February, 2001, the term "agile" applied to software development was born. Seventeen software developers took part of this meeting, including some of the creators and the driving-forces of software methodologies. This meeting had as an end to outline the values and principles that should allow teams to develop software quickly or to respond to changes that may arise throughout a project. Their intention was to offer an alternative to classic software development processes known for being rigid and led by the documentation generated in each of the development activities.

Following this meeting, "Agile Alliance" [WAGILEA] was created. It is a non-profit organization that promotes concepts related to the agile software development as well as to help other organizations to adopt these concepts. The starting point was the "Agile Manifesto" [WAGILMAN], a document which summarizes the principles of Agile Methodologies.

A1.2.1. The Agile Manifesto.

This Manifesto has come to value:

- **Individuals and interactions over processes and tools.** Individuals are the main success factor. It is more important to create a good team than a good environment. More often than not, environments are erroneously built up first in the hope that the team will automatically adapt to it. It is far better to create the team and that this one configures their own development environment based on their needs.

- **Working software over comprehensive documentation.** The rule of thumb is not to produce documents unless they are indispensable to take an important decision immediately. These documents have to be short and to focus on the essential.

- **Customer collaboration over contract negotiation.** The aim is the constant interaction between customer and development team. It is this collaboration that will beat the rhythm and guarantees success.

- **Responding to change over following a plan.** The ability to respond to changes that may arise along the project (changes in the requirements, the methodology, or the team) likewise determines its success or failure. Therefore planning should not be strict but open and flexible.

The previous values inspire the twelve principles in the Manifesto. They comprise characteristics that distinguish an agile process from a classical one. The first two principles are general and summarise a great deal of the agile spirit. The rest are related to the procedure to be followed and the development teamwork, as far as aims and organization are concerned. These principles are:

i. Priority is given to customer satisfaction by rapid and continuous delivery of useful software with a value added.

ii. Welcome changing requirements. Changes are captured for the customer to have competitive advantage.

iii. Working software is delivered frequently with short time intervals between deliveries (from two weeks to two months).

iv. Close daily cooperation between customers and developers throughout the project.

v. Projects are built around motivated individuals. They should be provided with the environment and support that they need and be trusted to accomplish their tasks.

vi. Face-to-face conversation is the most effective and efficient form of communicating information within a development team.

vii. Working software is the principal measure of progress.

viii. Agile processes promote sustainable development. Promoters, developers and users should be able get on well with each other.

ix. Continuous attention, technical quality and good design improve agility.

x. Simplicity is essential.

xi. The best architectures, requirements and designs emerge from self-organizing teams.

xii. In regular intervals the team reflects upon their effectiveness and adapts to changing circumstances accordingly.

A1.2.2. Comparison between agile and classical methodologies.

Table A1.1 gathers the main differences of Agile Methodologies in relation to the classical ones (non-agile, traditional or heavyweight methodologies). These differences do not only affect the process itself but also the context of the team together with its organization.

Agile Methodologies	Classical Methodologies
Based on heuristics resulting from code production practices	Based on commonly used standards by the developers' environment
Especially prepared for changes during the project	Higher efforts to address changes
Internally imposed (by the team)	Externally imposed
Less controlled process, with few principles	More controlled process, with numerous policies/rules
Absence of a traditional contract, and, if any, it is rather flexible	Presence of a prearranged contract
The customer is part of the development team	The customer interacts with the development team through meetings
Small groups (less than 10 members), working in the same location	Numerous groups that are possibly distributed
Fewer artifacts	More artifacts
Fewer roles	More roles
Less emphasis on software architecture	The software architecture is essential and it is expressed through models

Table A1.1: Agile versus classical methods.

A1.3. EXTREME PROGRAMMING.

The concept of Extreme Programming (XP) [WDEFXP] [BK2000] defines an agile methodology focussed on promoting interpersonal relations as the key to success in software development by promoting teamwork, caring about developers' learning and fostering a good working atmosphere.

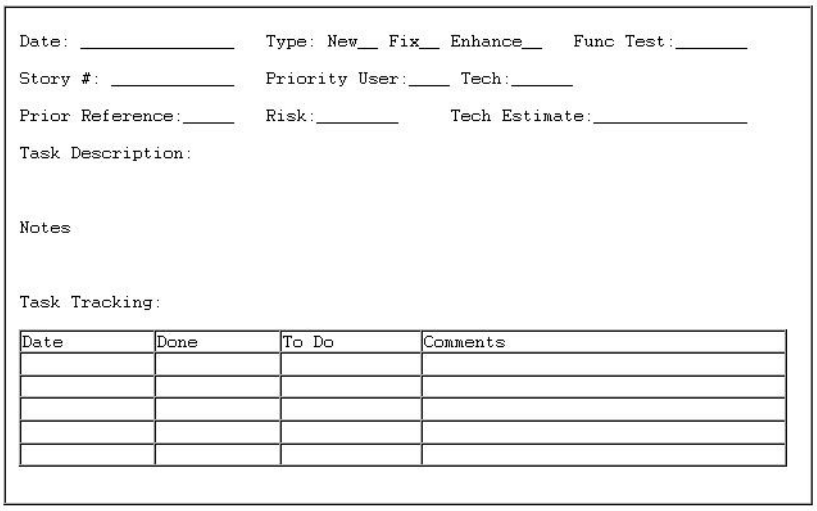
XP is based on continuous feedback between the customer and the development team, constant and current communication among all members, simplicity in implemented solutions and courage to face changes. XP is defined as specifically appropriate for projects with imprecise or volatile requirements with high technical risk.

The principles and practices are common sense, but taken to the extreme; hence its name. Kent Beck, the father of XP, describes the XP philosophy in [BK2000] without covering technical or implementation details. Subsequently, other experiences publications took on the responsibility for such a task.

The essential characteristics of XP will be presented in the following four sections: *i)* User stories, *ii)* Roles, *iii)* Process, and *iv)* Practices.

A1.3.1. User stories.

User stories comprise the technique used to specify software requirements. They consist in paper cards on which the customer briefly describes the characteristics the system must possess, either functional or non-functional requirements. The management of user stories is very dynamic and flexible. Each user story is sufficiently comprehensible, defined and with a limited scope for programming staff to implement it in a few weeks.


Date: _____ Type: New__ Fix__ Enhance__ Func Test:_____

Story #: _____ Priority User:____ Tech:_____

Prior Reference:_____ Risk:_____ Tech Estimate:_____

Task Description:

Notes

Task Tracking:

Date	Done	To Do	Comments

Figure A1.1: Customer story and task card, from [BK2000].

In his book [BK2000], Beck introduces a card model: the customer story and task card (depicted in Figure A1.1) on which the following contents can be identified: date, activity type (new, fix, enhance), functional test, story number, customer's and technical estimates about priority, description, notes, scheduling list with the date, status, features under development and comments.

The estimates of a user story may take from one to three weeks of planning time (in order not to surpass the iteration time). User stories are divided and presented on task cards and assigned to programmers to be implemented during the current iteration.

A1.3.2. Extreme Programming roles.

According to Beck's original proposal, the XP roles are the following:

- **Programmer**: writes the unitary tests and produces the system code.

- **Customer**: writes the user stories and the functional tests to validate their implementation. Furthermore, it assigns priority to user stories and decides which ones are implemented with the aim of contributing more value to the business.

- **Tester**: helps the customer to write functional tests. This person runs the tests regularly, spreads results among the team and is in charge of support tools for testing.

- **Tracker**: provides the team with feedback. This member verifies the degree of accuracy among the estimates carried out and the real time devoted in order to improve future estimates. This role conducts the task tracking during each iteration.

- **Coach**: is responsible for the global process. This role provides the team with guidelines for the XP practices to be applied and for the process to be appropriately executed.

- **Consultant**: is an external member of the team with specific knowledge on an essential and possibly problematic subject to the project.

- **Big Boss**: acts as link between customers and programmers. This person helps the team to work effectively by creating adequate working conditions. The essential work of this role is to deal with coordination.

A1.3.3. Extreme Programming process.

The life-cycle of the XP process consists, in outline, in the following steps:

1. Customer defines the value of the business to be implemented.

2. Programmer estimates the necessary effort for its implementation.

3. Customer selects what to build, according to their own priorities and time restrictions.

4. Programmer builds the selected business value, driven by the unit tests.

5. The acceptance tests are run against the user requirements.

6. Restart on step 1.

Both Customer and Programmer learn in each of the iterations of this cycle. No more working pressure than the estimated (in the steps 2 and 3) must be put on Programmer since software quality may be lowered or deadlines may be impossible to meet. Likewise, Customer has an obligation to manage the product delivery to make sure that the business has the highest possible value as a result of each iteration.

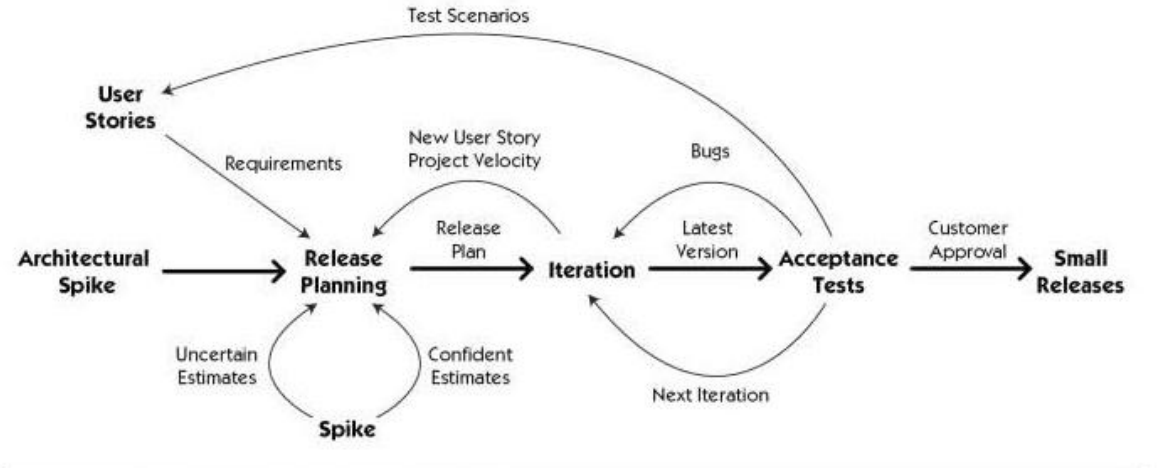


Figure A1.2: XP development process, from [WJ2001].

On the left side of the graphic depicted in Figure A1.2, the two elements that drive release planning and development are shown.

- User stories represent functionality to be implemented in the course of the release.

- An architectural spike is any required task that the team has to execute in order to lay in some architectural foundation, to explore a potential refactoring, or to look at new technology that may need to be included in the release. These inputs drive the release planning session.

The outcome of the release session is an iteration plan defining a set of iterations intended to accomplish the release.

To the right of Figure A1.2, and integral to the iteration, the ever-present acceptance tests are depicted, which are typically written by the customer and serve to test the functionality implemented against the user stories.

Finally, the result of the whole process is a series of small releases that rapidly evolve to address Customer needs.

The ideal XP life-cycle consists of six phases [BK2000]: *i)* Exploration, *ii)* Release, *iii)* Iterations, *iv)* Production, *v)* Maintenance, and *vi)* Project death.

A1.3.4. Extreme Programming practices.

The main supposition carried out in XP is that it is able to reduce the mythical exponential curve associated to the cost of changes throughout a project [BF1995]. XP reduces this cost to allow the evolutionary design model to work properly. The reduction is achieved by means of the technologies available to help in the software development and the disciplined application of the following practices:

- **The planning game**: there is frequent communication between Customer and the development team. The development team estimates the required effort to implement user stories and Customer decides on the scope and delivery times of each iteration.

- **Small deliveries**: the aim is to quickly produce working versions of the system, even if they do not include all the functional requirements. A version constitutes a valuable result for the business and its delivery time should not take more than three months.

- **Metaphor**: the system is defined by means of a metaphor or a group of metaphors shared by Customer and the development team. A metaphor is a shared story which describes the way the system should operate (set of names that act as vocabulary to tackle a system domain in order to help the nomenclature of classes and system methodologies).

- **Simple design**: the simplest solution must be designed in a way that it can be implemented at a certain point of the project.

- **TDD**: the production of the code is driven by unitary tests. These are established by Customer before the code is written and they are constantly rebuilt after each modification of the system.

- **Refactoring**: it is a constant activity of code reconstruction with the aim of removing code duplication, improving its legibility, simplifying it and making it more flexible to facilitate subsequent changes. The code internal structure is improved without altering its external behaviour [PP2003].

- **Pair programming**: every code production must be carried out through Programmers' pair work. This entails implicit benefits (lower error rate, higher design quality, and higher satisfaction among programmers.)

- **Code collective property**: all Programmers are authorised to change any part of the code at any time.

- **Continuous integration**: each part of the code is integrated into the system once it is ready. In doing so, the system may be integrated and rebuilt several times on the same day.

- **Forty hours per week**: people must work a maximum of forty hours per week. Overtime is not allowed in two consecutive weeks. Should this happen, a problem is occurring and should be settled. Working overtime discourages the team.

- **On-site Customer**: Customer must be present and available to the team at any moment. This is one of the main success factors of an XP project. Customer constantly leads work towards what contributes more value to the business and programmers are able to solve any related doubt immediately. Oral communication is more effective than written communication.

- **Coding standards**: XP emphasizes that Programmers' communication takes place through the code. Therefore it is essential to follow programming standards to assure code legibility.

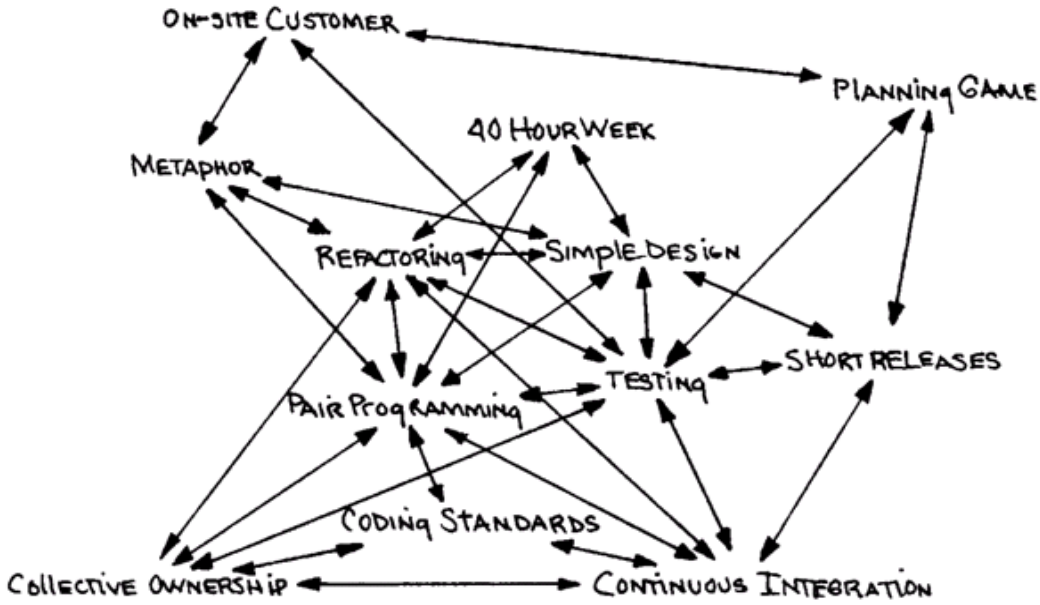


Figure A1.3: XP practices linkages and reinforcements, from [BK2000].

The most important benefit from these practices is obtained by means of their joint and balanced application, since ones rely on the others. This is shown in Figure A1.3 (retrieved from [BK2000]), where a line between two practices means that the two practices are interrelated and reciprocally reinforced.

A1.4. FINAL NOTES ON AGILE DEVELOPMENT.

There is not such a thing as a universal methodology to face successfully any software development project. All methodologies must be adapted to the context of the project (technical and human resources, development time, system type).

Traditionally, classic methodologies have tried to tackle most contextual situations in projects, by making considerable effort to adapt them, mainly in small projects and with changeable requirements. On the other hand, Agile Methodologies offer a solution that meets all requirements for a big amount of projects of this description.

One of the most high-standing qualities in Agile Methodologies is its simplicity, both in learning and application, with a resulting reduction in costs of their implementation in a development team.

Even though the creators and promoters of the most outstanding Agile Methodologies have signed the Agile Manifesto and coincide in the aforementioned principles, each methodology has its own characteristics and focuses on some specific aspects. Most of them were already used with success in real projects although they lack some diffusion and acceptance. Among the most well-known methodologies the following can be found: SCRUM [TN1986] [WSCRUM] [SBM2001], Crystal Methodologies [WCRYSTAL] [FBB1999], Dynamic Systems Development Method [WDSDM] [SJ1997], Adaptive Software Development [WASD] [HO2000], Feature-Driven Development [WFDD] [CLD1999], and Lean Development [WLD] [PP2003].

Within this group of technologies, XP stands out as it offers the biggest amount of available information and is, by far, the more widely used. This software development methodology is based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.

XP is intended to be a lightweight methodology for small to medium-size teams developing software in the face of vague or rapidly changing requirements, short deadlines, and/or dealing with new technologies. Its main characteristics can be summarized as follows:

- A small team of Programmers work at one location with Customer representation on-site.

- Development occurs in frequent builds or iterations, each of which is releasable and delivers incremental functionality.

- The code process is driven by the acceptance tests, written by Customer.

- Requirements are specified as user stories, each as a chunk of new functionality Customer requires.

- Programmers work in pairs, follow strict coding standards, and do their own unit testing.

- Requirements, architecture, and design emerge over the course of the project.

The vast majority of the practices proposed by XP are not original, but already introduced in Software Engineering with proved practical value in real world processes. The merit of XP is to integrate these practices in an effective way and to complement them with other ideas from a business, human-value and teamwork viewpoint. A historical analysis of ideas and practices that precedes the ones used in Agile Methodologies can be seen in [ASRW2002].

Appendix 2. USABILITY MODEL.

This appendix discusses the importance of applying a usability model to software development success. It tackles the concept of Usability, compiling its available mechanisms and resources. Finally, the usability model proposed by Francisco Montero [MF2005] is presented.

A2.1. INTRODUCTION TO USABILITY.

This section presents the growing concern for integrating usability models with development processes. The multiple definitions of Usability are analysed, taking into account international standards. Lastly, it is discussed how to consider Usability of a software development and how it is applied to the practical case of the project studied in this thesis.

The recent boom of IT has widely spread new technologies to all layers of society, increasing the Internet services consumption, and demanding the study of new user profiles.

Software applications are increasingly used by more different user profiles belonging to heterogeneous environments. The users of academic environment were the first group, then the business users, and nowadays the private end users. The behaviours of these different user profiles, their habits, expectations, and satisfaction rates have to be quantified by companies or institutions in order to redefine and improve the products and services they offer.

Furthermore, users are increasingly demanding software features to facilitate their work such as functionalities that allow the user to save time and automatically avoid mistakes and/or correct them. Companies are increasingly devoting time and resources to fix these issues.

The stated factors have led to require Software Engineering to incorporate Usability since the earliest phases of the development process in order to get better results.

A2.2. CONCEPT OF USABILITY.

The concept of Usability lacks a clear meaning from the academic viewpoint. Besides, Usability inherits many elements from other disciplines so that it is very important to obtain a concept definition in order to observe its relationship with these disciplines: UI

design, Graphical UI (GUI), Software Engineering, Human Computer Interaction (HCI), Information Architecture, Ergonomics, Psychology, Sociology and Linguistics, and novel disciplines, such as the study of User Experience (UX).

In scientific literature, the concept of Usability has been widely used and there are many proposed definitions. A few examples are introduced in the following paragraphs:

In Jakob Nielsen's model [NJ1993], Usability is "Part of the usefulness of the system, which is part of the practical acceptability and, finally, part of the acceptability of the system."

Jakob Nielsen is one of the pioneers in the Usability diffusion, and suggests that Usability is a multidimensional concept which establishes the attributes that a usable system must possess: *i)* Learnability, *ii)* Efficiency, *iii)* Memorability, *iv)* Error-tolerance, and *v)* Satisfaction.

- **Learnability**: "The system should be easy to use so that the user can rapidly start getting some work done with it."

- **Efficiency**: "The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible."

- **Memorability**: "The system should be easy to remember, so that the casual user is able to return to the system after some period of not having used it, without having to learn everything all over again."

- **Error-tolerance**: "The system should have a low error rate, so that the users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Further, catastrophic errors must not occur."

- **Satisfaction**: "The system should be pleasant to use, so that users are subjectively satisfied with using it; they like it."

Jenny Preece is the authoress of a great range of studies and several well-known books about Usability. Her proposed definition is the shortest one, but perhaps the most intuitive. It refers to Usability as "the development of easily usable and learnable systems" [PJ1994].

Niegel Bevan defines the Quality of Use as "the ease of use and acceptability of a system or product for a particular type of users who carry out specific tasks in a specific environment" [BKM1991].

Niegel Bevan has contributed very directly to the definition proposed by the ISO-9241-11 standard, as it might be easily concluded when this standard is observed later in this appendix.

Janice Reddish favours the idea that the goal of people working on Usability is just producing "useful tools for users" that allow users to be able to i) find what they need, ii) understand what they obtain, iii) act appropriately based on that understanding, and iv) do all this with the time and effort that they believe are necessary [RJ1995]. Her concept of Usability includes understanding the goals of users, the context of their work, and the knowledge and experience available. In other words, Usability does not just mean to make systems to be simple.

Whitney Quesenbery proposes extending the ISO-9241-11 standard to enhance its comprehensibility. This proposal comprises the definition of Usability on the basis of five features that a system should show users: *i)* Effectiveness, *ii)* Efficiency, *iii)* Engaging, *iv)* Error-Tolerance, and *v)* Easy-to-Learn [QW2001].

The ISO provides two definitions of Usability in two standards: ISO-9241-11 and ISO-9126 [BM1994].

- ISO/IEC-9241-11: "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."

This definition emphasizes the internal and external attributes of the product that contribute to its usability: functionality and efficiency. Usability does not only depend on the product but also on the user. For this reason, any software product is not an inherently usable element and will only be able to be used in a particular context and by some particular users. Usability cannot be estimated by studying a certain product in isolation.

- ISO/IEC-9126: "The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions".

This definition focuses on the concept of Usability, showing how the user carries out specific tasks in specific scenarios with effectiveness, efficiency and satisfaction. Effectiveness is understood as the precision and the plenitude with which users meet their specified aims. This idea is associated with the learnability, error-tolerance, and memorability of the system. Efficiency will be measured comparing the employed resources to the precision and plenitude obtained from the results. Satisfaction is related to the absence of inconvenience and to the positive attitude towards the use of the product.

The ISO-9241-11 definition of Usability is centred on the quality of the process itself: The second definition, by ISO-9126, focuses on quality as a result and regards the usability of a software product as a measurable feature of the product itself. [WISO].

The following section defines what a usability model is, and how it can be successfully integrated into the software development process.

A2.3. USABILITY MODELS.

The elaboration of a usability model implies the identification of those characteristics, both internal and external, that impact on the usability of a product. A usability model comprises a hierarchical decomposition of characteristics, factors and usability criteria. The development of the model can be tackled in different ways, including the identification of its objectives and metrics that allow it to characterize the usability.

Software Engineering has produced multiple models and standards associated with the usability that offers a software product. Mainly they can be grouped into two trends:

- The first group comprises those usability models that are intended to identify usability features in the developed software. Naming but a few: McCall, Boehm, and ISO-9216.

- The members of the second group are those usability models concerned with the existence of usability features of the development process itself. Being representative of this group: Software Process Improvement and Capability Determination (SPICE) [WSPICE], Cost Constructive Model II (COCOMO II) [WCOCOMO], ISO-9000, Capability Maturity Model Integration (CMMI) [WCMMI], Total Quality Management (TQM), Motorola's Six Sigma, and proposals by the European Foundation for Quality Management (EFQM) [WEFQM].

As a member of a group apart, HCI focuses on proposing models that capture the UX when using a software product. The interaction is made by means of the UI.

A2.4. IMPLEMENTING A USABILITY MODEL.

The available usability elements (factors, criteria and models) to apply to software development have been shown so far, but there are no fixed guidelines for their proper application. These elements have to be integrated into the development process and,

depending on the experience and viewpoint of the developer team, this integration tends to favour some Usability factors rather than others.

To successfully apply the criteria of Usability into a software development, the suitable international standards have to be considered. Thus depending on these standards, the proper models must be used.

However, less-experienced software engineers lack the necessary skills to select these models, to properly use them, and to successfully apply them to a development process. Furthermore, most of the proposed usability models are hard to handle and implement.

There is no closed-usability model, but at least the ISO-9126 standard constitutes a widespread and commonly-accepted first level of decomposition of Usability. On this basis, there is a commitment to use open usability models that allow software engineers to consider which the more suitable criteria are for them.

The proposal by Francisco Montero [MF2005] defines a user-centred usability model based on the deep knowledge of the tasks of the users and how they interact with the system.

The main characteristics of the model are:

- Based on internationals standards like ISO-9126.
- It is an open proposal, in which other quality criteria can be added and linked to the existing ones.
- It uses ergonomic criteria.
- It allows the Usability criteria to be put into practice.

Table A2.1 presents the proposal by Francisco Montero [MF2005], which advocates mixing the international standards with ergonomic criteria, using a user-centred point of view.

In this usability model the ergonomic criteria are associated to the criteria of ISO-9126 usability standard. These resulting criteria are grouped with the factors of understandability, learnability, and operability. In consequence, the criteria are directly linked to Usability.

Quality	Factor	Criteria	Importance Level
Usability	Understandability	Compatibility	High
		Legibility	Medium
		Prompting	Medium
		Immediate feedback	High
		Significance of codes and behaviours	High
		Helpfulness	Medium
	Learnability	Grouping	High
		Minimal action	High
		Conciseness	Low
		Consistency	High
		Information density	Medium
	Operability	Explicitly user action	High
		User control	High
		User experience	High
		Flexibility	Medium
		Error protection	High
		Quality of error messages	Low
		Error policies	Medium
		Privacy policies	Low
		Accessibility	High

Table A2.1: Usability model.

A2.5. FINAL NOTES ON USABILITY.

Usability is increasingly considered as an integral part of the development process instead of as an isolated activity performed by the quality assurance department. Furthermore, the usability criteria have to be applied from the earliest stages of the development project which, in consequence, allows lowering the economic costs.

Special attention should be paid to avoiding the erroneous supposition that Usability only affects the UI. From the user's viewpoint, the interface might represent the application in its entirety. However, the concept of Usability applies to more than a simple UI feature accompanying the application. This is reflected in the evolution of the distinctive characteristics associated to the UI.

Appendix 3. ELICITATED REQUIREMENTS.

The current appendix shows the obtained system requirements by using REMSS proposed by Amador Durán [DB2002].

A3.1. SYSTEM OBJECTIVES.

The system objectives depict, in a concise way, the main goals of the software system under development.

OBJ–01	**Manage the Registering of Attendance.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Description	The system must manage the information related to the registering of attendance at lectures and laboratory practices lessons.
Sub-objectives	None.
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	None.

Table A3.1: OBJ–01 Manage the registering of attendance.

A3.2. INFORMATION REQUIREMENTS.

IRQs reflect the information involved in the problem domain. Every used information unit is defined and its items are enumerated, along with the CRQs that they have to accomplish.

IRQ–01	**Information About Users.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• UC–01 Configure user information. • UC–02 Register attendance. • UC–03 Activate group. • UC–04 List attendees. • UC–06 Authenticate user. • UC–09 Check system logs. • UC–06.01 Check user type. • UC–06.02 Authenticate password. • UC–06.03 Validate group. • UC–06.04 Acquire student password. • UC–06.05 Acquire teacher password. • UC–06.06 Acquire student group. • UC–06.07 Acquire teacher group.	
Description	The system must store the information related to the users, in specific:	
Specific Data	• User full-name. • User login-name. • User password. • User type.	
Time to Life	**Average**	**Max**
	5 years	10 years
Concurrent Instances	**Average**	**Max**
	1	1
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The access of every user could only be in a sequential way. That is why the number of concurrent instances in the system is only one. The student preserves the same information during the whole duration of his studies.	

Table A3.2: IRQ–01 Information about users.

CRQ–01	**User Login-name Value.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	• OBJ–01 Manage the registering of attendance.
Related Requirements	• IRQ–01 Information about users. • UC–01 Configure user information.
Description	The information managed by the system must satisfy the following restriction check: the user login-name must be formed by a unique nine character sequence according to the following pattern: A0000000L.
Importance	Vital.
Urgency	Immediately.
Status	In development.
Stability	High.
Comments	A: alphanumerical character, 0: numerical digit, L: alphabetical character.

Table A3.3: CRQ–01 User login name.

CRQ–02	**User Password Value.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	• OBJ–01 Manage the registering of attendance.
Related Requirements	• IRQ–01 Information about users. • UC–06 Authenticate user.
Description	The information managed by the system must satisfy the following restriction check: the user password has to be formed by a four digits numerical sequence.
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	None.

Table A3.4: CRQ–02 User password value.

CRQ–03	**User Type Value.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	• OBJ–01 Manage the controlling of attendance.
Related Requirements	• IRQ–01 Information about users. • UC–01 Set up device configuration.
Description	The information managed by the system must satisfy the following restriction check: The user type must be member of the set {“teacher”, “student”}.
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	None.

Table A3.5: CRQ–03 User type value.

IRQ–02	**Information About Groups.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• UC–02 Register attendance. • UC–03 Activate group. • UC–04 List attendees. • UC–05 Check group activation. • UC–06 Authenticate user. • UC–07 Configure groups scheduling. • UC–08 Cancel group. • UC–09 Check system logs. • UC–06.03 Validate group. • UC–06.06 Acquire student group. • UC–06.07 Acquire teacher group.	
Description	The system must store the information related to the lecture groups or laboratory practices groups, in specific:	
Specific Data	• Group full-name. • Group code. • Group timetable.	
Time to Life	**Average**	**Max**
	1 year.	2 years.
Concurrent Instances	**Average**	**Max**
	10	100
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	A group comprises a division of the set of students enrolled in a subject. The initial set is split up into several smaller groups in order to guarantee the accommodation of the members of every group inside laboratories or classrooms, which can have a reduced capacity in comparison with the initial set.	

Table A3.6: IRQ–02 Information about groups.

CRQ–04	**Group Code Value.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	• OBJ–01 Manage the registering of attendance.
Related Requirements	• IRQ–02 Information about groups. • UC–07 Configure system.
Description	The information managed by the system must satisfy the following restriction check: The group code has to be formed by 9 unique numbers identifying the course, plus a consonant identifying the type of group (‘G’ for a lecture group, or ‘P’ for a laboratory practices group), plus a 2 digit sequence number.
Importance	Vital
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	Examples of well-formed group codes: 103112005G01, 103112005G02, 103112005P01, 103112005P02.

Table A3.7: CRQ–04 Group code value.

A3.3. FUNCTIONAL REQUIREMENTS.

A functional requirement defines the function of a software component. Every functional requirement is described by enumerating the required steps to accomplish the function, the related requirements and other considerations like possible exceptions raised or preconditions. This is the so-called “fully dressed” form of a functional requirement.

The following section addresses the use cases of the system.

A3.3.1. Use case diagrams.

Unified Model Language (UML) [WUML] use case diagrams of the system are used, as the technique recommended by REMSS. The UML use case diagrams are intended to enhance the understandability of the functional requirements definitions.

The UML use case diagrams overview the usage requirements for a system and, in outline, they depict the use cases themselves, actors and associations between them [AS2004]:

- **Use case**: describes a succession of actions that contribute something of quantifiable worth to an actor. Use cases are represented as horizontal ellipses.

- **Actor**: is a person, organization, or external system that plays a role in one or more relations with the system under development. Actors are represented as stick figures.

- **Association**: between an actor and a use case is represented by a solid line. An association exists each time an actor is involved with an interaction described by a use case.

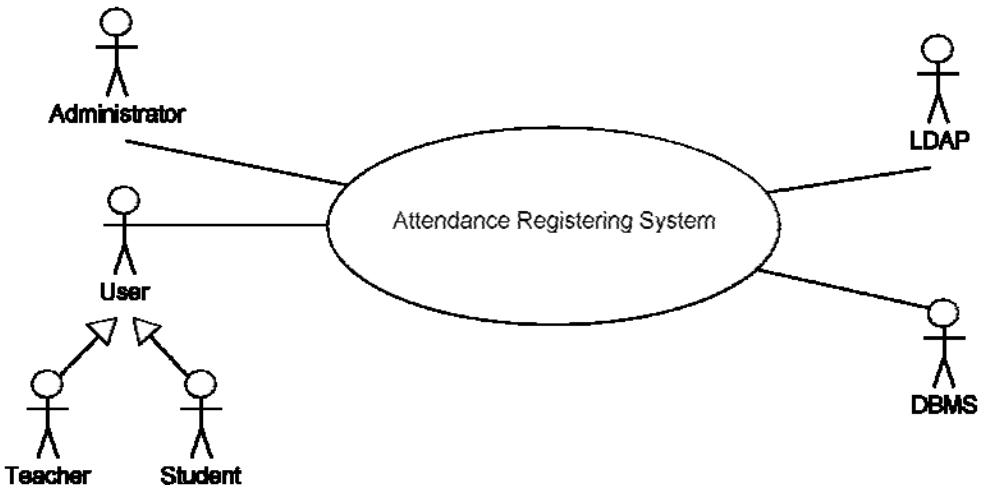


Figure A3.1: UML system diagram.

The UML use case diagram depicted in Figure A3.1 shows all the agents (human users and external systems) which interchange any information with the system under development, the Attendance Registering System.

As a convenient standard used by the UML designer community, the main actors are usually placed at the left side and the secondary actors at the right side. The use cases are placed on the centre, in a top-down reading order whenever possible.

On the left side of Figure A3.1, the actors correspond with human users interacting with the system. On the right side, there are two external systems, the LDAP server and the DBMS server. Being an actor means that they are out of the scope of the system itself, and they just act like agents who interact (interchanging data) with it.

The hierarchy of actors shown between user, teacher and student, indicates that the latter two actors share common properties present in super-actor named user.

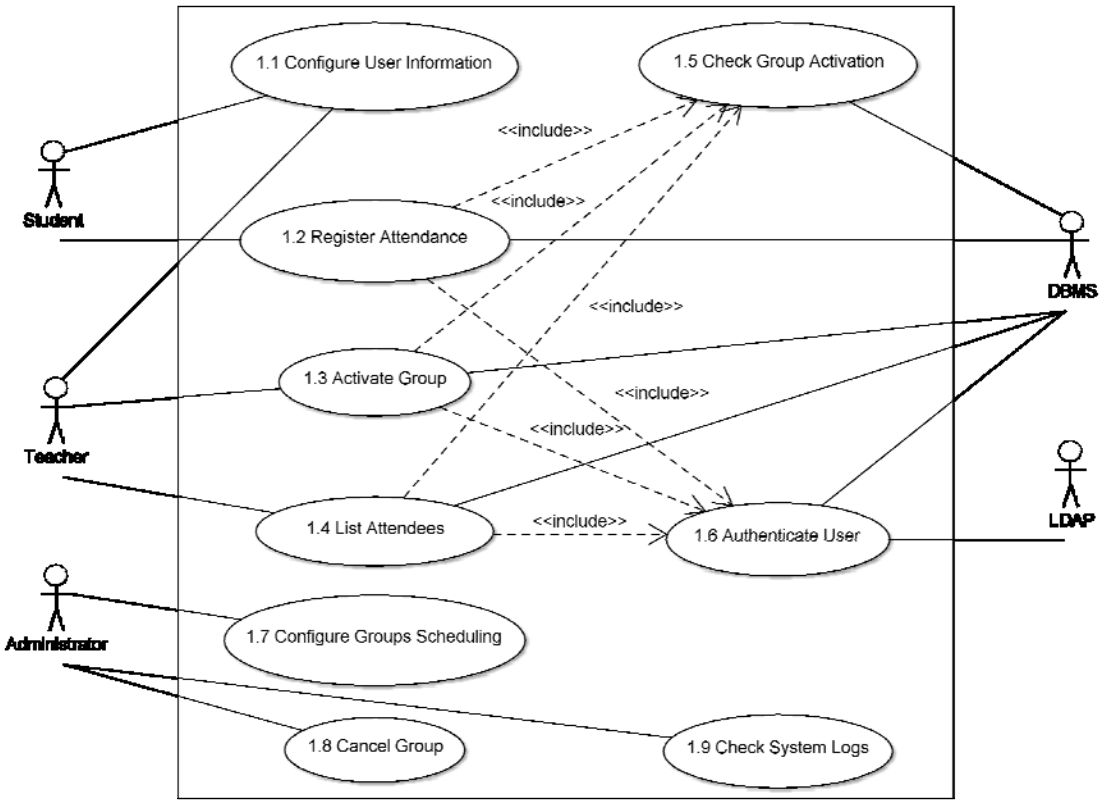


Figure A3.2: Level 1 UML use cases diagram.

In the diagram shown in Figure A3.2, the use cases (or functional requirements that the system has to meet) are depicted. Associations between actors and use cases indicate that these actors have the ability to interact with their associated functionalities.

There are associations with an arrow and which show the stereotype “include”. Those use cases where the arrowhead is aimed correspond to a common functionality for those where the arrow starts. As an example, the use case 1.6 (Authenticate User) is executed every time the functionality described by use cases 1.2, 1.3 or 1.4 is run.

It should be noted that the UML use case diagrams act as a valuable tool to enhance the RE process understandability, and to easily define the scope of the system. The level 1 case diagram depicts a rectangular shape containing the system scope (see Figure A.3.2).

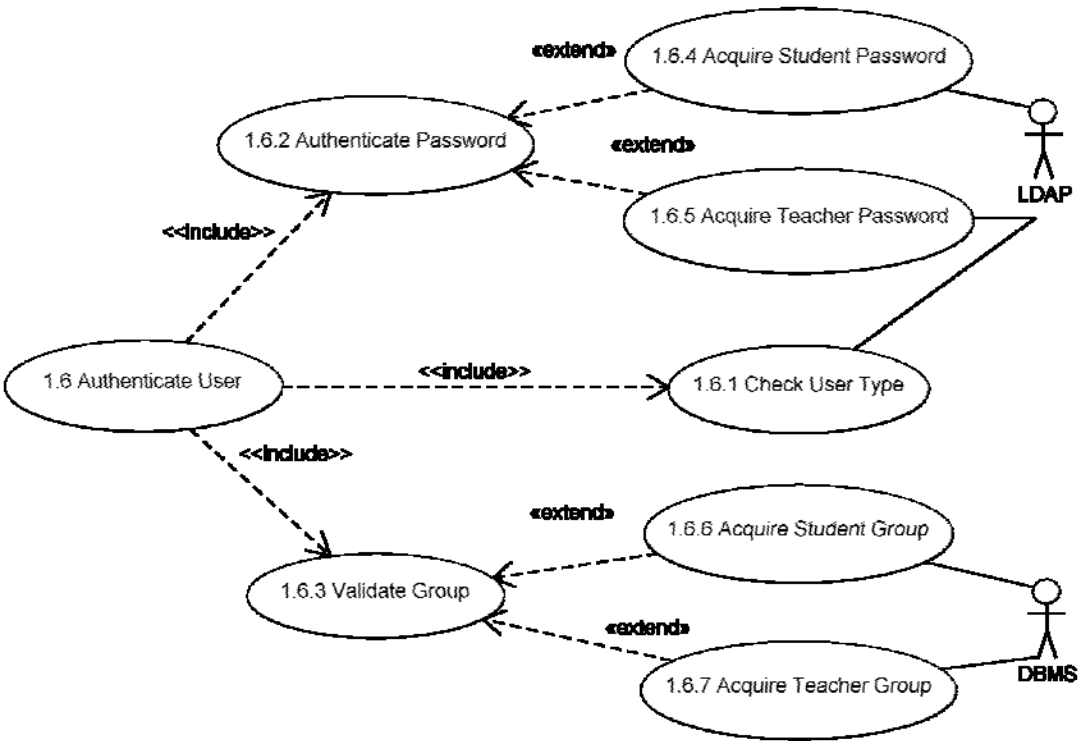


Figure A3.3: Authenticate User UML use case diagram.

Figure A3.3 depicts the use case diagram as a result of the "exploration" of the use case 1.6 (Authenticate User). Exploration, in this context, means that the use case 1.6 is complex enough to be divided into several use cases with less complexity.

There are associations with an arrow and which show the stereotype "extend". Those use cases where the arrow starts correspond to an optional functionality, and those where the arrowhead is aimed correspond to the place where the decision of running the associated optional functionality is taken. This decision is made taking into account a determined condition in execution time.

For instance, the use case 1.6.6 (Acquire Student Group) is run when, during the execution of use case 1.6.3, to gather information about a student is needed.

A3.3.2. Definition of actors.

As stated earlier in this chapter, an actor represents a person or an external system interacting with the focus software system (the system under development).

ACT–01	**User.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Description	This abstract actor represents a generalization for a student or a teacher who uses the system.
Comments	None.

Table A3.8: ACT–01 User.

ACT–02	**Student.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Description	This actor represents a student attending at lectures or laboratory practices lessons.
Comments	None.

Table A3.9: ACT–02 Student.

ACT–03	**Teacher.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Description	This actor represents a teacher responsible of lectures or laboratory practices lessons.
Comments	None.

Table A3.10: ACT–03 Teacher.

ACT–04	**Administrator.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Description	This actor represents a member of the system administrators group.
Comments	None.

Table A3.11: ACT–04 Administrator.

ACT–05	**LDAP.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Description	This actor represents an external system that stores directory information concerned with organizational units and people related to the university.
Comments	None.

Table A3.12: ACT–01 LDAP.

ACT–06	**DBMS.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Description	This actor represents an external system that stores relations between practice groups, timetables, responsible teachers, and students enrolled at every laboratory practices group.
Comments	None.

Table A3.13: ACT–06 DBMS.

A3.3.3. System use cases.

A use case describes a determined sequence of actions to accomplish a functional requirement of the system. That is, there is a one-to-one link between a functional requirement and a use case.

UC–01	**Configure User Information.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users.	
Start event	The system should behave as described in this use case when the user decides to configure his related information.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The user asks the system for starting the configure user information process.
	2	The system asks the user for the following related data of the user: user login-name and user password (IRQ–01 Information about users).
	3	The user provides the system with the required data and asks the system for storing it.
	4	The system checks if the user login-name (CRQ–01 User login-name value) and the user password (CRQ–02 User password value) are well-formed.
	5	The system stores the provided data.
	6	The system informs the user that the process has successfully finished.
Post-condition	The system has stored the user information.	
Exceptions	**Step**	**Action**
	3	If the user decides to cancel the process, then the use case finishes with no effect.
	4	If the information provided by the user is not well-formed the system allows the user to repeat the attempt (steps 2-4) indefinitely.
Performance	**Step**	**Time Boundary**
	5	2 seconds.
Frequency	60 times / first week of every semester.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	At step 5, the user password could be stored as a Hash to improve the system security (NFR–03 Security of private user data).	

Table A3.14: UC–01 Configure user information.

UC–02	**Register Attendance.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.	
Start event	The system should behave as described in this use case when a student decides to record his attendance.	
Precondition	The student has to configure his own data in advance (UC–01 Configure user information).	
Normal Sequence	**Step**	**Action**
	1	The student asks the system for starting the register attendance process.
	2	The system automatically obtains the current system date and time.
	3	The use case UC–05 (Check group activation) is run, using the date and time retrieved at step 2.
	4	The use case UC–06 (Authenticate user) is run, using the current date and time retrieved at step 2, and the group code retrieved at step 3 (IRQ–02 Information about groups).
	5	The system asks the DBMS for storing the attendance record of the student: the current date and time at step 2, the retrieved group code at step 3 (IRQ–02 Information about groups), and the user login-name at step 4 (IRQ–01 Information about users).
	6	The DBMS stores the attendance record of the student.
	7	The system writes a new record in the system log with the following information: current date and time of creation; the "Attendance" literal; student login-name (IRQ–01 Information about users); and group code (IRQ–02 Information about groups).
	8	The system informs the student that the process has successfully finished.
Post-condition	The DBMS has stored the information corresponding to the student attendance.	
Exceptions	**Step**	**Action**
	3	If there is no information about groups scheduling stored in the system, the system communicates this situation to the student and then, the use case finishes with no effect.
	3	If the current scheduled group has not been activated yet, the system communicates this situation to the student and then, the use case finishes with no effect.
	3	If the current scheduled group has been cancelled, the system communicates this situation to the student and then, the use case finishes with no effect.
	4	If the student is not authenticated, the system communicates this situation to him explaining the reason, and then, the use case finishes with no effect.
	4	If the user type is not a student, the system communicates this situation to him and then, the use case finishes with no effect.
	5	If the attendance has already been registered, the system communicates this situation to the student and then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	2	1 second.
	7	2 seconds.
Frequency	20 times /minute.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The expected frequency of usage is only reached during the ten previous minutes to the start of a lecture or a laboratory practices lesson (when the students register their attendance). The developed platform is able to attend up to 50 incoming user's commands per minute (not having into account the user's response time).	

Table A3.15: UC–02 Register attendance.

UC–03	**Activate Group.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.	
Start event	The system should behave as described in this use case when a teacher decides to activate a group.	
Precondition	The teacher has to configure his own data in advance (UC–01 Configure user information).	
Normal Sequence	**Step**	**Action**
	1	The teacher asks the system for starting the group activation process.
	2	The system automatically obtains the current system date and time.
	3	The use case UC–05 (Check group activation) is run, using the date and time retrieved at step 2.
	4	The use case UC–06 (Authenticate user) is run, using the current date and time retrieved at step 2, and the group code retrieved at step 3 (IRQ–02 Information about groups).
	5	The system asks the DBMS for storing the activated status of the current group, using the retrieved group code at step 3.
	6	The DBMS stores the activated status for the current group.
	7	The system writes a new record in the system log with the following information: current date and time of creation; the "Activate group" literal; teacher login name (IRQ–01 Information about users) (received at step 4); and group code (IRQ–02 Information about groups).
	8	The system informs the teacher that the process has successfully finished.
Post-condition	The DBMS has stored the information corresponding to the group activation.	
Exceptions	**Step**	**Action**
	3	If there is no information about groups scheduling stored in the system, the system communicates this situation to the teacher and then, the use case finishes with no effect.
	3	If the current scheduled group has already been activated, the system communicates this situation to the teacher and then, the use case finishes with no effect.
	3	If the current scheduled group has been cancelled, the system communicates this situation to the teacher and then, the use case finishes with no effect.
	4	If the teacher is not authenticated, the system communicates this situation to him explaining the reason and then, the use case finishes with no effect.
	4	If the user type is not a teacher, the system communicates this situation to him and then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	2	1 second.
	7	2 seconds.
Frequency	Once / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The system is in charge of closing the groups automatically according to preset schedule for every group.	

Table A3.16: UC–03 Activate group.

UC–04	**List Attendees.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.	
Start event	The system should behave as described in the following use case when the teacher decides to list the attendees at the current group.	
Precondition	The teacher has to configure his own data in advance (UC–01 Configure user information).	
Normal Sequence	**Step**	**Action**
	1	The teacher asks the system for starting the list attendees' process.
	2	The system automatically obtains the current system date and time.
	3	The use case UC–05 (Check group activation) is run, using the date and time retrieved at step 2.
	4	The use case UC–06 (Authenticate user) is run, using the current date and time retrieved at step 2, and the group code retrieved at step 3 (IRQ–02 Information about groups).
	5	The system asks the DBMS for information about the attendees at current group, using the date and time retrieved at step 2 and the group code retrieved at step 3 (IRQ–02 Information about groups): group full-name, student full name, student login-name (IRQ–01 Information about users).
	6	The DBMS provides the system with the required information.
	7	The system writes a new record in the system log with the following information: date and time of creation; the "Listing" literal; teacher login-name (IRQ–01 Information about users) (received at step 4); and group code (IRQ–02 Information about groups).
	8	The system displays a sorted list in alphabetical order by user name with those data obtained at step 6, with the following information: user full-name and user login name (IRQ–01 Information about users); group full-name and group code (IRQ–02 Information about groups).
Post-condition	None.	
Exceptions	**Step**	**Action**
	3	If there is no information about groups scheduling stored in the system, the system communicates this situation to the teacher and then, the use case finishes with no effect.
	3	If the current group has not been activated yet, the system communicates this situation to the teacher and then, the use case finishes with no effect.
	3	If the current group has been cancelled, the system communicates this situation to the teacher and then, the use case finishes with no effect.
	4	If the teacher is not authenticated, the system communicates this situation to him and then, the use case finishes with no effect.
	4	If the user type is not a teacher, the system communicates this situation to him and then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	2	1 second.
	7	2 seconds.
Frequency	Once / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	None.	

Table A3.17: UC–04 List attendees.

UC–05	**Check Group Activation.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–02 Information about groups.	
Start event	The system should behave as described in this abstract use case during the course of the following use cases: • UC–02 Register attendance. • UC–03 Activate group. • UC–04 List attendees.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The system receives the current date and time.
	2	The system retrieves the current scheduled group code (IRQ–02 Information about groups), using the received current date and time at step 1, according to the information stored in the system.
	3	The system asks to the DBMS if the current group code (IRQ–02 Information about groups) has already been activated, using the group code retrieved at previous step.
	4	The DBMS provides information about the group status to the system.
	5	The system returns the current activated group code (IRQ–02 Information about groups).
Post-condition	None.	
Exceptions	**Step**	**Action**
	2	If there is no information about groups scheduling stored in the system, an error condition is arisen, then, the use case finishes with no effect.
	2	If the current scheduled group has been cancelled, an error condition is arisen, then, the use case finishes with no effect.
	4	If the group has not been activated yet, an error condition is arisen, then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	2	2 seconds.
Frequency	25 times / minute.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The expected frequency of usage is only reached during the ten previous minutes to the start of a lecture or a laboratory practices lesson (when the students register their attendance).	

Table A3.18: UC–05 Check group activation.

UC–06	**Authenticate User.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.	
Start event	The system should behave as described in this abstract use case during the course of the following use cases: • UC–02 Register attendance. • UC–03 Activate group. • UC–04 List attendees.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The system receives the current date and time.
	2	The system receives the current group code (IRQ–02 Information about groups).
	3	The system asks the user for permission to retrieve the user data stored in it: user login-name and user password (IRQ–01 Information about users).
	4	The user allows the system to use the required data at step 3.
	5	The system retrieves the required data at step 3.
	6	The use case UC–06.01 (Check user type) is run, using the retrieved user login-name at step 5 (IRQ–01 Information about users).
	7	The use case UC–06.02 (Authenticate password) is run, using the retrieved user login-name and the user password at step 5, and the user type at step 6 (IRQ–01 Information about users).
	8	The use case UC–06.03 (Validate group) is run, using the received current date and time at step 1, group code at step 2 (IRQ–02 Information about groups), the user login-name at step 5 and user type at step 6 (IRQ–01 Information about users).
	9	The system returns the authenticated user login-name and the user type.
Post-condition	None.	
Exceptions	**Step**	**Action**
	4	If the user decides to cancel the process, then the use case finishes with no effect.
	6	If there is some error condition checking the user type, an error condition is arisen, then, the use case finishes with no effect.
	7	If there is some error condition authenticating the user, an error condition is arisen, then, the use case finishes with no effect.
	8	If there is some error condition validating the group, an error condition is arisen, then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	–	–
Frequency	25 times / minute.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The expected frequency of usage is only reached during the ten previous minutes to the start of a lecture or a laboratory practices lesson (when the students register their attendance).	

Table A3.19: UC–06 Authenticate user.

UC–07	**Configure Groups Scheduling.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–02 Information about groups.	
Start event	The system should behave as described in this use case when the administrator decides to configure the groups scheduling.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The administrator asks the system for starting the groups scheduling configuration process.
	2	The system asks the administrator for the following data for configure the groups scheduling: group code, group timetable (IRQ–02 Information about groups).
	3	The administrator provides the required data and asks the system for storing it.
	4	The system checks if the group code is well-formed (CRQ–04 Group code value).
	5	The system stores the provided data.
	6	The system informs the administrator that the process has successfully finished.
	7	If the administrator wants to specify the data of another group, the system allows the administrator to repeat the entire process (steps 2-7) indefinitely.
Post-condition	The system has stored the information about the groups scheduling.	
Exceptions	**Step**	**Action**
	3	If the administrator decides to cancel the process, then the use case finishes with no effect.
	4	If the group code is not well-formed the system allows the administrator to repeat the attempt (steps 2-4) indefinitely.
Performance	**Step**	**Time Boundary**
	5	1 second.
Frequency	Once / semester.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The administrator does not have to authenticate into the system because he uses the system console.	

Table A3.20: UC–07 Configure groups scheduling.

UC–08	**Cancel Group.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–02 Information about groups.	
Start event	The system should behave as described in this use case when the administrator decides to cancel the scheduling of a group.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The administrator asks the system for starting the cancel group process.
	2	The system asks the administrator for the following data for cancel a group scheduling: group code (IRQ–02 Information about groups) and date to cancel.
	3	The administrator provides the required data to the system.
	4	The system checks if the group code is well-formed (CRQ–04 Group code value).
	5	The system modifies the groups scheduling in order to reflect the cancelled status of the group corresponding to the group code provided at step 3, for the date given at step 2.
	6	The system informs the administrator that the process has successfully finished.
Post-condition	The system has cancelled the group for the group code given.	
Exceptions	**Step**	**Action**
	3	If the administrator does not provide any date, the system assumes that the group is cancelled from current date onwards.
	3	If the administrator decides to cancel the process, then the use case finishes with no effect.
	4	If the group code is not well-formed the system allows the administrator to repeat the attempt (steps 2-4) indefinitely.
Performance	**Step**	**Time Boundary**
	5	1 second.
Frequency	Once / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The administrator does not have to authenticate into the system because he uses the system console.	

Table A3.21: UC–08 Cancel group.

UC–09	**Check System Logs.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.	
Start event	The system should behave as described in this use case when the administrator decides to check the system logs.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The administrator asks the system for starting the check system logs process.
	2	The system retrieves the system logs.
	3	The system elaborates and displays a sorted list in descending order by date and time of the log records with those data obtained in the previous step, with the following information: date and time of creation; literal, user login name (IRQ–01 Information about users); and group code (IRQ–02 Information about groups).
Post-condition	None.	
Exceptions	**Step**	**Action**
	2	If they does not exist any log, the system communicates this situation to the administrator and then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	2	2 seconds.
Frequency	5 times / day.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The administrator does not have to authenticate into the system because he uses the system console.	

Table A3.22: UC–09 Check system logs.

UC–06.01	**Check User Type.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users.	
Start event	The system should behave as described in this abstract use case during the course of the following use case: • UC–06 Authenticate user.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The system receives the user login-name and the user password (IRQ–01 Information about users).
	2	The system asks the LDAP for checking if the login-name received at step 1 is member of the students group.
	3	The system asks the LDAP to checking if the login-name received at step 1 is member of the teachers group.
	4	The system returns the user type (IRQ–01 Information about users).
Post-condition	None.	
Exceptions	**Step**	**Action**
	4	If the user does not belong to any group, an error condition is arisen, then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	–	–
Frequency	50 times / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	None.	

Table A3.23: UC–06.01 Check user type.

UC–06.02	**Authenticate Password.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users.	
Start event	The system should behave as described in this abstract use case during the course of the following use case: • UC–06 Authenticate user.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The system receives the user login-name, the user password and the user type (IRQ–01 Information about users).
	2	If the user is a student the use case UC–06.04 (Acquire student password) is run, using the received user login-name at step 1.
	3	If the user is a teacher the use case UC–06.05 (Acquire teacher password) is run, using the received user login-name at step 1.
	4	The system compares the received user password at step 1 with the user password acquired.
	5	The system informs that the user password has been authenticated.
Post-condition	None.	
Exceptions	**Step**	**Action**
	4	If the comparison is not successful, an error condition is arisen, then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	–	–
Frequency	50 times / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The user password could be Hash-formed to improve the system security. The way the system asks the LDAP for the user password would be different whether the user is actually a student or a teacher.	

Table A3.24: UC–06.02 Authenticate password.

UC–06.03	**Validate Group.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.	
Start event	The system should behave as described in this abstract use case during the course of the following use case: • UC–06 Authenticate user.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The system receives the current date and time.
	2	The system receives the group code of the current group (IRQ–02 Information about groups).
	3	The system receives the user login-name and the user type (IRQ–01 Information about users).
	4	If the received user is a student the use case UC–06.06 (Acquire student group) is run, using the received current date and time at step 1 and the user login-name at step 3.
	5	If the received user is a teacher the use case UC–06.07 (Acquire teacher group) is run, using the received current date and time at step 1 and the user login-name at step 3.
	6	The system compares the received group code at step 2 with the group code acquired.
	7	The system informs that the group code has been validated.
Post-condition	None.	
Exceptions	**Step**	**Action**
	6	If the comparison is not successful, an error condition is arisen, then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	–	–
Frequency	50 times / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The way the system asks the DBMS for the user group will be different whether the user is actually a student or a teacher.	

Table A3.25: UC–06.03 Validate group.

UC–06.04	**Acquire Student Password.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users.	
Start event	The system should behave as described in this abstract use case during the course of the following use case: • UC–06.02 Authenticate password.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The system receives the student login-name (IRQ–01 Information about users).
	2	The system asks the LDAP for the actual student password, using the received login-name.
	3	The system returns the acquired student password (IRQ–01 Information about users).
Post-condition	None.	
Exceptions	**Step**	**Action**
	2	If no password or an error condition is returned by the LDAP, an error condition is arisen; then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	–	–
Frequency	45 times / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The student password could be Hash-formed to improve the system security. The error condition returned at the exception for step 2 could be a special invalid password.	

Table A3.26: UC–06.04 Acquire student password.

UC–06.05	**Acquire Teacher Password.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users.	
Start event	The system should behave as described in this abstract use case during the course of the following use case: • UC–06.02 Authenticate password.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The system receives the teacher login-name (IRQ–01 Information about users).
	2	The system asks the LDAP for the actual teacher password, using the received teacher login-name.
	3	The system returns the acquired teacher password (IRQ–01 Information about users).
Post-condition	None.	
Exceptions	**Step**	**Action**
	2	If no password or an error condition is returned by the LDAP, an error condition is arisen; then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	–	–
Frequency	5 times / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The teacher password could be Hash-formed to improve the system security. The error condition returned at the exception for step 2 could be a special invalid password.	

Table A3.27: UC–06.05 Acquire teacher password.

UC–06.06	**Acquire Student Group.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.	
Start event	The system should behave as described in this abstract use case during the course of the following use case: • UC–06.03 Validate group.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The system receives the current date and time.
	2	The system receives the student login-name (IRQ–01 Information about users).
	3	The system asks the DBMS for the actual student group, using the received current date and time at step 1 and the student login-name at step 2.
	4	The system returns the acquired group for the student (IRQ–02 Information about groups).
Post-condition	None.	
Exceptions	**Step**	**Action**
	3	If no group code or an error condition is returned by the DBMS, an error condition is arisen; then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	–	–
Frequency	45 times / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The user password could be Hash-formed to improve the system security. The error condition returned at the exception for step 3 could be a special invalid group code.	

Table A3.28: UC–06.06 Acquire student group.

UC–06.07	**Acquire Teacher Group.**	
Version	1 (18/10/2010).	
Authors	• Alfonso De Gea.	
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).	
Related Objectives	• OBJ–01 Manage the registering of attendance.	
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.	
Start event	The system should behave as described in this abstract use case during the course of the following use case: • UC–06.03 Validate group.	
Precondition	None.	
Normal Sequence	**Step**	**Action**
	1	The system receives the current date and time.
	2	The system receives the teacher login-name (IRQ–01 Information about users).
	3	The system asks the DBMS for the actual teacher group, using the received current date and time at step 1 and the teacher login-name at step 2.
	4	The system returns the acquired group for the teacher (IRQ–02 Information about groups).
Post-condition	None.	
Exceptions	**Step**	**Action**
	3	If no group code or an error condition is returned by the DBMS, an error condition is arisen; then, the use case finishes with no effect.
Performance	**Step**	**Time Boundary**
	–	–
Frequency	5 times / hour.	
Importance	Vital.	
Urgency	Immediately.	
Status	Finished.	
Stability	High.	
Comments	The error condition returned at the exception for step 3 could be a special invalid group code.	

Table A3.29: UC–06.07 Acquire teacher group.

A3.4. NON-FUNCTIONAL REQUIREMENTS.

A non-functional requirement represents a demanded feature of the system under development, of the development process itself, of the production service or of any other development aspect, which usually indicates a restriction on them.

This section compiles the non-functional requirements of the system. The fields are self-described, being the focus of interest the description of the requirement, as well as the related comments to it, which serve as a start point to address the requirement.

NFR–01	**Production Environment.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	–
Related Requirements	–
Description	The system shall operate in an environment of an Intel Pentium IV workstation server with 1024 megabytes of main memory, 5 Gigabytes free hard drive space and 10/100 Ethernet network connection. The operating system of the workstation will be Microsoft Windows XP. The users' devices are NFC-enabled mobile phones (handsets or smart-phones).
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	None.

Table A3.30: NFR–01 Production environment.

NFR–02	**Portability.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	–
Related Requirements	–
Description	The system should be easily ported into another modern operating systems of Microsoft Windows family (like Vista or Windows 7) or Linux distributions (Ubuntu, Suse, Debian, Red Hat, etc). Similarly, the system must be prepared for changes in the operating system of the mobile clients (handsets).
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	To address the first question, portability at the server side, Java free software-based products may be chosen, taking advantage of the portability of the Java architecture. To solve the portability issues at the clients, programmatic standards for mobiles devices must be used (such as MIDP, CLID or JSR-257, etc) easily portables between different handset platforms.

Table A3.31: NFR–02 Portability.

NFR–03	**Usability.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	–
Related Requirements	–
Description	The system should provide a UI design having suitable Usability features to mobile devices, specifically for handsets. For example, internationalisation of the UI and adaptation to the way users interact with the handset (via keypad).
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	A previous study has to be made.

Table A3.32: NFR–03 Usability.

NFR–04	**Economic Cost Reduction.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	–
Related Requirements	–
Description	The developed platform shall take into account the reduction of monetary investment incurred for license acquisition and maintenance of the software tools used.
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	This issue has to be considered in the feasibility study of the system in order to choose the software tools to implement the system (repository, programming libraries, application server, etc).

Table A3.33: NFR–04 Economic cost reduction.

NFR–05	**Security of Private User Data.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	OBJ–01 Manage the registering of attendance.
Related Requirements	IRQ–01 Information about users.
Description	The developed platform shall take into account the security of the private user data stored or managed by it and protecting them against possible hacking attacks.
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	The passwords could be stored as a Hash (using SSHA algorithm for example). The "secure element" could be used for storing the user information (IRQ–01 Information about users).

Table A3.34: NFR–05 Security of private user data.

NFR–06	**Deployment of the Client Software.**
Version	1 (18/10/2010).
Authors	• Alfonso De Gea.
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).
Related Objectives	–
Related Requirements	–
Description	The client software will be available to users via Internet, using for this purpose the telematics resources of the university.
Importance	Vital.
Urgency	Immediately.
Status	Finished.
Stability	High.
Comments	A candidate system for the delivery of the client software to the users could be Moodle [DT2003], the e-learning system owned by the UPCT.

Table A3.35: NFR–06 Deployment of the client software.

Appendix 4. BATTERY OF UNIT TEST CASES.

This appendix shows the battery of unit test cases designed for proving the proper running of all the system functionalities.

A4.1. TEST CASES FOR THE SYSTEM FUNCTIONALITIES.

The selected design technique for unit testing is based on black-box tests, where the tester checks the behaviour of each unit by just providing data entries to it (see Chapter 5).

The design of a test case suite for each functional requirement (also known as "use case") allows the tester to check all the system functionalities in an isolated way before their integration to constitute the whole software system.

A unit test case comprises a prearranged number of tests of two types: *i)* one normal-typed test, and *ii)* several exception-typed tests (see Table A4.1).

TC–01	**Unit Test Cases for UC–01 (Configure User Information).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–01 Configure user information.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Login name="T0000002T", password="0002" (teacher).
		Output:	Message showing the successful user configuration.
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1, with login name="".
		Reason:	Login name cannot be empty (CRQ–01).
		Output:	Error message showing the exception reason.
		Result:	Success.
	3	**Type:**	Exception.
		Values:	Same as test 1, with password="002".
		Reason:	Password length has to be at least 4 digits (CRQ–02).
		Output:	Error message showing the exception reason.
		Result:	Success.
	4	**Type:**	Exception.
		Values:	Same as test 1, with retyped password="0004".
		Reason:	Password and retyped password have to be equals.
		Output:	Error message showing the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	None.		

Table A4.1: TC–01 Configure user information.

The normal case checks the proper execution of the implemented functionality. The exceptional test cases are intended to trigger the exceptions that the module can throw when it finds any error condition or bad input data. The data input is designed depending on the test case type (see Section 5.5 in Chapter 5).

TC–02	**Unit Test Cases for UC–02 (Register Attendance).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–02 Register attendance.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Login name="S0000001S", password="0001" (student).
		Output:	Message showing the successful attendance registering.
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	There is no group scheduled.
		Output:	Error message showing the exception reason.
		Result:	Success.
	3	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The current group has not been activated.
		Output:	Error message showing the exception reason.
		Result:	Success.
	4	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The current group has been cancelled.
		Output:	Error message showing the exception reason.
		Result:	Success.
	5	**Type:**	Exception.
		Values:	Login name="00000000X", password="0000" (invalid).
		Reason:	Bad credentials.
		Output:	Error message showing the exception reason.
		Result:	Success.
	6	**Type:**	Exception.
		Values:	Login name="T0000002T", password="0002" (teacher).
		Reason:	The user is not a student.
		Output:	Error message showing the exception reason.
		Result:	Success.
	7	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The attendance has already been registered.
		Output:	Error message showing the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	In test 2, 3 and 4, the exception is reproduced changing the group scheduling (null, inactive or cancelled). It does not depend only on the input values.		

Table A4.2: TC–02 Register attendance.

TC–03	**Unit Test Cases for UC–03 (Activate Group).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–03 Activate group.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Login name="T0000002T", password="0002" (teacher).
		Output:	Message showing the successful group activation.
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	There is no group scheduled.
		Output:	Error message showing the exception reason.
		Result:	Success.
	3	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The current scheduled group has already been activated.
		Output:	Error message showing the exception reason.
		Result:	Success.
	4	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The current group has been cancelled.
		Output:	Error message showing the exception reason.
		Result:	Success.
	5	**Type:**	Exception.
		Values:	Login name="00000000X", password="0000" (invalid).
		Reason:	Bad credentials.
		Output:	Error message showing the exception reason.
		Result:	Success.
	6	**Type:**	Exception.
		Values:	Login name="S0000001S", password="0001" (student).
		Reason:	The user is not a teacher.
		Output:	Error message showing the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	In test 2, 3 and 4, the exception is reproduced changing the group scheduling (null, active or cancelled). It does not depend only on the input values.		

Table A4.3: TC–03 Activate group.

TC–04	**Unit Test Cases for UC–04 (List Attendees).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–04 List attendees.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Login name="T0000002T", password="0002" (teacher).
		Output:	Message showing the successful group activation.
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	There is no group scheduled.
		Output:	Error message showing the exception reason.
		Result:	Success.
	3	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The current scheduled group has not been activated yet.
		Output:	Error message showing the exception reason.
		Result:	Success.
	4	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The current group has been cancelled.
		Output:	Error message showing the exception reason.
		Result:	Success.
	5	**Type:**	Exception.
		Values:	Login name="00000000X", password="0000" (invalid).
		Reason:	Bad credentials.
		Output:	Error message showing the exception reason.
		Result:	Success.
	6	**Type:**	Exception.
		Values:	Login name="S0000001S", password="0001" (student).
		Reason:	The user is not a teacher.
		Output:	Error message showing the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	In test 2, 3 and 4, the exception is reproduced changing the group scheduling (null, inactive or cancelled). It does not depend only on the input values.		

Table A4.4: TC–04 List attendees.

TC–05	**Unit Test Cases for UC–05 (Check Group Activation).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–05 Check group activation.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Date="2010-11-29", time="11:10".
		Output:	"103112006P01".
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	There is no group scheduled.
		Output:	Error code indicating the exception reason.
		Result:	Success.
	3	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The current scheduled group has not been activated yet.
		Output:	Error code indicating the exception reason.
		Result:	Success.
	4	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The current group has been cancelled.
		Output:	Error code indicating the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	In test 2, 3 and 4, the exception is reproduced changing the group scheduling (null, inactive or cancelled). It does not depend only on the input values.		

Table A4.5: TC–05 Check group activation.

TC–06	**Unit Test Cases for UC–06 (Authenticate User).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–06 Authenticate user.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Date="2010-11-29", time="11:10", group code="103112006P01", login name="S0000001S", password="0001" (student).
		Output:	"Student".
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1, with login name="00000000X", password="0000" (invalid).
		Reason:	Bad credentials.
		Output:	Error code indicating the exception reason.
		Result:	Success.
	3	**Type:**	Exception.
		Values:	Same as test 1, with group code="103112006P02".
		Reason:	Group not allowed.
		Output:	Error code indicating the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	None.		

Table A4.6: TC–06 Authenticate user.

TC–07	**Unit Test Cases for UC–07 (Configure Groups Scheduling).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–07 Configure groups scheduling.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Group code="103112006P01", date="2010-11-29", start time="11:00", end time="12:00".
		Output:	Message showing the successful group scheduling.
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1.
		Reason:	The current scheduling overlaps a former one.
		Output:	Error message showing the exception reason.
		Result:	Success.
	3	**Type:**	Exception.
		Values:	Same as test 1 with group code="103112006000".
		Reason:	The group code has to be well formed (CRQ–04).
		Output:	Error message showing the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	None.		

Table A4.7: TC–07 Configure groups scheduling.

TC–08	**Unit Test Cases for UC–08 (Cancel Group).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–08 Cancel group.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Group="103112006P01", date="2010-11-29", time="11:00".
		Output:	Message showing the successful group cancellation.
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1 with group code="103112006000".
		Reason:	The group code has to be well formed (CRQ–04).
		Output:	Error message showing the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	In the normal case, the group already can be cancelled or activated in advance. The result is a cancelled group anyway. A group not scheduled is not selectable for its cancelling.		

Table A4.8: TC–08 Cancel group.

TC–09	**Unit Test Cases for UC–09 (Check System Logs).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–09 Check system logs.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	None.
		Output:	System logs (Step 3 UC–09)
		Result:	Success.
	2	**Type:**	Exception.
		Values:	None.
		Reason:	There is no record in the system log yet.
		Output:	Error message showing the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	In this test case no input has to be provided. It is only intended to check the proper running of the functionality.		

Table A4.9: TC–09 Check system logs.

TC–10	**Unit Test Cases for UC–06.01 (Check User Type).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–06.01 Check user type.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Login name="T0000002T", password="0002" (teacher).
		Output:	"Teacher".
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Login name="00000000X", password="0000" (invalid).
		Reason:	The user type is not correct (CRQ–03).
		Output:	Error indicating the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	The user type in the normal case can be "student" or "teacher", having selected the latter type.		

Table A4.10: TC–06.01 Check user type.

TC–11	**Unit Test Cases for UC–06.02 (Authenticate Password).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–06.02 Authenticate password.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Login =“T0000002T”, password=“0002”, type=“Teacher”.
		Output:	None.
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Login name=“T0000002T”, password=“2220”, user type=“Teacher”.
		Reason:	The password is different from the stored one.
		Output:	Error code indicating that reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	The test 1 does not provide any output. Thus indicating everything works fine.		

Table A4.11: TC–06.02 Authenticate password.

TC–12	**Unit Test Cases for UC–06.03 (Validate Group).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–06.03 Validate group.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Date=“2010-11-29”, time=“11:10”, group=“103112006P01”, login name=“S0000001S”, user type=“Student”.
		Output:	Message showing the successful group validation.
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Date=“2010-11-29”, time=“11:10”, group=“103112006P02”, login name=“S0000001S”, user type=“Student”.
		Reason:	The group is not correct.
		Output:	Error code indicating the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	None.		

Table A4.12: TC–06.03 Validate group.

TC–13	**Unit Test Cases for UC–06.04 (Acquire Student Password).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–06.04 Acquire student password.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Login name="S0000001S" (student).
		Output:	"0001".
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Login name="00000000X" (invalid).
		Reason:	The system could not find the student's password.
		Output:	Error code indicating the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	None.		

Table A4.13: TC–06.04 Acquire student password.

TC–14	**Unit Test Cases for UC–06.05 (Acquire Teacher Password).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–06.05 Acquire teacher password.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Login name="T0000002T" (teacher).
		Output:	"0002".
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Login name ="00000000X" (invalid).
		Reason:	The system could not find the teacher's password.
		Output:	Error code indicating the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	None.		

Table A4.14: TC–06.05 Acquire teacher password.

TC–15	**Unit Test Cases for UC–06.06 (Acquire Student Group).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–06.06 Acquire student group.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Date="2010-11-29", time="11:10", login name="S0000001S" (student).
		Output:	"103112006P01".
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1, with date="1990-01-01".
		Reason:	No group code has been found.
		Output:	Error code indicating the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	None.		

Table A4.15: TC–06.06 Acquire student group.

TC–16	**Unit Test Cases for UC–06.07 (Acquire Teacher Group).**		
Version	1 (20/11/2010)		
Authors	• Alfonso De Gea.		
Sources	• Robert Langwieser (ITC). • Victoria Bueno (ETSIT).		
Related Use Case	• UC–06.07 Acquire teacher group.		
Related Objectives	• OBJ–01 Manage the registering of attendance.		
Related Requirements	• IRQ–01 Information about users. • IRQ–02 Information about groups.		
Test Cases	**Num.**		
	1	**Type:**	Normal.
		Values:	Date="2010-11-29", time="11:10", login name="T0000002T" (teacher).
		Output:	"103112006P01".
		Result:	Success.
	2	**Type:**	Exception.
		Values:	Same as test 1, with date="1990-01-01".
		Reason:	No group code has been found.
		Output:	Error code indicating the exception reason.
		Result:	Success.
Status	Passed.		
Stability	High.		
Comments	None.		

Table A4.16: TC–06.07 Acquire teacher group.

Appendix 5. JAVA FOR MOBILE DEVICES.

This appendix summarizes the Java mobile devices programming, since the platform development process uses Java as the coding language. The focus is on the issues related to the Nokia 6212 Classic mobile phone.

A5.1. INTRODUCTION.

This appendix discusses the application of Java Platform technologies on mobile programming, focusing on the N6212.

Java is distributed free of charge, under an open-source license, and is the OO programming language with the more widespread acceptance in the programmers' community. Figure A5.1 depicts the four different currently available Java editions in the Java Platform [WJAVA]:

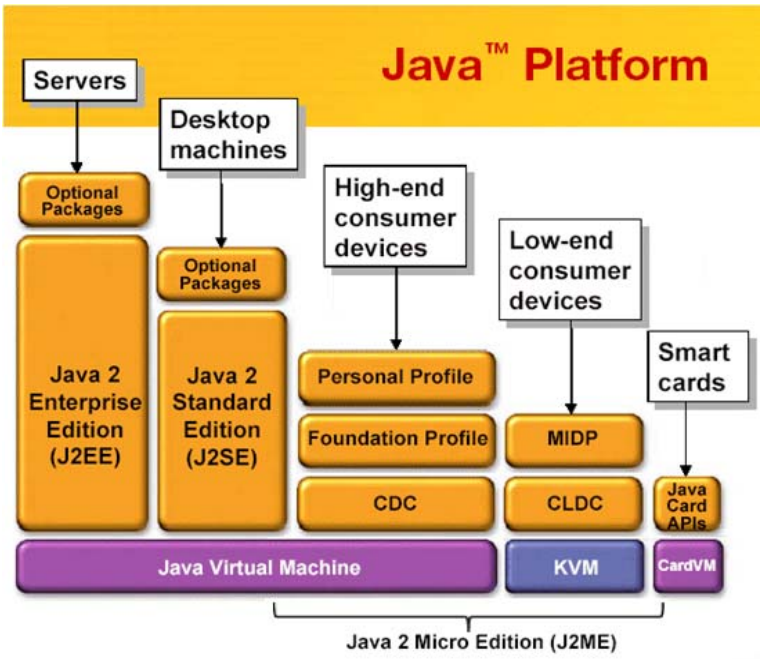


Figure A5.1: Java Platform, from [WJAVA].

1. Java 2 Standard Edition (J2SE) is aimed at developing and deploying Java applications on desktops and servers [WJ2SE].

2. Java 2 Enterprise Edition (J2EE) is used on servers to run web applications with special APIs for this purpose.

3. Java 2 Micro Edition (J2ME) focuses on embedded systems, or hand-held devices like mobile phones [WJ2ME].

4. Java Card Platform (JCP) enables Java technology to be used on smart cards.

The current versions are J2SE 6.23, J2EE 6, J2ME 3.0, and JCP 3.1 [WJAVA].

The following paragraph briefly summarizes the history of the inception and evolution of the Java Platform [WJAVA]:

- 1990: Java started as an internal project named “Oak” at Sun Microsystems.
- 1995: Initial release of Java 1.0, defining applets and servlets.
- 1997: Sun unveiled the JCP 2.0.
- 1999: JavaOne conference created a subdivision in the Java technologies: the Java 2 technology platform was born: J2SE, J2EE, and J2ME.
- 2000: First mobile phones with support for J2ME.
- 2006: Java technology is open-sourced by Sun Microsystems.
- 2009: J2SE reaches version 6.
- 2009: Oracle Corporation acquires Sun Microsystems. Java becomes an Oracle project.
- 2010: JCP 3.0 comes to light.

The following sections discuss the J2ME and JCP related APIs to the N6212.

A5.2. JAVA 2 MICRO EDITION.

The main APIs of J2ME are included in:

- Connected Limited Device Configuration (CLDC): basic API set.
- Mobile Information Device Profile (MIDP): CLDC extension API set.

J2ME and J2SE share the same programming language: Java, as stated in the CLDC 1.1 specification [JSR139]: “A CLDC implementation must be able to read Java class files in all the formats supported by J2SE up to version 1.4”.

However, the difference between J2SE and J2ME lies in their deployment issues and in their available APIs. J2ME replaces some J2SE APIs with optimised ones for mobile devices. For instance, APIs dealing with the UI, considering the different mobile input mechanisms and screen sizes.

Besides, additional APIs can be added at the mobile phone manufacturer’s discretion. These are so-called Original Equipment Manufacturer (OEM) APIs, such as some of Nokia APIs included in the N6212, discussed in the following sections.

An application written for J2ME compatible devices is called a “MIDlet” and is bundled in a “MIDlet suite”, which can contain one or more MIDlets. From the point of view of a MIDlet suite, each MIDlet can be compared to a standard J2SE Java class with a “main” method where the application execution can be started.

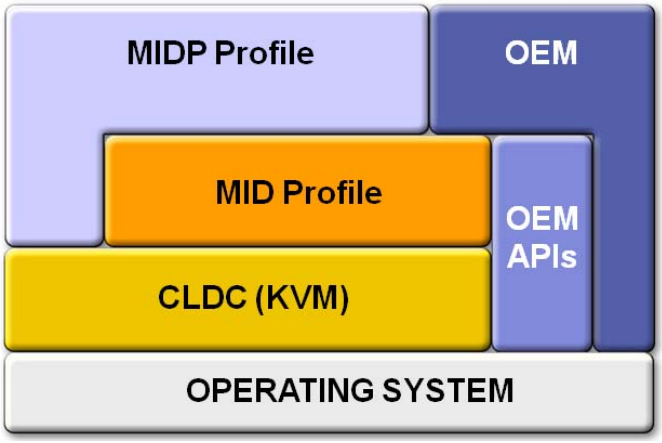


Figure A5.2: Java 2 Micro Edition.

Figure A5.2 depicts J2ME APIs which are described in the following sections, focusing on the mandatory APIs in a J2ME implementation: CLDC and MIDP.

A5.2.1. Connected Limited Device Configuration.

The CLDC API set specification is defined in the Java Specification Request (JSR) 139 [JSR139]. CLDC includes the base API set and a suitable JVM for resource-constrained devices such as mobile phones, or Personal Digital Assistants (PDAs). Hence, CLDC provides these devices with a basic Java system containing:

- Java language and the J2ME Kilobyte Virtual Machine (KVM): a suitable JVM for mobile devices.
- Core Java libraries: “java.lang.*” and “java.util.*”.
- Input/output library: “java.io.*”.
- Security libraries.
- Networking libraries.
- Internationalization libraries.

The current version of CLDC is 1.1, and the N6212 is compatible with it (see Appendix 8).

A5.2.2. Mobile Information Device Profile.

MIDP API set is a CLCD extension. Its specification is defined in JSR-118 [JSR118] where it is briefly described as: “The MIDP lets developers write downloadable applications and services for network-connectable mobile devices”. MIDP is built on top of CLDC, extending it by including new functionalities for mobile devices:

- Application delivery and billing.
- Application lifecycle, defining the semantics of a MIDP application and how it is controlled (see Section A5.4).
- Application signing model (see Section A5.4) and privileged domains security model.
- End-to-end transactional security (HTTPS protocol).
- MIDlet Push Registry.
- Networking: higher level network protocols.
- Persistent storage: RMS API extension (see Section A5.2.2.2).
- Sound.

- Timers.
- UI: display and input.

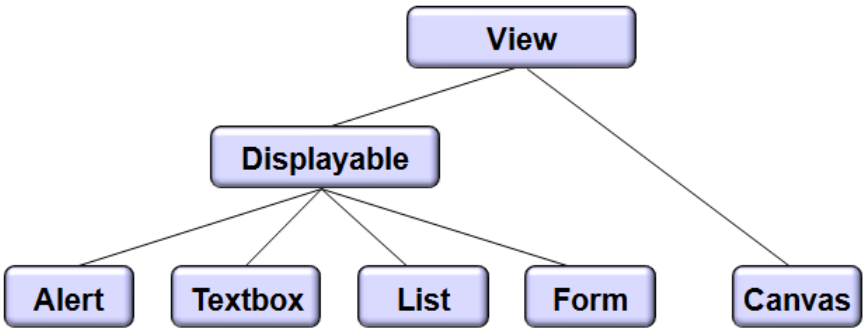


Figure A5.3: MIDP UI classes.

Figure A5.3 schematically depicts the MIDP UI classes: “Alert”, “TextBox”, “List”, and “Form”. The “Canvas” class is aimed at precisely managing the screen pixels.

Currently, two versions are in use: MIDP 2.0 and MIDP 2.1. N6212 is compatible with MIDP 2.1 (see Appendix 8).

The following sections present the MIDP extension APIs Personal Information Management (PIM) and FileConnection.

A5.2.2.1. Personal Information Management.

PIM API is an MIDP extension aimed at managing contacts, events, and to-do items on the internal phone memory or on the phone Subscriber Identity Module (SIM) card. The PIM specification is defined in JSR-75 [JSR75].

```
1   private Vector getContactList() throws PIMException {
2           PIM pim = PIM.getInstance();
3           // Get access to the SIM card
4           ContactList cL=(ContactList) pim.openPIMList(PIM.CONTACT_LIST,PIM.READ_WRITE,”SIM”);
5           Vector v = new Vector();
6           // Retrieve all the contacts
7           Enumeration e = cL.items();
8           while (e.hasMoreElements()) {
9                   Contact contact = (Contact) e.nextElement();
10                  int cV = contact.countValues(Contact.FORMATTED_NAME);
11                  for (int i = 0; i < cV; i++) {
12                          // Extract a contact name and add to the result vector
13                          String contactName = contact.getString(Contact.FORMATTED_NAME,i);
14                          v.addElement(contactName);
15                  }
16          }
17          return v;
18  }
```

Table A5.1: PIM example.

Table A5.1 contains an example code snippet which lists all contact names stored on the phone SIM card. Line 4 gets access to the SIM card and then, in line 7, the contact list is retrieved. The loop in line 8 is responsible for extracting the contact names.

A5.2.2.2. Record Management System.

The RMS provides a private and persistent storage space on the phone to MIDlet suites: a "record store". The RMS is part of the PIM API, as stated in MIDP 2.0 specification [JSR118].

A MIDlet can use the RMS to persistently store data such as application settings. However, a MIDlet is only allowed to access to those record stores property of its MIDlet suite.

```
1  // open or create the record store if not exist
2  RecordStore rs = RecordStore.openRecordStore("settings", true);
3  // store an integer (as a byte array)
4  byte[] s = String.valueOf(123456).getBytes();
5  // add a new record
6  rs.addRecord(s, 1, s.length);
7  // retrieve the first record
8  s = rs.getRecord(1);
```

Table A5.2: RMS example.

Table A5.2 contains an example code snippet which shows how to instantiate a record store, add some data to it and then, retrieve the data. The record store is opened or created when it does not exist yet.

The record store associates a unique identifier to each stored data. When a new record is added its identifier is increased by one. The identifier of a deleted record is never used again.

A5.2.2.3. FileConnection.

FileConnection specification is defined in JSR-75 [JSR75]. This is an optional MIDP API aimed at managing files on the phone's internal file system and/or on the phone's removable memory card. However, the API is not allowed to access to "special" files such as MIDlet suite files or record stores (see Section A5.2.2.2).

Table A5.3 shows an example code snippet to list all the available root directories on the phone.

```
1  Enumeration e = FileSystemRegistry.listRoots();
2  while (e.hasMoreElements()) {
3          String root = (String) e.nextElement();
4          // The String "root" contains the name of the file system root
5  }
```

Table A5.3: FileConnection example 1.

Table A5.4 shows a "Hello-World" example code snippet which writes a file on the "E:" drive (usually a removable memory card).

```
1  FileConnection fc = (FileConnection) Connector.open("file:///E:/file.txt")
2  if (!fc.exists()) fc.create()
3  PrintStream ps = new PrintStream(fc.openOutputStream())
4  ps.println("Hello World")
5  ps.flush()
6  ps.close()
```

Table A5.4: FileConnection example 2.

A MIDlet intended to access the file system has to be signed and granted in advance with the following permissions (see Section A5.4.1):

- Read access: "`javax.microedition.io.Connector.file.read`".
- Write access: "`javax.microedition.io.Connector.file.write`".

A5.3. SECURE ELEMENT.

The N6212 SE related APIs are:

- Contactless Communication API: is standardized in JSR-257 [JSR257] and provides access to the phone SE.
- Contactless Communication Extension API: Nokia proprietary (OEM) API which provides the NF6212 with access to the phone's NFC functionality and with extended access to the SE [WFNLIB].

The N6212 SE contains a tamper resistant NFC controller chip, allowing the phone to act as:

1. An NFC smart card reader.
2. An NFC smart card: a JCP smart card, or the so-called "card emulation mode" which acts as a MiFare smart card.

3. An NFC P2P device: using the NFCIP protocol (see Chapter 2).

A MIDlet intended to access the SE has to be signed and granted in advance with the following permission (see Section A5.4.1):

- SE access: "`javax.microedition.apdu.aid`" (see Section A5.4.1).

The APIs related with the SE are of no application in this thesis (see Section 3.6.5.1 in Chapter 3 for a further explanation); hence, they have just been briefly described.

A5.4. MIDLET SUITE CONSTRUCTION.

A MIDlet suite is a collection of one or more MIDlets together in a Java Archive (JAR) file, and accompanied by an optional Java Application Descriptor (JAD) file.

A MIDlet is a Java class that extends the abstract class "`javax.microedition.midlet.MIDlet`" which gives the phone operating system the ability to control the MIDlet status: "active", "paused", "destroyed" (see Figure A5.4). A MIDlet can be seen as the entry point of execution of a MIDlet suite, like the "main" method in J2SE applications.

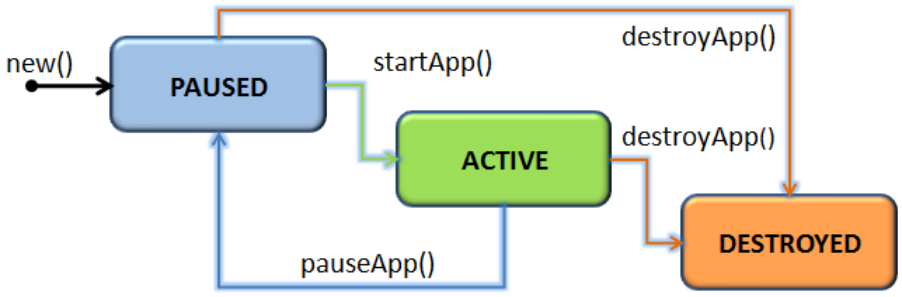


Figure A5.4: MIDlet lifecycle.

The MIDlet suite construction consists of the following steps:

1. Compile the MIDlet suite source code to Java classes. This implies some differences in comparison with compiling J2SE applications:

   - The CLDC and MIDP APIs, and other possible extension APIs, are part of the application class path instead of the regular J2SE runtime classes.

- The Java compatibility version has to be minor 1.4. CLCD 1.1 compliant devices, such as N6212, can use version 1.4.

2. Preverify the classes. The preverification modifies the byte code in the class files to improve the performance in the resource-constrained devices.

3. Create the manifest comprising at least the following information [WJ2SE]:

   - MIDlet-Name: contains the name of the MIDlet suite.

   - MIDlet-Version: contains the version number of the MIDlet suite.

   - MIDlet-Vendor: contains the vendor name of the MIDlet suite.

4. Package the preverified classes and the manifest in a JAR archive.

5. Create the application descriptor JAD file. The JAD file must contain the following entries [JSR118]:

   - MIDlet-Version: Contains the version number of the MIDlet suite.

   - MIDlet-Vendor: Contains the vendor name of the MIDlet suite.

   - MIDlet-Jar-URL: Contains the link to the JAR archive file.

   - MIDlet-<n>: Contains a comma separated list with the name of the MIDlet, the optional file name of the icon, and the class name of the MIDlet. <n> is incremented for each MIDlet belonging to the suite.

   - MicroEdition-Profile: Contains the MIDP version number. This is usually either MIDP-2.0 or MIDP-2.1.

   - MicroEdition-Configuration: Contains the CLDC version number.

   - MIDlet-Name: Contains the name of the MIDlet suite.

Table A5.5 contains an example JAD file gathering the described entries.

```
1  MIDlet-Version: 1.0.0
2  MIDlet-Vendor: UPCT
3  MIDlet-Jar-URL: StudentNFCClient.jar
4  MicroEdition-Configuration: CLDC-1.1
5  MicroEdition-Profile: MIDP-2.1
6  MIDlet-1: Student NFC Client,,es.upct.teleco.nfc.client.student.NFCClient4Student
7  MIDlet-Name: Student NFC Client
```

Table A5.5: Java Application Descriptor example.

5.1. Add the permissions required by the MIDlets (in signed applications only).

5.2. Sign the suite by adding the JAR file signature and certificate chain (optional).

6. Deploy the MIDlet suite. There are a few ways to transfer MIDlet suites to a mobile phone:

   - Using a Bluetooth or a USB connection.
   - Using a memory card.
   - Internet provisioning, downloading the MIDlet suite on the phone.

A5.4.1. Permissions.

Signing a MIDlet suite is mandatory when the suite performs certain sensitive operations, such as accessing the file system (using JSR-75), or accessing the SE (using JSR-257). In these cases, the suit requires special permissions which can only be granted if the suite is signed. Failing to request these permissions will result in a "SecurityException" and the suit is not allowed to perform the required task.

Permission	Description
`javax.microedition.io.Connection.file.read`	Request general read permission: file system and address book.
`javax.microedition.io.Connector.file.write`	Request general write permission: file system and address book.
`javax.microedition.io.PushRegistry`	Request permission to use the Push Registry.
`javax.microedition.apdu.aid`	Request permission to open an APDU connection (with the SE).

Table A5.6: MIDlet suite permissions.

A list of some permissions and their description can be found in Table A5.6. For a complete MIDlet suite permission list see [JSR118].

Appendix 6. USER INTERFACE FUNCTIONALITIES.

This appendix details the UI functionalities, associating them with their respective functional requirements. The association enhances keeping the trace of the implemented requirements. The system UI is presented in three versions, corresponding to the three distinguished system user types: student, teacher, and administrator.

A6.1. COMMON FUNCTIONALITIES.

The system interacts with three distinguished human actors: the student, the teacher, and the administrator (see Section A3.3.2 in Appendix 3). Both the teacher's and the student's interaction with the system is based on NFC mobile phones, as stated in Section 3.6 in Chapter 3. However, different functionalities are dedicated to these actors; hence, the usable UI design presents three different versions, depending on the user (see Section 4.1 in Chapter 4).

Figure A6.1: UI main menu.

Figure A6.1 depicts the UI main menu, which follows an "action menu" screen pattern (see Section 4.1 in Chapter 4). Student and teacher versions present common functionalities:

- "Configure User Information": this UI functionality implements the functional requirement UC–01 Configure user information (see Table A3.14 in Appendix 3).

- "Configure Language": this functionality is aimed at improving the UI Usability (see Section 4.1 in Chapter 4).

Figure A6.2 depicts the “Configure Language” menu, which also follows an “action menu” screen pattern. The UI can be presented in three languages: Spanish, English and German. The user’s preference is persistently stored for further usage.

Figure A6.2: Configure Language.

Figure A6.3 depicts the “Configure User Information” screen, which follows an “edit” pattern. The UI stores the user’s credentials to be able to automatically provide the server with them; hence, this feature improves the UI Usability (Section 4.1 in Chapter 4).

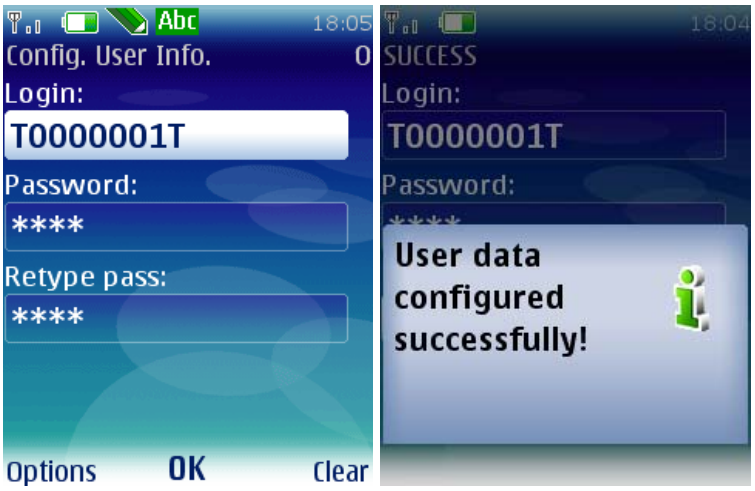


Figure A6.3: Configure User Information.

The UI controls the proper input of the user’s credentials by implementing the required restrictions checks (see Section A3.2 in Appendix 3):

- CRQ–01 User login-name value: which has to meet the pattern “L9999999A”, where ‘L’ stands for an alphanumeric character, ‘9’ for a digit, and “A” for an alphabetical character.

- CRQ–02 User password value: the student password is formed by four digits, and the teacher password is formed by four to fifteen alphanumeric characters.

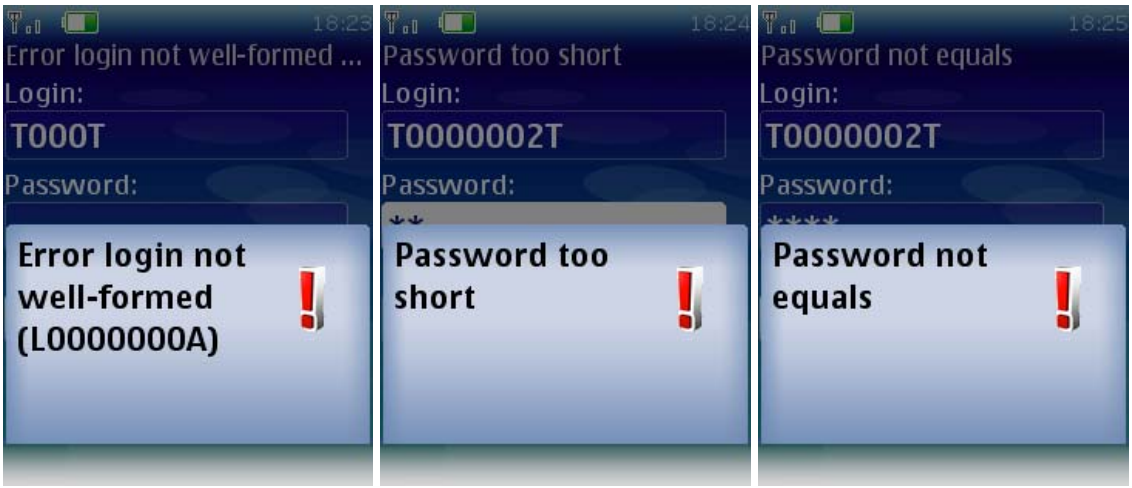


Figure A6.4: User's credentials restriction checks.

In addition to the previous checks, the user has to retype the password to avoid possible typing mistakes. Figure A6.4 depicts the possible warning conditions that can occur during the user information configuration process.

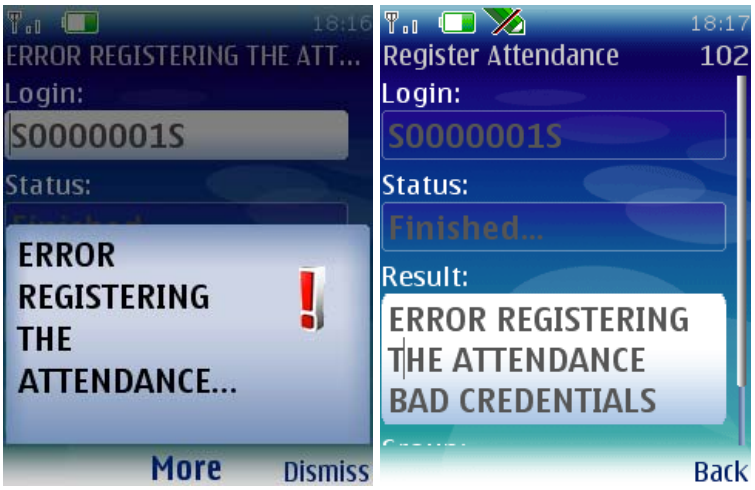


Figure A6.5: System warnings.

The system monitors undesired operations: *i)* incorrect user parameters, such as bad credentials, *ii)* an incoming user request with an unrecognised command, meaning a recoverable application protocol error, *iii)* an abnormal operation due to external system disconnections, or *iv)* an unexpected error. Figure A6.5 depicts the case i), and Figure A6.6 depicts the cases ii), iii) and iv), from left to right respectively.

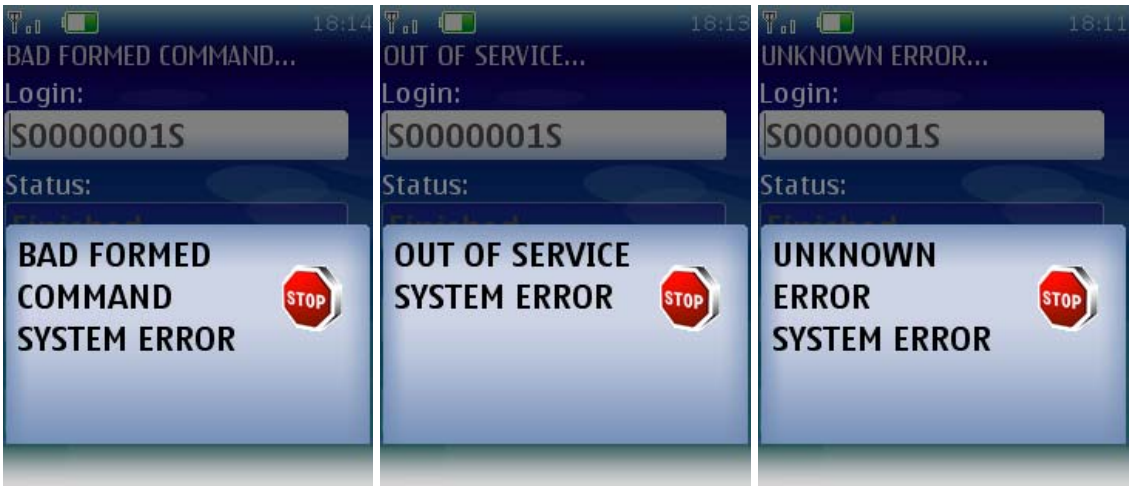


Figure A6.6: System errors.

The error cases are defined in the application protocol design (see Table 4.5 in Chapter 4).

The icons meaning “info” (or “success”), “warning”, and “error” are defined in Section 4.1 in Chapter 4.

A6.2. STUDENT VERSION.

The left side of Figure A6.7 depicts the student UI main menu. Once the student provides the UI with well-formed credentials, in addition to the common functionalities the UI shows another: “Register Attendance”.

Figure A6.7: Register Attendance.

This functionality implements the functional requirement UC–02 Register attendance (see Table A3.15 in Appendix 3).

When the student selects this functionality the UI shows the "Register Attendance" screen (in the middle of Figure A6.7), which uses an "information" pattern (see Section 4.1 in Chapter 4). When the attendance is properly registered, the system shows a "success" message in the "Result" field (right side of Figure A6.7). During the operation, the system can find some exception cases (see Section 4.5 in Chapter 4), informing the student like depicted by Figure A6.5.

Further application executions highlights the "Register Attendance" option by default, which favours meeting the "Minimal action" criteria of the usability model applied (see Section 4.1 in Chapter 4).

A6.3. TEACHER VERSION.

Left side of Figure A6.8 depicts the teacher UI main menu. Once the teacher provides the UI with well-formed credentials, two new UI functionalities are added to the common ones: *i)* "Activate Group", implementing the functional requirement UC–03 Activate group (see Table A3.16 in Appendix 3); and *ii)* "List attendees", implementing the functional requirement UC–04 List attendees (see Table A3.17 in Appendix 3).

When the teacher selects i), the UI shows the "Activate Group" screen. When the group is properly activated, the system shows a "success" message in the "Result" field.

Figure A6.8: Activate Group.

When the teacher selects ii), the UI shows the "List Attendees" screen depicted in Figure A6.9.

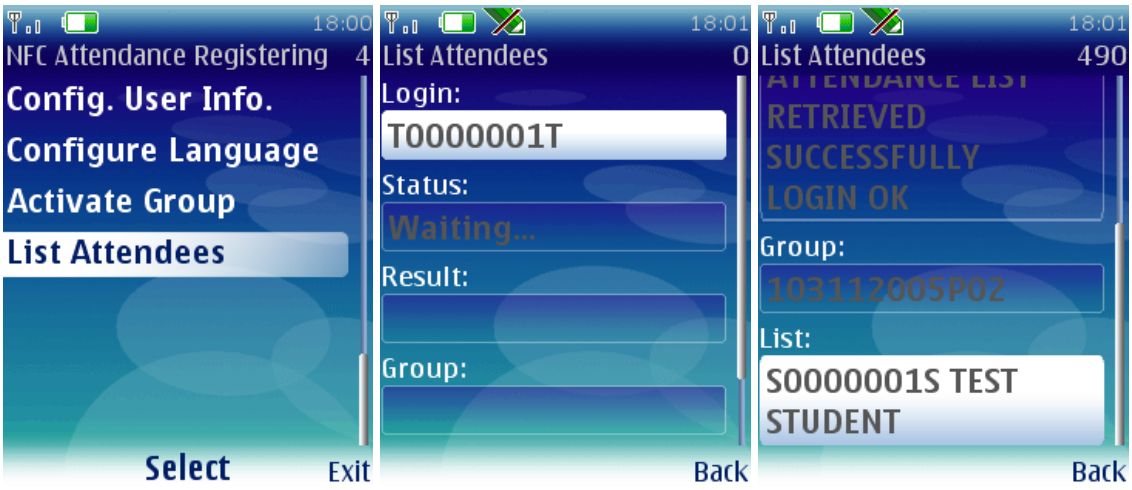


Figure A6.9: List Attendees.

When the attendance list is properly retrieved, the system shows a "success" message in the "Result" field. The "List" field gathers the login-name and the name of the attendees.

Both "Activate Group" and "List Attendees" screens use an "information" pattern (see Section 4.1 in Chapter 4). During both operations, the system can find any exception case (see Section 4.5 in Chapter 4), informing the teacher like depicted by Figure A6.5.

Further application executions highlights the "Activate Group" option by default, which favours meeting the "Minimal action" criteria of the usability model applied (see Section 4.1 in Chapter 4).

A6.4. ADMINISTRATOR VERSION.

The administrator UI functionalities are provided by means of MySQL Workbench [WMYSQL].

Figure A6.10 depicts the functionality provided by the UI corresponding to the functional requirement UC–07 Configure groups scheduling (see Table A3.20 in Appendix 3). The administrator introduces new records with scheduled laboratory practices or lectures. The red circle highlights the "save button".

When data of a new or updated record overlaps the previous scheduling, the system informs the user with the following error message: "Cannot schedule overlapped groups", aborting the operation (see Figure A6.11).

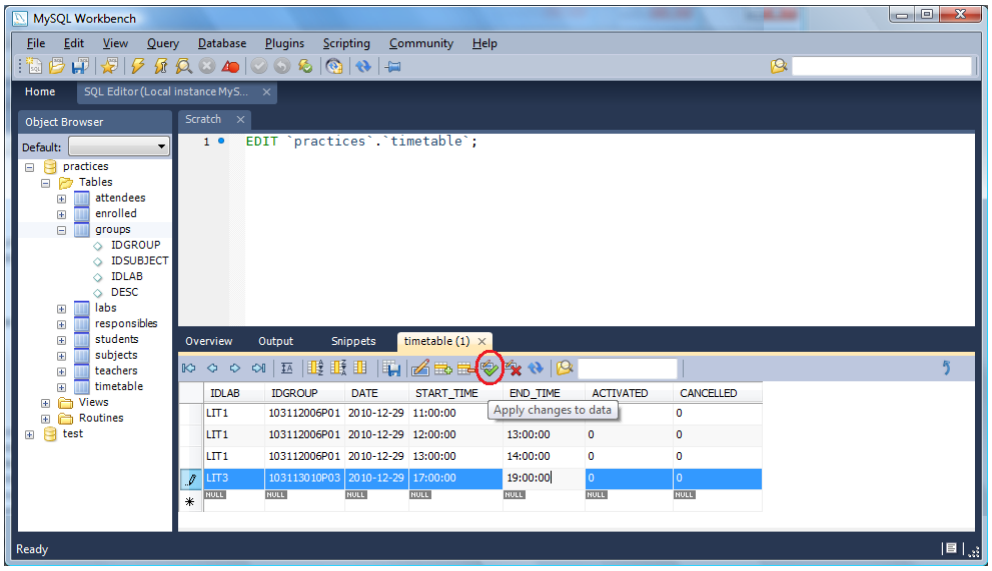


Figure A6.10: Configure Groups Scheduling.

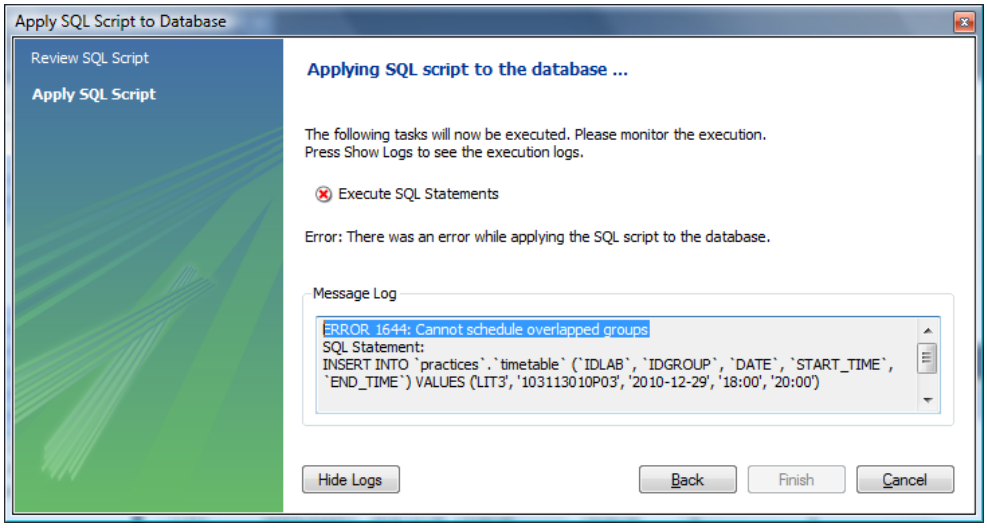


Figure A6.11: Overlapped group error.

Figure A6.12 depicts the functionality provided by the UI corresponding to the cancellation of a group session scheduled in advance. The administrator performs this task by means of updating the “cancelled” field to 1 (boolean “true”). Red circles highlight the important areas. The functional requirement implemented is UC–08 Cancel group (see Table A3.21 in Appendix 3).

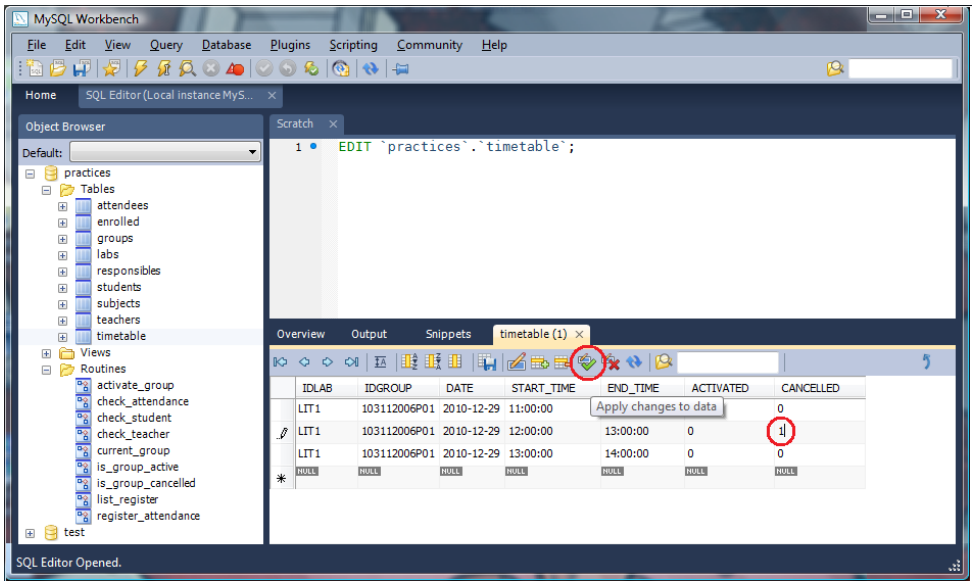


Figure A6.12: Cancel Group.

The UC–09 Check system logs requirement (see Table A3.22 in Appendix 3) is implemented by means of the Log4j Java library (see Section 3.6.5.4 in Chapter 3). The administrator should inspect the system logs on a regular basis. Just using a text editor is required. Figure A6.12 depicts the activity log, where the user requests are recorded.

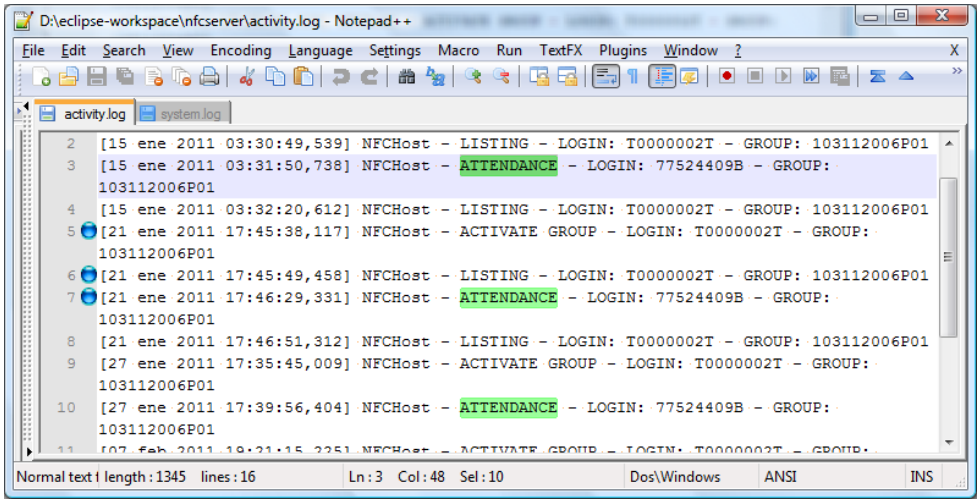


Figure A6.13: Activity log.

In addition to the activity log, the system log store information about the system operation, such as debug or error messages (see Figure A6.14). Checking this log on a regular basis helps the administrator to know the actual system status or even to identify potential problems, such as continuous external systems disconnections. During the commissioning phase can be as useful as it is during the development process to keep the trace of the systems steps.

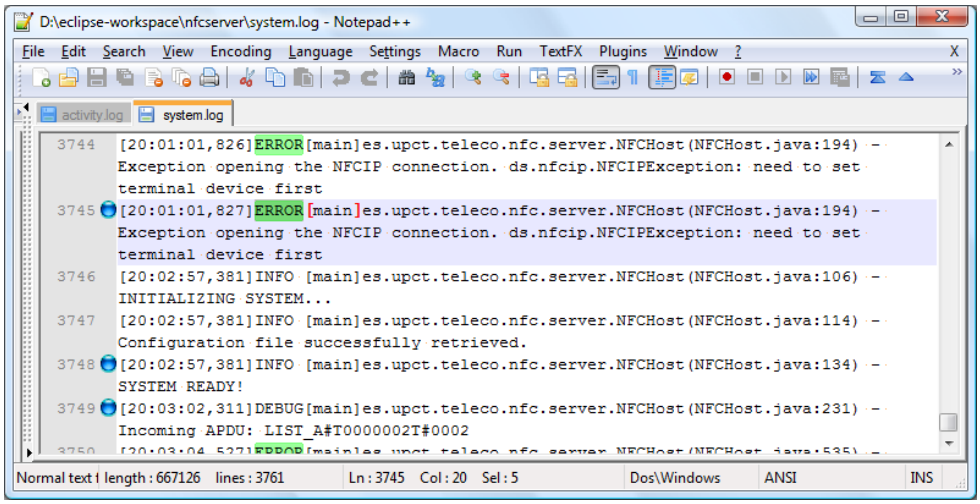


Figure A6.14: System log.

Appendix 7. ACS-ACR122U TECHNICAL SPECIFICATIONS.

This appendix summarizes the ACS-ACR122U NFC reader technical specifications extracted from [WACR122U].

ACS-ACR122U NFC Reader Technical Specifications.		
USB interface	Power source	From USB
	Speed	12 Mbps (USB Full Speed)
	Supply voltage	Regulated 5 V DC
	Supply current	200 mA (maximum); 50 mA (standby); 100 mA (normal)
Contactless smart card interface	Standard	ISO/IEC 18092 NFC, ISO 14443 A and B, FeliCa, MiFare,
	Protocol	FeliCa protocol, T=CL protocol
	Operating frequency	13.56 MHz
	Smart card read/write speed	106 kbps, 212 kbps, 424 kbps
Casing	Dimensions	98 mm (L) x 65 mm (W) x 12.8 mm (H)
	Weight	70 grams
	Material	Polycarbonate (PC)
	Color	Pearl White
	Antenna size	50 mm x 40 mm
	Operating distance	Up to 50 mm (depending on tag type)
Built-in peripherals	Bi-color LED	Red and green
	Buzzer	Monotone (optional)
Operating conditions	Temperature	0 - 50º C
	Humidity	10% - 80%
Cable connector	Length	1.0 m (USB)
Certifications/compliance	PC/SC, CCID, CE, FCC, VCCI, RoHS Compliant, USB Full Speed. Microsoft WHQL 2000, Server 2003, XP, Vista, Server 2008, 2008 R2, 7	
Device driver operating system support	Windows 98, ME, 2000, Server 2003, XP, Vista, Server 2008, 2008 R2, 7 Windows CE 5.0 Linux, Mac	

Table A7.1: ACS-ACR122U NFC reader technical specifications.

Appendix 8. NOKIA 6212 CLASSIC TECHNICAL SPECIFICATIONS.

This appendix summarizes the Nokia 6212 Classic phone technical specifications extracted from [WNE6212].

Nokia 6212 Classic Technical Specifications.		
General	Form factor	Block
	Antenna type	Internal
	SAR Value	0.790 W/Kg
Size	Weight	88.0 g (with battery)
	Dimensions	114.7 x 47.1 x 14.5 mm
Connectors	Micro USB 2.0	
	2.5mm AV	
	microSD slot	
Operating frequency	GSM/EDGE	850/900/1800/1900
	WCDMA (UMTS)	850/2100
Battery power management	Type	Li-Ion
	Amperage	1000 mAh
	Standby time	GSM: 300h / 3G: 300h
	Talk time	GSM: 3h 20m / 3G: 2h 45m
Communications interfaces	GPRS multi-slot class 10	4+1/3+2 slots, 32-48 kbps
	EDGE multi-slot class 10	236.8 kbps
	3G (UMTS)	384 kbps
	WAP	
	Near Field Communication	read/write/sharing
	Bluetooth	2.0
	USB	2.0
Java compatibility	J2ME CLCD	1.1
	J2ME MIDP	2.0, 2.1
Display	Type	QVGA 16 M colours, TFT
	Size	2.00 inch
	Resolution	240 x 320 pixels
Memory	Numbers in phone	Up to 2000
	Received calls	Up to 20
	Outgoing calls	Up to 20
	Lost calls	Up to 20
	Internal shared memory	22 MB
	Memory card (optional)	microSD (TransFlash), up to 4 GB
Browsing	XHTML browser over TCP/IP	
	WAP 2.0	
	Opera Mini browser	Pre-installed
Messaging	SMS	
	MMS	1.2
Email client	Push E-Mail	
	Instant Messaging	
	Push to talk	
Built-in camera	CMOS sensor	2 M pixel
	Resolution	Up to 1600 x 1200 pixels
	Flash	LED
	Secondary	CIF camera
Ringtones	Polyphonic, MP3, AAC	
Miscellaneous features	Predictive text T9	
	Vibration	
	Hands-free	
	FM Radio	
	MP3/MPEG4 player	

Table A8.1: Nokia 6212 Classic technical specifications.

Appendix 9. GLOSSARY.

This appendix gathers an alphabetically ordered list of used terminology by this thesis, giving a short description of the terms, thus improving the comprehensibility of the reading.

A

Agile Methodologies: are the software development methodologies focussed on the individual, the collaboration with customers, and the incremental development of software with very short iterations, thus speeding up the software development process.

API: is a software library which offers access to services provided by software components or hardware devices; hence an API serves as an access interface to this devices or components.

M

MIDlet: is an application that conforms to the MIDP standard.

MIDlet suite: a collection of MIDlets packaged into a JAR file. The suite also contains a JAD file describing the suite.

MIDP: This is a set of J2ME APIs that define how software applications interface with cellular phones.

N

NFC-enabled: refers to those devices, such as mobile phones, PDAs, or laptops, which have a built-in NFC chip providing them with the NFC technology.

O

Open-source license: is a license aimed primarily at protecting the free distribution, modification, and use of software. The license agreement uses to demand expressly forbidden commercial use of the software, and the inclusion of references about its authoring. There is a wide range of open-source licenses such as: GNU-General Public License (GPL), Apache License, or Nokia Open Source License. As a curiosity, the GNU acronym is recursively defined as “GNU is not UNIX”.

P

Push Registry: internal mechanism of the mobile phones which is intended to execute phone applications when external events occur.

R

Regular expression: in computing, a regular expression, also referred to as "regexp", provides a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters.

S

Secure Element: a tamper resistant chip inside the NFC-enabled phones. This chip allows these phones to act as an NFC reader or as a smart card.

Smart poster: a poster with an embedded chip (a passive tag) that makes the poster able to offer some kind of service or information. The chip communicates with a reader by RF.

U

Usability: is a term which refers to a product quality that implies easy of use, understanding, and learning by the end user.

Usability model: is a quality model intended to be applied to a development process in order to achieve usable products.

X

XP: is an agile methodology focussed on promoting interpersonal relations as the key to success in software development. XP is especially aimed at those projects with small development teams, with short deadlines, volatile requirements, and/or based on new technologies.

Appendix 10. ACRONYMS.

This appendix gathers an alphabetically ordered list of spread acronyms over this thesis, indicating what they stand for, thus enhancing the understandability of the reading.

3

3G Third Generation (related to UMTS).

A

APDU Application Protocol Data Unit.

API Application Programming Interface.

ASK Amplitude Shift Keying.

B

BBT Black-Box Testing.

BNF Backus-Naür Form.

BTP Binary Tree Protocol.

C

CAPDU Command APDU.

CDC Connected Device Configuration.

CLDC Connected Limited Device Configuration.

CMMI Capability Maturity Model Integration.

COCOMO Cost Constructive Model.

CRQ Check Requirement (related to RE).

CS Carrier Sense.

D

DB Database.

DBMS DB Management System.

E

ECMA European Computer Manufacturers Association.

EFQM European Foundation for Quality Management.

EHEA European Higher Education Area.

ETSI European Telecommunications Standards Institute.

ETSIT Escuela Técnica Superior de Ingeniería Telemática (related to UPCT).

G

GIT Grupo de Ingeniería Telemática (related to ETSIT).

GPL General Public License.

GSM Global Systems Mobile.

GUI Graphical UI.

H

HCI Human Computer Interaction.

I

IDE Integrated Development Environment.

IEC International Electrotechnical Commission (related to ISO).

INTHFT Institut für Nacrichtentechnik und Hochfrequenztecnik (related to TUW).

IRQ Information Requirement (related to RE).

ISM Industrial, Scientific and Medical.

ISO International Organization for Standardization.

IT Information Technologies.

ITC Institute of Telecommunications (formerly known as INTHFT).

J

J2EE Java 2 Enterprise Edition.

J2ME Java 2 Micro Edition.

J2SE Java 2 Standard Edition.

JAD Java Application Descriptor.

JAR Java Archive.

JCP Java Card Platform.

JSR Java Specification Requests.

JVM Java Virtual Machine.

K

Kbps Kilobits per second.

KVM Kilobyte Virtual Machine.

L

LAN Local Area Network.

LDAP Lightweight Directory Access Protocol.

LLCP Logical Link Control Protocol.

M

MAN Metropolitan Area Network.

MIDP Mobile Information Device Profile.

N

NDEF NFC Data Exchange Format.

NFC Near Field Communication.

NFCIP NFC Interface and Protocol.

NFR Non-Functional Requirement (related to RE).

O

OBJ Objective (related to RE).

OEM Original Equipment Manufacturer.

OO Object-Oriented.

OOA Object-Oriented Analysis.

OOD Object-Oriented Design.

OU Organizational Unit (related to LDAP).

P

P2P Peer-to-Peer.

PC Personal Computer.

PCD Proximity Coupling Device.

PDA Personal Digital Assistant.

PICC Proximity Integrated Circuit Card.

PIM Personal Identification Management.

R

RAPDU Response APDU.

RE Requirements Elicitation.

REMSS RE Methodology for Software Systems.

RF Radio Frequency.

RFC Request For Comments.

RFID RF Identification.

RM Relational Model.

RMS Record Management System.

RTD Record Type Definition.

S

SDK Software Development Kit.

SE Secure Element.

SIM Subscriber Identity Module.

SPICE Software Process Improvement and Capability Determination.

SQL Structured Query Language.

SSHA Seeded Signature Hash Algorithm.

T

TC Test Case.

TDD Test Driven Development.

TQM Total Quality Management.

TUW Technische Universität Wien.

TV Television.

U

UC Use Case (related to RE).

UI User Interface.

UML Unified Modeling Language.

UMTS Universal Mobile Telecommunications System.

UPCT Universidad Politécnica de Cartagena.

URL Uniform Resource Locator.

USB Universal Serial Bus.

UX User Experience.

V

VCD Vicinity Coupling Device.

X

XP Extreme Programming.

Appendix 11. BIBLIOGRAPHY AND REFERENCES.

[AS2004] Scott W. Ambler. "The Object Primer: Agile Model-Driven Development with UML 2.0". Cambridge University Press, 2004.

[AS2005] Scott W. Ambler. "The Elements of UML 2.0 Style". Cambridge University Press, 2005.

[ASRW2002] P**.** Abrahamsson, O. Salo, J. Ronkainen and J. Warsta. "Agile Software Development Methods: Review and Analysis". VTT Publications, 2002.

[BD2006] Genevieve Bell and Paul Dourish. "Yesterday's Tomorrows: Notes on Ubiquitous Computing's Dominant Vision". Article included in the book "Personal and Ubiquitous Computing". Springer-Verlag, 2006.

[BF1995] Frederick Brooks. "The Mythical Man-Month: Essays on Software Engineering". Addison-Wesley, 1995.

[BK2000] Kent Beck. "Extreme Programming Explained: Embrace Change". Pearson Education, 2000.

[BKM1991] Niegel Bevan, J. Kirakowsky and J. Maissel. "What is Usability?". Proceedings of fourth International Conference on Human Computer Interaction, 1991.

[BM1994] Niegel Bevan and M. Macleod. "Usability Measurement in Context". National Physical Laboratory, Teddington, Middlesex, UK. Behaviour and Information Technology, 1994.

[BPDG2011] María Victoria Bueno Delgado, Pablo Pavón Mariño and Alfonso Diego De Gea García. "NFC Technology and its Application in a University Environment". Espacio-Teleco magazine, second number, 2011.

[BRJ1999] Grady Booch, James Rumbaugh and Ivar Jacobson. "The Unified Modeling Language User Guide". Addison-Wesley, 1999.

[BVEG2009] María Victoria Bueno Delgado, Javier Vales Alonso, Enrique Egea López and Joan García Haro. "Radio-Frequency Identification Technology: Handbook of Enterprise Intregration". Auerbach Publications, CRC Press, pp. 429-466, 2009.

[BVG2009] María Victoria Bueno Delgado, Javier Vales Alonso and Francisco José González Castaño. "Analysis of DFSA Anti-collision Protocols in Passive RFID Environments". 35th International Conference of the IEEE Industrial Electronics Society, pp. 2610-2617, 2009.

[CB1998] A. P. Chandrakasan and R.W. Brodersen. "Low-Power CMOS Design". Wiley-IEEE Press, 1998.

[CJ2002] Jim Conallen. "Building Applications with UML". Addison-Wesley, 2003.

[CLD1999] P. Coad, E. Lefebvre and J. De Luca. "Java Modeling In Color With UML: Enterprise Components and Process". Prentice Hall, 1999.

[CLP2005] José H. Canós, Patricio Letelier and María Del Carmen Penadés. "Agile Methodologies for Software Development". Departamento de Sistemas Informáticos de la Universidad Politécnica de Valencia, 2005.

[DB2002] Amador Durán Toro and Beatriz Bernárdez Jiménez. "Requirements Elicitation Methodology for Software Systems. Version 2.3". Escuela Técnica Superior de Ingeniería Informática de Sevilla, 2002.
URL: http://www.lsi.us.es/~amador/publicaciones/metodologia_elicitacion_2_3.pdf.zip

[DC2003] C. J. Date. "An Introduction to Database Systems (8th Edition)". Pearson Education, 2003.

[DT2003] Martin Dougiamas and Peter C. Taylor. "Moodle: Using Learning Communities to Create an Open Source Course Management System". ED-MEDIA: World Conference on Educational Multimedia Hypermedia and Telecommunications, Honolulu, Hawaii, USA, 2003.
URL: http://dougiamas.com/writing/edmedia2003/

[ECMA340] Standard ECMA-340. Near Field Communication Interface and Protocol 1 (NFCIP-1).
URL: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-340.pdf

[ECMA352] Standard ECMA-352. Near Field Communication Interface and Protocol 2 (NFCIP-2).
URL: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-352.pdf

[ECMA356] Standard ECMA-356. NFCIP-1 RF Interface Test Methods. URL: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-356.pdf

[ECMA362] Standard ECMA-362. NFCIP-1 Protocol Test Methods. URL: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-362.pdf

[FBB1999] M. Fowler, K. Beck and J. Brant. "Refactoring: Improving the Design of Existing Code". Addison-Wesley, 1999.

[FK2003] Klaus Finkenzeller. "RFID-Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. Second edition". John Wiley & Sons Ltd, 2003.

[GHJV1995] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley, 1995.

[GPSEMM] GlobalPlatform's Proposition for NFC Mobile: Secure Element Management and Messaging. URL: http://www.globalplatform.org/documents/GlobalPlatform_NFC_Mobile_White_Paper.pdf

[HB2006] Ernst Haselsteiner and Klemens Breitfuss. "Security in Near Field Communication". Philips Semiconductors. Workshop on RFID Security RFIDSec, 2006.

[HG2008] Gerhard Hancke. "Eavesdropping Attacks on High-Frequency RFID Tokens". Workshop on RFID Security RFIDSec, 2008.

[HNS1999] C. Hofmeister, R. L. Nord and D. Soni. "Describing Software Architecture with UML". Proceedings of the First Working IFIP Conference on Software Architecture. Kluwer Academic Publishers, 1999.

[HO2000] J. Highsmith and K. Orr. "Adaptive Software Development: A Collaborative Approach to Managing Complex Systems". Dorset House, 2000.

[JTSM2007] P. Jaring, V. Törmänen, E. Siira and T. Matinmikko. "Improving Mobile Solution Workflows and Usability Using Near Field Communication Technology". Springer-Verlag, 2007.

[JSR118] JSR-118: Mobile Information Device Profile (MIDP) API. URL: http://jcp.org/en/jsr/detail?id=118

[JSR139] JSR-139: Connected Limited Device Configuration (CLDC) API.
URL: http://jcp.org/en/jsr/detail?id=139

[JSR257] JSR-257: Contactless Communication API.
URL: http://jcp.org/en/jsr/detail?id=257

[JSR75] JSR-75: Personal Information Management (PIM) API.
URL: http://jcp.org/en/jsr/detail?id=75

[KH2007] Adam Kolawa and Dorota Huizinga. "Automated Defect Prevention: Best Practices in Software Management". Wiley-IEEE Computer Society Press, 2007.

[LL2008] Lu Luo. "Designing Energy and User Efficient Interactions with Mobile Systems". PhD Thesis. School of Computer Science, Institute for Software Research, Carnegie Mellon University, 2008.

[LSG2009] J. Langer, C. Saminger and S. Grunberger. "A Comprehensive Concept and System for Measurement and Testing Near Field Communication Devices". IEEE Region 8 Conference EUROCON, pp. 2052-2057, 2009.

[MF2005] Francisco Montero. "Integración de Calidad y Experiencia en el Desarrollo de Interfaces de Usuario Dirigida por Modelos". PhD thesis. Universidad Politécnica de Castilla-La Mancha, 2005.

[MG1979] Glenford J. Myers. "The Art of Software Testing". John Wiley and Sons, 1979.

[NJ1993] Jakob Nielsen. "Usability Engineering". Academic Press Professional, 1993.

[ML1986] M. Marcotty and H. Ledgard. "The World of Programming Languages". Springer-Verlag, 1986.

[NT2006] Robert L. Nord and James E. Tomayko: "Software Architecture-Centric Methods and Agile Development". IEEE Software, volume 23, number 2, 2006.

[OG2001] C. Enrique Ortiz and Eric Giguère. "The Mobile Information Device Profile for Java 2 Micro Edition: Professional Developer's Guide". John Wiley and Sons Limited, 2001.

[PJ1994] Jacob Preece. "Human-Computer Interaction". Academic Press Professional, 1994.

[PP2003] M. Poppendieck and T. Poppendieck. “Lean Software Development: an Agile Toolkit for Software Development Managers”. Addison-Wesley, 2003.

[QW2001] W. Quesenbery. “What Does Usability Mean? Looking Beyond ‘Ease of Use’”. Proceedings of the 48th Annual Conference, Society for Technical Communication, 2001.
URL: http://www.wqusability.com/articles/more-than-ease-of-use.html

[RFC2307] RFC-2307: “An Approach for Using LDAP as a Network Information Service”.
URL: http://tools.ietf.org/html/rfc2307

[RFC4510] RFC-4510: Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map.
URL: http://tools.ietf.org/html/rfc4510

[RJ1995] Janice Reddish. “Are We Really Entering a Post-Usability Era?”. ACM SIGDOC Asterisk Journal of Computer Documentation, Vol. 19, pages 18-24, 1995.

[SBM2001] K. Schwaber, M. Beedle and R. C. Martin. “Agile Software Development with SCRUM”. Prentice Hall, 2001.

[SC1997] Carl Sagan: “The Demon-Haunted World: Science as a Candle in the Dark”. Ballantine, 1997.

[SHY2007] Esco Strömmer, Mika Hillukkala and Arto Ylisaukko-oja. “Ultra-low Power Sensors with Near Field Communication for Mobile Applications”. Wireless sensor and actor networks, International Federation for Information Processing (IFIP). Springer, 2007.

[SJ1997] J. Stapleton. “DSDM: Dynamic Systems Development Method. The Method in Practice”. Addison-Wesley, 1997.

[SW2003] Will Scott. “Extreme Programming”. Year 2003.
URL: http://www.vsj.co.uk/articles/display.asp?id=124

[TN1986] Hirotaka Takeuchi and Ikujiro Nonaka. “The New New Product Development Game”. Harvard Business Review, 1986.

[WABIR] ABI research.
URL: http://www.abiresearch.com/

[WACR122U] ACR122U NFC Contactless Smart Card Reader. Advanced Card Systems (ACS) Limited.

URL: http://www.acs.com.hk/index.php?pid=product&id=ACR122U

[WAGILEA] The Agile Alliance.
URL: http://www.agilealliance.com

[WAGILMAN] The Agile Manifesto.
URL: http://www.agilemanifesto.org

[WASD] Adaptive Software Development (ASD).
URL: http://www.adaptivesd.com

[WASVN] Project Details for Apache Subversion. Apache Group.
URL: http://projects.apache.org/projects/subversion.html

[WC2W] Content Creation Wiki. Cunningham & Cunningham, Inc.
URL: http://c2.com/cgi/wiki

[WCCJP] Code Conventions for the Java Programming Language. Oracle corporation, 1999.
URL: http://www.oracle.com/technetwork/java/codeconv-138413.html

[WCCMI] Capability Maturity Model Integration (CMMI). Software Engineering Institute. Carnegie Mellon.
URL: http://www.sei.cmu.edu/cmmi/

[WCOCOMO] Constructive Cost Model II (COCOMO II). University of Southern California. Center for Systems and Software Engineering.
URL: http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html

[WCRYSTAL] Crystal Methodologies.
URL: http://www.crystalmethodologies.org

[WDEFXP] Extreme Programming (XP) definitions.
URLs: http://www.extremeprogramming.org, http://www.xprogramming.com, http://www.c2.com/cgi/wiki?ExtremeProgramming

[WDSDM] Dynamic Systems Development Method (DSDM).
URL: http://www.dsdm.org

[WECLIPSE] The Eclipse Foundation open-source community website. Eclipse Foundation Incorporated.
URL: http://www.eclipse.org/

[WEFQM] European Foundation for Quality Management.
URL: http://www.efqm.org/en/

[WEHEA] The Bologna Process - Towards the European Higher Education Area (EHEA), 2010.
URL: http://ec.europa.eu/education/higher-education/doc1290_en.htm

[WETSI] European Telecommunications Standards Institute (ETSI)
URL: http://www.etsi.org/

[WFDD] Feature Driven Development (FDD).
URL: http://www.featuredrivendevelopment.com

[WFNLIB] Forum Nokia Library.
URL: http://library.forum.nokia.com/

[WI2010] Wireless Intelligence. "Global mobile connections surpass 5 billion milestone".
URL: http://www.wirelessintelligence.com/analysis/2010/07/global-mobile-connections-surpass-5-billion-milestone/

[WISO] International Organization for Standardization (ISO) website.
URL: http://www.iso.org

[WJ2ME] Java 2 Micro Edition website. Oracle Corporation.
URL: http://www.oracle.com/technetwork/java/javame/

[WJ2SE] Java 2 Standard Edition. Oracle Corporation.
URL: http://www.oracle.com/technetwork/java/javase/

[WJ2001] Jason D. Wells: "Extreme Programming: A Gentle Introduction". Year 2001.
URL: http://www.extremeprogramming.org

[WJAVA] Oracle Technology Network for Java Developers. Oracle Corporation.
URL: http://www.oracle.com/technetwork/java/

[WKL2005] L. Williams, W. Krebs and L. Layman. "Extreme Programming Evaluation Framework for Object-Oriented Languages (Version 1.4)". North Carolina State University. Department of Computer Science. Raleigh, 2005.

[WLD] Lean Development (LD).
URL: http://www.poppendieck.com

[WLOG4J] Apache log4j 1.2 - Short introduction to log4j.
URL: http://logging.apache.org/log4j/1.2/manual.html

[WMF10] Mobey Forum. "Nokia begins shipping smartphones with NFC technology". URL: http://www.mobeyforum.org/Press-Documents/Industry-News/Nokia-Begins-Shipping-C7-Smartphone-with-NFC-Chip-Inside2

[WMYSQL] Website of the MySQL project. Project lead by Oracle corporation.
URL: http://www.mysql.com/

[WN6212SDK] Series 40 Nokia 6212 NFC SDK. Forum Nokia.
URL: http://www.forum.nokia.com/info/sw.nokia.com/id/5bcaee40-d2b2-4595-b5b5-4833d6a4cda1/S40_Nokia_6212_NFC_SDK.html

[WNAME] Network for Agile Methodologies Experience (NAME).
URL: http://www.name.case.unibz.it

[WNE6212] Nokia 6212 Classic. Nokia Europe.
URL: http://europe.nokia.com/find-products/devices/nokia-6212-classic/

[WNFCF] NFC Forum.
URL: http://www.nfc-forum.org/home/

[WNFCIPJ] Nfcip-java Project.
URL: http://code.google.com/p/nfcip-java/

[WNXPP] Philips: Near Field Communication PN531-µC based Transmission module, Revision 2.0, February 2004.
URL: http://www.nxp.com/documents/data_sheet/100020.pdf

[WOLDAP] OpenLDAP Project, main page.
URL: http://www.openldap.org/

[WOLDAPW] OpenLDAP for Windows.
URL: http://www.userbooster.de/download/openldap-for-windows.aspx

[WREGEXPME] Regexp-me Project.
URL: http://code.google.com/p/regexp-me/

[WSCRUM] SCRUM Project.
URL: http://www.controlchaos.com

[WSONYF] FeliCa technology. Sony global website.
URL: http://www.sony.net/SonyInfo/technology/technology/theme/felica_01.html

[WSPICE] Software Process Improvement and Capability Determination (SPICE).
URL: http://www.sqi.gu.edu.au/spice/

[WTMOV] Tecnomovilidad.
URL: http://www.tecnomovilidad.com/

[WUML] Website of the Unified Modeling Language (UML). Project lead by the Object Management Group (OMG).
URL: http://www.uml.org/

[WVSVNS] VisualSVN Server. Subversion server for Windows.
URL: http://www.visualsvn.com/server/

[YG1997] G. K. Yeap. "Practical Low Power Digital VLSI Design". Springer, 1997.