UNIVERSIDAD DE DEUSTO

DYNAMIC USER INTERFACE ADAPTATION ENGINE THROUGH SEMANTIC MODELLING AND REASONING IN MOBILE DEVICES

Tesis doctoral presentada por Eduardo Castillejo

Dirigida por Dr. Diego López-de-Ipiña
y Dr. Aitor Almeida

Bilbao, Enero de 2015

UNIVERSIDAD DE DEUSTO

DYNAMIC USER INTERFACE ADAPTATION ENGINE THROUGH SEMANTIC MODELLING AND REASONING IN MOBILE DEVICES

Tesis doctoral presentada por Eduardo Castillejo

dentro del Programa de Doctorado en Doctorado en Informática y Telecomunicación

Dirigida por Dr. Diego López-de-Ipiña

y Dr. Aitor Almeida

El doctorando El director El director

Bilbao, Enero de 2015

Author: Eduardo Castillejo
Advisors: Dr. Diego López-de-Ipiña & Dr. Aitor Almeida

The following web-page address contains up to date information about this dissertation and related topics:
http://paginaspersonales.deusto.es/eduardo.castillejo/
Text printed in Bilbao
First edition, February 2015

A mis padres y a mi hermano.

Abstract

Since the birth of the first Graphical User Interfaces (GUIs), Adaptive User Interfaces (AUIs) have been used to cover a wider range of possibilities seeking alternatives for the presentation of the information. Starting with small personalization features, more related to practical interaction, first customizable menus and graphical elements arose. Subsequently, the possibility of breaking down the interaction barriers has grown. From the user perspective, these limitations are usually caused by physiological disabilities, which imped users to properly interact with or consume information.

With the arrival of mobile telephony and portable devices, a wider range of possibilities regarding AUIs have emerged. This market evolves continuously, bringing smaller, more powerful and wearable devices. This trend has strengthened the AUIs within this market. Nevertheless, the user interface adaptation for these devices is far from the advances in static devices, which are able to perform complexer computations. Besides, more problems are added to the equation if the context is considered. The context situation, characterized by the set of singular features which define it and their quality might make the context dynamic. Thus, in each case, different adaptations or configurations might be needed. In fact, users may suffer from specific and/or temporary disabilities due to the context situation. This issue brings new challenges to AUI systems. Hence, from these challenges new adaptive tools started to be included and integrated with the purpose of minimising the identified interaction barriers. This also aimed to allow users to feature a sufficient interaction experience.

In this dissertation these problems are faced, aiming to reduce the boundaries between the user and the device in several limiting situations. To solve them, a dynamic and mobile user adaptation system is presented, principally supported by a semantic model which includes a conceptualization of the user, the current context and the device. Its design allows increasing the expressiveness of the user interaction contextual and interaction needs, also taking into account the features provided by the user's device.

Resumen

La adaptación de interfaces de usuario nos acompaña desde los comienzos de los primeros sistemas operativos basados en interfaces gráficas. El objetivo de estos sistemas de adaptación AUI es buscar alternativas para la presentación de la información. Comenzando con pequeñas opciones de personalización, más orientadas a la búsqueda de la practicidad y eficiencia en la interacción, aparecieron los primeros menús y elementos personalizables. Más adelante, se comenzó a considerar la posibilidad de romper ciertas barreras de interacción. Desde el punto de vista del usuario estas barreras están causadas principalmente por discapacidades, las cuales pueden impedir a un usuario interaccionar de forma apropiada.

Con la llegada de la telefonía móvil y dispositivos portátiles se abre un rango más amplio de posibilidades para los sistemas de adaptación. El mercado de dispositivos evoluciona continuamente, aportando alternativas cada vez más pequeñas y portables. Esta tendencia ha fortalecido a los sistemas de adaptación. Sin embargo, la adaptación de interfaces de usuario en estos dispositivos dista en gran medida de las capacidades de los dispositivos estáticos. Además, la ecuación se complica cuando añadimos una variable como el contexto. La situación del contexto, caracterizada por el conjunto de características singulares que lo definen, describen su calidad y dinamicidad. Por tanto, en cada caso pueden ser necesarias adaptaciones diferentes. De hecho, en ciertas situaciones el usuario puede sufrir discapacidades relacionadas o causadas por el contexto, denominadas temporales. Este problema origina nuevos desafíos para los sistemas de adaptación de interfaces.

En esta tesis se hace frente a estos problemas. Para ello se presenta un sistema de adaptación dinámico y móvil, basado principalmente en un modelo semántico de usuario, contexto y dispositivo que permita la caracterización de dichas entidades en un modelo dinámico que se abstraiga de capacidades o discapacidades concretas y se centré así en las necesidades de interacción en cada momento. Esto permitirá una mayor expresividad de las necesidades contextuales de interacción del usuario, teniendo además en cuenta las características y posibilidades que su dispositivo ofrezca.

Acknowledgements

Escribo esto sin medir la razón, dejándome llevar por lo que siento en este momento. Como se trata de agradecer, voy a empezar por el principio. Esto es, por supuesto, por aita, ama y Lander, que siempre han estado y estarán ahí. A los aitas por darme la oportunidad de estudiar, de ser siempre libre de elegir, y de darme el apoyo moral y económico en tantos momentos, algunos de ellos muy complicados. Y a mi hermano Lander por ser como es, por ese desparpajo y esa cabezonería. Pero sobre todo por darme la posibilidad de reir día tras día. Y la verdad, en general a toda mi familia, siempre preocupados, siempre sacando el tema en los momentos más inoportunos, pero siempre orgullosos. Gracias a mis abuelos por haber tenido una familia tan increíble y habernos educado así. Me considero un tipo con suerte.

Y después de agradecer a mi familia, no puedo olvidarme de mis amigos. Como de alguna forma he de comenzar, elegiré el clásico método cronológico. Primero, gracias a la cuadrilla de Basauri. A Iván, también compañero de cruzadas doctorales, a Iñaki y Molly, cuya boda en Wisconsin no me perdería por nada del mundo, y a Israel y David, por compartir además muchas mañanas de deporte. También gracias a Mikel e Imanol, por tantas escapadas y tantas aventuras. Mención especial para Roberto, siempre dispuesto, siempre al otro lado del teléfono para lo que fuera. Mil gracias tío. Punto y aparte para la cuadrilla macarra, tipos que he conocido en los últimos 3 años y les puedo considerar mis hermanos. Habéis redefinido las bases del significado de amistad. A Ieltxu, por tantos viajes, conciertos, aventuras y desventuras. Por tantas horas en coche juntos, en avión, y en países extranjeros, y por vivir en la zona de los impulsos conmigo. A Dani, empezando desde aquellos años de música prohibida a la tremenda amistad que nos une ahora. A Jon, por empezar compartiendo un pupitre durante tantos años en la universidad y acabar convirtiéndose en un gran amigo. A Endika, eterno, el tío más íntegro que conozco. Otro de estos que siempre está al otro lado de un click de ratón o de la línea de teléfono... y al otro lado de la mesa, compartiendo un buen trago. Compañero también de aventuras y barras de bar. A Igor, Sara, Urko, Aritz y Esti, Luis... Mil gracias por ser como sois. Hasta a Javi

le voy a dar las gracias. Y saliéndome de Euskadi, gracias, Marina, sabes que eres eres mucho más que una amiga para mi. Y gracias Mique por haberla hecho así y por acogerme bajo tu techo. Gracias por ayudarme a evadirme tantas veces, por esos asados argentinos, y por esa cerveza de importación. Y no me olvido de ti, Cata. Sos impresionante. Te escribo esto mientras escucho *"Ya despiértate nena, sube al rayo al fin"*. Malditos argentinos y vuestra hospitalidad y genialidad. Os quiero.

A nivel académico, gracias Diego por haberme dado esta oportunidad, por darme la ocasión de trabajar en este maravilloso grupo de investigación. Gracias por tu sinceridad y cercanía, por tener siempre la puerta abierta, por el tiempo concedido, y por tu valioso feedback en tantas ocasiones. Y gracias, Aitor Almeida, por servirme de guía, de apoyo, por tantos capones y tanta dedicación y paciencia. Gracias por las razones que te llevaron a co-dirigir esta tesis. Desde luego no lo habría conseguido sin este apoyo tan sincero desde el primer día.

Gracias, compañeros del grupo MORElab. Por ayudarme en los momentos malos y por ofrecerme tantos momentos buenos. Gracias a los senior, Aitor, Pablo y Unai, por representar la voz de la experiencia en tantas ocasiones. Y gracias a los demás doctorandos compañeros de batalla, especialmente a Aitor Gómez-Goiri, Iván Pretel y Xabier Eguiluz, porque recorriendo este camino de la mano se ha hecho menos duro. Gracias a los demás compañeros por amenizar cada día en el laboratorio, y gracias a aquellos que os marchásteis y me regalásteis momentos similares.

Thanks to all the inspiring people I met at University of Ulster in Belfast. Specially to Dr. Luke Chen and Professor Christopher Nugent, who helped and guided me with their experience and wisdom. I would like to thank every single researcher and teacher I had the opportunity to talk with in Belfast. I will never forget such a welcoming, aid, feedback and support.

En definitiva, gracias a todos. A aquellos que me habéis apoyado, soportado, aguantado, torturado, y maldecido. Todos sois un poco responsables de este trabajo.

Eskerrik asko,

Eduardo Castillejo

February 2015

Contents

List of Figures

List of Tables

List of listings

List of Definitions

Acronyms

AAL Ambient Assisted Living 49

ADB Android Debug Bridge xxiii

AGPL Affero General Public License 102

AI Artificial Intelligence 11

API Application Programming Interface xii

AUI Adaptive User Interface iii

CAR Circadian Activity Rhythm 50

CC/PP Composite Capabilities/Preferences Profiles xix

CDMA/BREW Code Division Multiple Access/Binary Runtime Environment for Wireless

CoBrA Context Broker Architecture x

CPU Central Processing Unit xx

CSCP Comprehensive Structured Context Profiles 55

dB decibel 74

DDR Device Description Repository xi

DEC Digital Equipment Corporation xvii

DDWG Device Description Working Group 58

DL Description Logic xx

DYNUI Capability and Context-aware Dynamic Adaptation of User Interfaces for Ambient Assisted Living 18

DOI Digital Object Identifier

EU European Union 8

FOAF Friend of a Friend xix

GB Gigabyte xx

GHz Gigahertz xx

GSM Global System for Mobile Communications

GPS Global Positioning System 156

GUI Graphical User Interface iii

GUMO General User Modelling Ontology xv

GUMS General User Modelling System 39

HCI Human-Computer Interaction xv

ICF International Classification of Functioning, Disability and Health xix

IDE Integrated Development Environment

IE Intelligent Environments 47

IEC International Electrotechnical Commission 110

ISBN International Standard Book Number 189

ISO International Organization for Standardization 109

ISSN International Standard Serial Number

JAXB Java Architecture for XML Binding 104

JCR Journal Citacion Reports

KB Knowledge Database

KMS Knowledge Management System 31

LGPL The GNU Lesser General Public License 104

lx luxes 74

MB Megabyte 147

MMS Multimedia Messaging Service 57

MUMMS Measuring the Usability of Multi-Media Software 171

OOP Object-Oriented Programming

OS Operative System 74

OSGi Open Service Gateway initiative 47

OWL The Web Ontology Language 15

PIRAmIDE Personalizable Interactions with Resources on AMI-Enabled Mobile Dynamic Environments 17

PIVon Pervasive Information Visualization Ontology 60

QUIS Questionnaire for User Interaction Satisfaction 172

RAM Random Access Memory xx

RDF Resource Description Framework 55

REST Representational state transfer 142

SGML Standard Generalized Markup Language 54

SI International System of Units 98

SMS Short Message Service 152

SUMI Software Usability Measurement Inventory 171

SOCAM Service-Oriented Context-Aware Middleware x

SOUPA Standard Ontologies for Ubiquitous and Pervasive Applications 60

SPE Sensation-perception-emotion 33

SUS System Usability Scale xiii

SWRL Semantic Web Rule Language xii

TV Television

TTS TextToSpeech 97

UAProf User Agent Profile x

UI User Interface xxi

UML Unified Modelling Language 55

UMT User Modelling Tool 39

URI Uniform Resource Identifier 116

U.S. United States xvi

XML Extensible Markup Language 54

W3C The World Wide Web Consortium 56

WHA World Health Assembly 4

WHO World Health Organization 4

WSN Wireless Sensor Network 49

WURFL Wireless Universal Resource FiLe xi

You take the blue pill, the story ends, you wake up in your bed and believe whatever you want to believe. You take the red pill, you stay in Wonderland, and I show you how deep the rabbit hole goes.

The Matrix

CHAPTER

1

Introduction

Over the years we are witnessing the growth of new and heterogeneous mobile and smart devices and services with a wider range of possibilities. Faster processors, larger memories and more accessible sensor capabilities have allowed the community to develop context-aware applications which, taking into account the user preferences and the context situation, can be customized for the final user. This usually results into the same application having different behaviours, aspects and available features depending on the specific target user. Besides, the spread of intelligent environments provides relevant information about the current context of the agents and involved entities. As a consequence, local governments and public administrations have discovered the importance of working with context data [59]. Thus, they try to improve cities infrastructures and citizens satisfaction.

This situation has brought the appearance of new research domains. This dissertation focuses on one of these domains: adaptive user interfaces. Adaptive user interfaces arise from the need to cover a wider range of users and environment conditions. This area is related to different research domains or trends. For instance, universal or inclusive design. Universal design refers to a set of guidelines for producing different kind of environments (i.e., buildings, software applications and any kind of product) accessible and usable to both people with and without disabilities and dependant people (as the elderly).

During this dissertation we have studied how each user has his own preferences, even those who suffer from similar disabilities. We have also found that usual market applications and devices are usually unable to guarantee a comfortable interaction in many situations for these users. Although several accessibility tools are provided (related to smart devices), these users prefer non smart devices due to the physical interaction that they provide (as physical touch buttons, not touchable screens). This kind of devices are

easy to use and the feedback that they provide is also easy to understand. Hence, this situation has revealed a lack of substantial efforts in the adaptive user interfaces domain. This makes this group of users suffer from interaction inattention.

Hardware advances have brought haptic displays, high definition, curved screens and cameras, accelerometers and different kind of sensors, faster connectivity capabilities, and so forth. On the other hand, software development allow us to use Internet services as we did with a computer. Nevertheless, this progress does not reflect the current society requirements (including inclusive design). While these advances keep reaching the ubiquitous future there are several groups of people that suffer from inattention: the elderly and people with disabilities. These users have special needs. The elderly usually have mobility, sight and hearing problems, while the disabled suffer from more specific and usually severe impairments. There are several accessibility tools that try to reduce the existing interaction boundaries between disabled users and devices [86][55]. For example, the Android operating system allows developers to build more accessible applications by using custom controls and interaction alternatives (e.g., gestures)[1]. Apple's iOS uses zoom, larger text, colours inversions, dictation and voice control to avoid interaction issues[2]. Regrettably, these tools are too static. For example, they do not take the user context into account and they do not understand and learn from the users' experiences. Besides, they are more adaptable than adaptive. Adaptive systems have the ability to dynamically adapt themselves to the current task and user. On the other hand, adaptable systems' functionalities are changed with user intervention [82].

The fact is that user interface adaptation has evolved in the latests 20 years. From the simplest preferences (e.g., modifying the resolution of a screen or monitor) to more sophisticated solutions (e.g., smartphones' brightness automatic controls), the community has attempted to customize applications as far as possible. These adaptations have grown in complexity, covering a wider range of users, as well as taking into account the current context situation. Therefore, context and user modelling have become a real challenge in this domain. This is because of the context variability and the set of different capabilities that users can have.

However, current mobile devices offer new possibilities due to their computational capabilities. Hence, during the following chapters we present a series of contributions which are grouped together forming AdaptUI, a user interface adaptation framework which has a twofold purpose: to reduce the usability problems when users interact with their devices, and to encourage developers to include adaptive engines in their applications to make them more inclusive.

[1] http://developer.android.com/guide/topics/ui/accessibility/index.html

[2] https://www.apple.com/accessibility/ios/

1.1 Background

To clarify the foundations in which this work relies, and to provide a starting point for understanding the rest of the thesis, this chapter introduces and describes two related and significant research motivations in which this dissertation is based. First, introducing our previous experience with AUI systems; and secondly, describing one of the basis that support the work described in this dissertation, which deals with the problem of identifying the user's disabilities and needs.

Consequently, Section 1.1.1 introduces Imhotep, an AUI framework whose benefits and drawbacks have driven the research performed in this thesis. Next, in Section 1.1.2 ICF and the user capabilities related to interaction are addressed.

1.1.1 Previous Experience with AUIs: The Imhotep Framework

Conceived in 2010, Imhotep [49] stands as the result of our first approach to AUI systems. Imhotep is a framework whose main goal is to ease the development of adaptable and more accessible user interfaces. Designed by developers and *for developers*, this framework allows writing applications in a way in which developers do not have to worry about the adaptation of the user interface. The paradigm in which Imhotep is supported deals with the definition of a series of preprocessor directives. With these directives both the user capabilities and the device characteristics are taken into account.

To make the developed applications available to users, they are uploaded to a public repository. Thus, users can download them through an application download tool which sends to the server the user's and device's profile. Hence, the server compiles the best user interface for these profiles.

One of the benefits of Imhotep is the level of expression of the preprocessor directives. Developers can establish their own variables and rules. Listing 1 shows a piece of pseudo-code where the developer defines new variables.

```
1 IF screensize IS big AND resolution IS normal
2 THEN video IS high;
3 IF screensize IS big AND RESOLUTION IS big
4 THEN video IS very_high;
```

Listing 1: Imhotep pseudo-code defining variables [13].

These variables, rules and possible values are defined by the developer using a web based wizard. Furthermore, the concepts, such as *"resolution is big"* are created by the system

taking into account the information of the mobile devices (provided by WURFL[1]) and pondering it with their popularity (with Google Trends[2] data).
The results obtained in Imhotep have motivated this dissertation. Moreover, Imhotep serves as a good metric to evaluate the benefits of AdaptUI (the AUI platform described in this thesis). A more specific review of Imhotep is given in Chapter 5, in which the Imhotep's architecture is reviewed and its performance compared with AdaptUI.

1.1.2 User's Capabilities and Interaction

Researchers have developed and improved several techniques to model users for the past 20 years [126][81]. Modelling users implies gathering knowledge of their capabilities, drawbacks and limitations. During these first decades there was not any official and medical-based study to support user capabilities. Nevertheless, in 2001 this situation changed. The World Health Assembly (WHA), a forum through which the World Health Organization (WHO) is governed, published the ICF[3]. ICF is a classification of human functioning and disability. It classifies every function state associated with health (e.g., diseases, disruptions, injuries and traumas). Its purpose is to identify the low-level capabilities relevant to product design in several domains. As was written by experts in the area, ICF is a reference for identifying several user capabilities in any interaction process. Its main goals are the following:

- To provide a scientific basis to study and understand health and health-related states, outcomes and determinants.
- To establish a common language for describing health-related states in order to improve communication between different users, such as health care workers, researchers, policy-makers and the public, including people with disabilities.
- To permit comparison of data across countries, health care disciplines, services and time.
- To provide a systematic coding scheme for health information systems.

ICF is organized into two main groups. On the one hand, Part 1 deals with *Functioning and Disability*, indicating problems (e.g. impairment, activity limitation or participation restriction summarized under the umbrella term disability). On the other hand, Part 2 covers *Contextual Factors*. This group gathers a list of *Environmental Factors* which

[1] http://wurfl.sourceforge.net/
[2] https://www.google.com/trends/
[3] http://www.who.int/classifications/icf/en/

have an impact on all components of functioning and disability. The most significant function groups of Part 1 are highlighted below:

- *Body functions*, which are the physiological functions of body systems (including psychological functions). These functions encompass:
  - Mental functions.
  - *Sensory functions and pain*.
  - Voice and speech functions.
  - *Neuromusculoskeletal and movement-related functions*.
- Body structures, as the anatomical parts of the body such as organs, limbs and their components.
- Activities and participation. An activity is defined as the execution of a task or action by an individual. Participation is involvement in a life situation.
- Environmental factors, which make up the physical, social and attitudinal environment in which people live and conduct their lives.

In Part 2, *Environmental factors* include:

- Products and technology.
- *Natural environment and human-made changes to environment.* It encompasses:
  - Physical geography.
  - Population.
  - Flora and fauna.
  - *Climate*.
  - Natural events.
  - Human-caused events.
  - *Light*.
  - *Time-related changes*.
  - *Sound*.
  - *Vibration*.
  - Air quality.
- Support and relationships.
- Attitudes.
- Services, systems and policies.

Two significant terms that this dissertation uses are *impairment* and *environmental factors*, defined by ICF as follows:

Category	Component group	Function	Description
Body functions (Part I)	Seeing and related functions	Visual acuity	Seeing functions of sensing from and contour, both binocular and monocular, for both distant and near vision.
	Hearing and vestibular functions	Hearing	Sensory functions relating to sensing the presence of sounds and discriminating the location, pitch, loudness and quality of sounds.
	Neuromusculos-keletal and movement-related functions	Mobility of a joint	Functions of the range and ease of movement of a joint.
Natural environment and human-made changes to environment (Part II)	Climate	Temperature Precipitation Wind	Meteorological features and events, such as the weather.
	Natural events		Geographic and atmospheric changes that cause disruption in an individual's physical environment.
	Light	Intensity	Level or amount of energy being emitted by either a natural or an artificial source of light.
	Time-related changes		Natural, regular or predictable temporal change.
	Sound	Intensity	Level or volume of auditory phenomenon determined by the amount of energy being generated.

Table 1.1: ICF components considered in this dissertation.

Impairments, by ICF

> *"Impairments are problems in body function or structure such as a significant deviation or loss".*

Environmental factors, by ICF

> *"Environmental factors make up the physical, social and attitudinal environment in which people live and conduct their lives".*

ICF and its considerations, its mentioned and highlighted functions classification support the motivation for this thesis.

1.2 Motivation

HCI studies the design of the interaction between computers and users. Carlisle [61] was the first author who wondered about the interaction between humans and machines and several possible improvements. However, the term HCI was not used until 1980 by Card et al. [60].

Figure 1.1: First computer mouse by Douglas Engelbart, formed by two wheels representing the two axis on the display, and a single button.

But the concept of HCI is not relegated to the past evolution of computers and industry. It has kept evolving and improving from the invention of the first computer mouse (see Figure 1.1) by Douglas Engelbart during the sixties to nowadays interaction advances (see Figure 1.2). This is thanks to emerging mobile and ubiquitous devices' capabilities, which have opened new research domains and their power have outstripped the last decades machines' computational capabilities. Besides, the *mobile device* concept is almost obsolete, since it has been replaced with what we know as *smartphones*. Smartphones are feature/-mobile phones built over a mobile operative system and more connectivity and computing capabilities.

Figure 1.2: Google Glasses navigation interface [44].

Nevertheless, these devices are not just characterized by their power, process, sensors and connectivity. They also allow developers and designers to include customization and personalization features. Therefore, they are becoming more than simple mobile phones.

Now they are personal and intimate. They bring us Internet, email, social networks (e.g., Facebook and Twitter) and so forth. They use our habits and personal skills to recommend different resources, activities and even people. And all of this just within reach.

But then, which is the motivation for this dissertation? The answer is found regarding the target users of the cited adaptation approaches and advances. To us developing for dependant users and users with disabilities is highly needed. Not significant efforts are currently being developed in this area, and how important might result having new tools for developers to make their applications inclusive. Besides, something that most systems regarding these users lack, is the consideration that users with similar disabilities might behave in different ways. This is due not only to their capabilities, but also due to the way they suffer them, or the influence of other capabilities. For example, blindness can be caused by several diseases, injuries, genetic defects or poisoning. In addition, people with the same disability might have different orientation perception, which might lead into different ways of suffering blindness.

Another reason which motivates this dissertation is the fact that nowadays the share of people aged over 65 represent a 17% of the current European population. By the year 2060 this figure is projected to rise to 30% [77]. As a consequence, and as the European Commission states, *"the European Union (EU) would move from having four people of working-age to each person aged over 65 years to about two people of working-age"*. The current situation shows that it is still a small group, but with a high expected increasing ratio. Nevertheless, this implies that we are still in time of accommodating, adapting and overtaking for future economic and demographic consequences.

Besides, systems personalization and environment components adaptation have been demonstrated to benefit both users and service providers [106]. However, to achieve a satisfactory adaptation, it is necessary to have several inputs, for example, a user characteristics model. Hence, the service provider will be able to apply the corresponding adaptations for the corresponding user. In addition, current context conditions [102] and user's device capabilities are also crucial within this domain.

On the other hand, being aware of what happens in the user's surroundings is what researchers called context-awareness. Context-aware applications and systems have been supported by the community for their significance within user's focused computing [133] [66]. It is known that context affects user capabilities. In this dissertation we will see how these capabilities are somehow influenced by context and how any adaptive system should react to assure a minimum level of interaction with the user. As a consequence, we will study each user, context and device capability and we will present a model which allows to represent the interaction that these entities may have.

In spite of the research made by Kobsa [106] and Fink and Kobsa [80] to get generic user models (more focused on Artificial Intelligence) there is a lack of a common, exportable and standard models for these environments (adaptive user interface environments) (more details provided in Chapter 2). Most of the solutions presented in Chapter 2 are strongly domain dependent. They rarely represent, for example, the influence of context variables in the current domain to perform adaptations.
Therefore, there is a necessity for a solution that models every entity that participates in an adaptive user interface environment and a methodology which permits dynamic adaptations of these entities based on their mutually affecting capabilities. This methodology will have to take into account several user reactions with the adapted user interfaces. Consequently, an interaction model for these environments will be required to evaluate user satisfaction with the presented user interfaces

1.2.1 User's Context Temporary Disabilities

Product design, by Nelson [116]

> *"Designing an object to be simple and clear takes at least twice as long as the usual way. It requires concentration at the outset on how a clear and simple system would work, followed by the steps required to make it come out that way—steps which are often much harder and more complex than the ordinary ones. It also requires relentless pursuit of that simplicity even when obstacles appear which would seem to stand in the way of that simplicity."*

Nelson introduced the problems that designing a product entails. One of the most significant issues to face during this process is the usability. According to the ISO/IEC 9126 standard [42], quality represents a property of the software product defined in terms of a set of interdependent attributes (i.e., usability, security, reliability, performance, complexity, readability, reusability) expressed at different levels of detail and also taken into account the particular context of software use. At this point, the ISO 9241-11 standard states that usability is the extent to which a product can be used by the specified set of users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [100].
The interaction with devices needs to be satisfactory for the users. The ISO/IEC 9126-1 [42] presents and details a two-part model for software product quality:

1. *Internal and external quality* (see Figure 1.3): Internal Quality is the totality of attributes of the software product from an internal view (e.g., spent resources,

analysability). It is measured and improved during the code implementation, reviewing and testing. External Quality is the quality when software is running in terms of its behaviour (e.g., number of wrong expected reactions). It is measured and evaluated for software testing in a simulated environment.

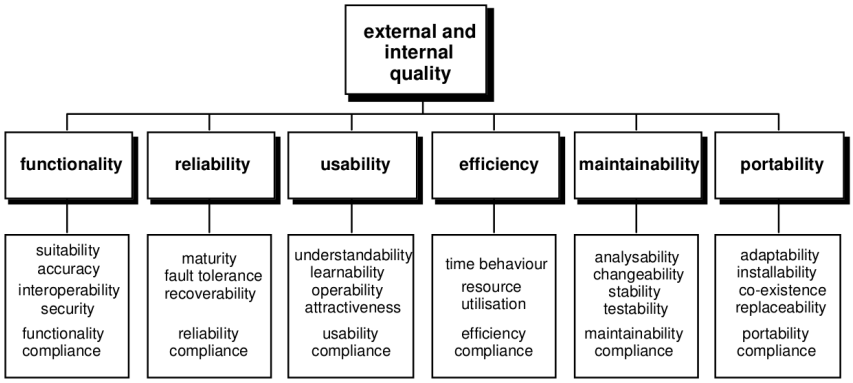

Figure 1.3: Quality model for external and internal quality [42].

2. *Quality in use* (see Figure 1.4): It is the capability of the software product to enable specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use.

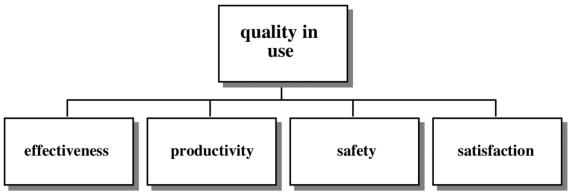

Figure 1.4: Quality model for quality in use [42].

However, the design process becomes troublesome because of the nature of each user. Users are very different from each others. They like different things and they sense and perceive differently. Moreover, they have different capabilities. Besides, there are several groups which suffer these differences more deeply: people with disabilities and the elderly. People with disabilities suffer from different impairments which are responsible for limiting several capabilities in a certain way. For example, users with sight disabilities will suffer from interaction problems with their devices if this interaction is based on visual stimulus (e.g., using a device display). On the other hand, elderly people usually suffer similar interaction troubles due to their ageing. As their senses tend to tire their capabilities and interaction levels decrease. Current technology trends try to reduce the interaction barriers that elderly suffer with nowadays devices. Mobile phones have

audio control interaction and screen augmentation, TVs have zoom and subtitles capabilities, and so on. Nevertheless, the elderly are used to use the products they already know [132][138].

Nonetheless, people without disabilities are not exempt of suffering from very similar situations. There are many conditions in which people without disabilities feel like if they had one. Using our smartphone when it is raining or with direct sunlight might affect our interaction with the device. These situations limit users' capabilities. They are examples of what context is and what it is capable of during an interaction process [73]. Desktops are less prone to suffer from context conditions (obviously certain situations are impossible to avoid, like infrastructure problems). On the other hand, mobile devices are predisposed to experience problems due to current context situation.

Context is essentially characterized by its capability to change. Besides, as is detailed in this dissertation context characteristics can affect several user's capabilities. Furthermore, it can change users' capabilities usually reducing them as they have temporary disabilities. For example, direct sunlight on a mobile phone screen reduces our sight capability; traffic or crowded streets reduces our attention and hearing capabilities; several activities (e.g., driving and cooking) and weather conditions (e.g., raining) affect our attention and mobility. These are examples of *user's context disabilities*. The first approximation to this idea was conceived by reviewing the literature. More specifically, and as is shown in Chapter 2, the thesis by Heckmann [92] presents the following illustration.

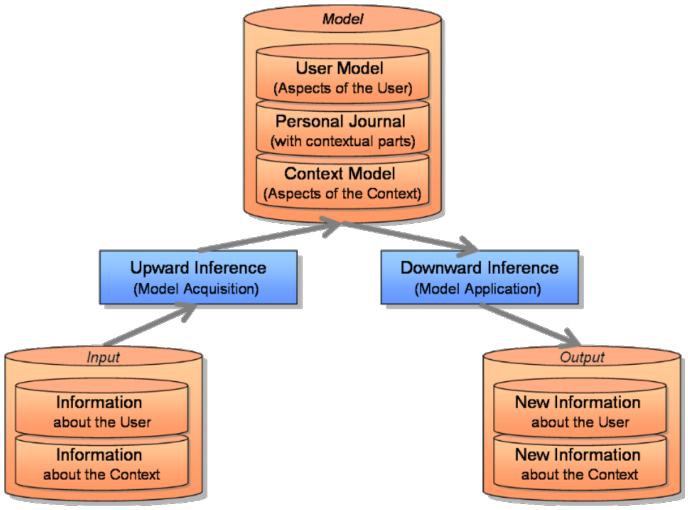

Figure 1.5: Extended processing schema within context-aware user-adaptive systems, derived from [102] and [105], as appears in [92].

Figure 1.5 represents a conceptual view of the theory of user modelling with integrated context-awareness. The figure presents model acquisition and model application information flows. From an Artificial Intelligence (AI) perspective, Heckmann considers that

the input data concerning the user and input data concerning the context are processed upward, as an inference step. This is what the author calls model acquisition. On the contrary, the model application works downward, calculating a new hypothesis about the user or the context. This idea made us understand the user, the context and the device as evolutionary entities which, in each case, need from specific inference to result into satisfactory usability with the user interface.
Hence, the main purpose of this dissertation is to dynamically reduce the disabilities caused by context on mobile devices by adapting their user interface. This will help to maintain certain levels of interaction with the users which would be impossible to reach in natural conditions.

1.2.2 Definitions

During this dissertation several concepts are often named and referenced. Thus, in this section these concepts, related to user interface adaptation, the main discipline tackled by this dissertation, are given.

Universal design, by Story et al. [139]

> *"... the design of products and environments to be usable to the greatest extent possible by people of all ages and abilities."*

User-adaptive system, by Jameson [103]

> *"An interactive system that adapts its behaviour to individual users on the basis of processes of user model acquisition and application that involve some form of learning, inference, or decision making."*

Adaptive user interface, by Jameson [103]

> *"A software artefact that improves its ability to interact with a user by constructing a user model based on partial experience with that user."*

User

As the main entity of the AdaptUI ecosystem, a user is understood as an individual who has a set of characteristics which define the interaction. These characteristics might represent *capabilities* or *disabilities*. AdaptUI represents a user entity through a semantic model which avoids the explicit representation of such concepts.

Context (I), by Dey [73]

"Contex is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." Context represents the second main entity in the AdaptUI environment. In this dissertation this definition of context is considered, as we believe that is the most popular definition regarding the literature. As the user and the device, context is semantically represented as a set of characteristics which defines the current situation.

Device

As the third entity included in the studied ecosystem, the device represents the device that the user manipulates within the current context. Devices are also understood as a series of characteristics which identify them, including both software and hardware related.

Context-aware, by Schilit and Theimer [134]

"Context-aware computing is the ability of mobile user's applications to discover and react to changes in the environment they are situated in." In this dissertation several references to context-aware systems are made, specially in Chapter 2.

Distinguishing between adaptable and adaptive user interfaces is significant to better understanding the goals of this dissertation[1]. Hence, two definitions remarking the differences between these two concepts are given by Fischer:

[1] Although in dissertation several adaptable approaches and solutions are analysed (more concretely in Chapter 2), the final goal of the presented framework is to be adaptive.

Adaptable user interface, by [82]

In this dissertation we assume that adaptable user interfaces, as Fischer [82] defines them, are those that change due to the user intervention. In other words, not any user interface component adapts its shape or behaviour without the explicit user specification.

Adaptive user interface, by [82]

On the contrary, related to the previous definition, adaptive user interfaces have the ability to dynamically adapt themselves to the current task and user, not needing the user intervention [82]. In the following chapters, the reader will easily understand the differences between adaptable and adaptive through several examples and the AdaptUI's architecture detail.

User disability

This dissertation tries to reduce the possible (or temporary) disabilities that users might suffer when using their devices. To this end, a formal definition of what a disability is, it is needed. As the ICF document defines, disability *"serves as an umbrella term for impairments, activity limitations or participation restrictions."*

Context disabilities

AdaptUI focuses on the interaction of users suffering temporary disabilities caused by the current context conditions. Thus, in this dissertation we introduce the concept of context disabilities. To us, context disabilities are basically temporary disabilities caused by the current context situation, which might limit several user normal abilities or capabilities.

Physiological capabilities

During the following chapters we refer to physiological capabilities as those capabilities that are included in the ICF document under the sensory functions classification.

Ontology, by Gruber [87]

Ontologies are a *"explicit specification of a conceptualization."* In other words, it is a formal mechanism to formally represents concepts of a concrete domain.

Reasoning engine

A reasoning engine (or semantic engine) is a piece of software which is able to infer logical consequences from a set of axioms or assertions. As defined in [27], *"a reasoner is a key component for working with The Web Ontology Language (OWL) ontologies. In fact, virtually all querying of an OWL ontology (and its imports closure) should be done using a reasoner. This is because knowledge in an ontology might not be explicit and a reasoner is required to deduce implicit knowledge so that the correct query results are obtained."*

Inclusive design

The design of mainstream products and/or services that are accessible to, and usable by, people with the widest range of abilities within the widest range of situations without the need for special adaptation or design [43].

1.3 Hypothesis, Goals and Limitations

Based on the current state of Adaptive User Interfaces, the following hypothesis is developed:

User interaction limitations with user interfaces in mobile devices due to users' context temporary disabilities are reduced by dynamically adapting the corresponding applications' user interfaces through a semantic reasoning process. This process includes their capabilities as users, the set of characteristics which defines the current environment where they actually are, and the characteristics of the devices they use.

This hypothesis is validated undertaking the following main goal:

> To design and implement an adaptive user interface system which runs fully in the user's device, includes current context situation, and considers several temporary or enduring user disabilities, supported by different sets of rules which make the adaptation transparent for the user.

This objective is achieved through the attainment of the following more specific objectives:

1. To study the current state of the art on user interface adaptation systems; user, context and device modelling; and mobile reasoning engines.
2. To design an ontology which models user capabilities through an abstract perspective, the context situation and a set of device's static and non static characteristics. The ontology must consider possible interactions between each entity.
3. To design a set of rules which allow the interaction between the cited entities and the final adaptation of the user interface.
4. To design and implement a reasoning mobile engine which allows reasoning in Android based devices.
5. To provide an API for developers to make available the design of adaptive user interfaces and the edition of existing rules sets, as the knowledge represented through the ontology.
6. To validate the obtained results both qualitatively and quantitatively.

The resulting methodology should also satisfy the following requirements:

- The designed model should be descriptive, complete and robust enough to be able to represent any possible context-aware situation with the users and their devices.
- The model should represent several user capabilities through an abstract perspective, considering that designers, developers and users might lack physiological background of capabilities.
- The designed model should be fully domain independent. Consequently, it should be exportable and reusable to external user interface adaptation environments.

- The model should represent physiological capabilities transparently for the user and the developer of the adaptive user interface.

- The designed and implemented reasoning engine should be compatible with Android based devices, allowing the use of rules and reasoning features.

- The designed API should allow developers to design automatically adaptive user interfaces and also the modification of the knowledge represented by the ontology.

The following features will be considered beyond the scope of this research:

- Only capabilities related to visual and hearing are taken into account for the dynamic adaptation and the user context disability approach.

- No cognitive problems have been faced.

- In the adaptation process the device battery level is not taken under consideration.

1.4 Thesis Context

This thesis has been developed in the context of the research centre Deusto Institute of Technology, DeustoTech, University of Deusto[1]. The work that has made possible the development of this thesis is in the context of the following research projects:

- PIRAmIDE: Funded by the Spanish Ministry of Industry, Tourism and Trade, the Personalizable Interactions with Resources on AMI-Enabled Mobile Dynamic Environments (PIRAmIDE) project proposes to use user mobile device as a catalizer of the interaction of users with their environment, acting as a sixth sense which aids and assists us facilitating and improving our daily interactions with the objects that surround us in our workplace, home or public administrations.

- UCADAMI: The User and Context-aware Dynamically Adaptable Mobile Interfaces project, funded by the Industry, Innovation, Commerce and Tourism Department of the Basque Government aimed to create a technological framework which allows dynamic user interface adaptation based on the usage and context of use of the digital interactive content consumed through mobile devices.

[1] `http://www.deustotech.deusto.es/`

- DYNUI: The aim of Capability and Context-aware Dynamic Adaptation of User Interfaces for Ambient Assisted Living (DYNUI) is to define an intelligent platform that facilitates the development and deployment of user-environment interfaces adaptable to the users, their interaction devices and their context. These interfaces have to be adapted both at the beginning and during the execution of services taking into account the users' capacities, their interaction devices and the users' and their environment's current context.

1.5 Methodology

Achieving the detailed goals requires the following research strategy:

a) To perform an exhaustive review of the state of the art in the areas of user, context and device modelling, semantic reasoning and ontologies. This analysis has been reinforced by attending specialized scientific congresses.

b) Perform a critical evaluation of the existing solutions, analysing their limitations and scope and identifying the corresponding areas where it was possible to make a contribution to the state of the art.

c) To design and develop the different modules of an adaptive system infrastructure aware of the context capabilities of users, by gradually extending their scope and capabilities.

d) To carry out several experiments and evaluate the performance of the developed modules.

e) Attending congresses to present the achieved contributions with the purpose of receiving the corresponding feedback from the scientific community.

f) Network with experts at conferences and meetings.

g) Update the contributions and redesign the system with the feedback attained from the previous actions.

h) To develop and deploy the final dynamic user interface adaptation infrastructure, which takes into account the user, context and device current characteristics to provide the best suitable user interface.

i) Disseminate the results obtained to the scientific community.

Figure 1.6 summarizes and illustrated the cited and followed methodology.

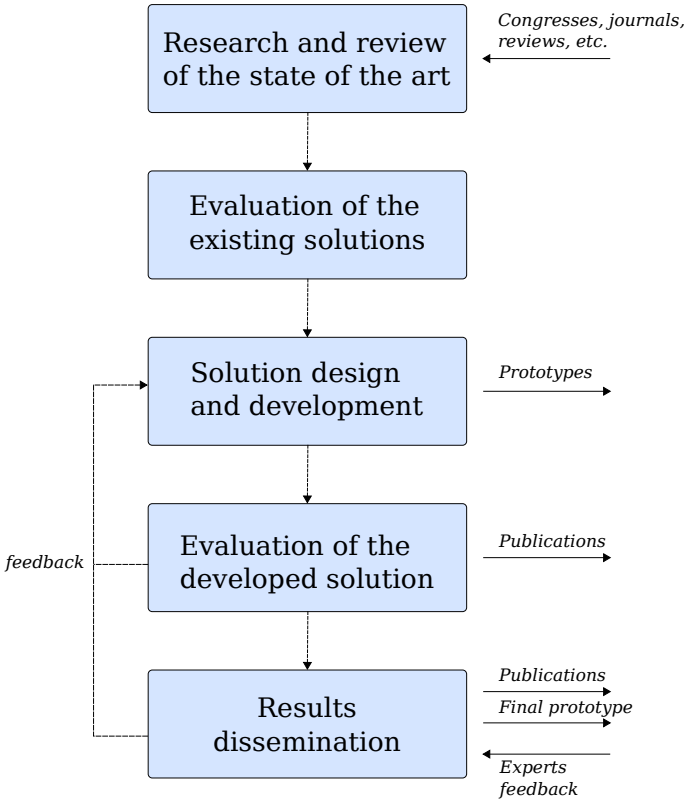

Figure 1.6: Followed research methodology.

1.6 Outline of this Thesis

This thesis is presented in 6 chapters (including the current one). As a guide to the organization of the remainder of this thesis:

Chapter 1 outlines the motivation, hypothesis, goals and limitations of this research. It also describes the followed research methodology.

Chapter 2 analyses the state of the art regarding user, context and device modelling, as well as the different adaptation solutions in the last twenty years.

Chapter 3 presents AdaptUIOnt: an ontology for dynamic user interface adaptation specially designed for this dissertation, but also aiming to be used by the scientific community in the user interface adaptation domain.

Chapter 4 presents the three layered architecture of the AdaptUI system, decoupling its modules, their purposes, and main characteristics.

Chapter 5 outlines and analyses the corresponding experiments, specially performed to evaluate the whole AdaptUI platform. This evaluation is divided in two different parts. The first one, regarding only technical issues of the developed system. The second one, pursuing the satisfaction of users.

Chapter 6 draws the conclusions of this research work. We discuss the achieved goals and analyse the advantages and drawbacks of the developed system. Some future lines of research are proposed and the chapter ends with some final remarks.

The lover of life's not a sinner, the ending is just a beginner. The closer you get to the meaning, the sooner you'll know that you're dreaming.

Black Sabbath

CHAPTER

2

State of the Art

This chapter introduces several significant models and techniques related to adaptive user interfaces and systems during the past 20 years. Besides, user-oriented, context-aware and device models and solutions are reviewed. Once the general domain in which this dissertation lies has been analysed during this chapter, the specific approaches that are close to this proposal are examined. Thus, the way previous proposals have addressed this dissertation's aspects will be analysed, highlighting their goodness and weaknesses. First, in Section 2.1 several important adaptation models are introduced. The performed methodology and modelled parameters are analysed in detail. Next, Section 2.2 focuses its attention in the models that have been used during the past 20 years for characterizing the user and his/her capabilities. Section 2.3 analyses the most significant context-aware systems and the way these approaches model the context and its characteristics. Devices are also one of the main entities that are taken into account in this dissertation for the adaptation process. Therefore, Section 2.4 examines several modelling techniques and the devices' capabilities in different domains. Besides, a chronological review of the evolution of the presented user and context models will be depicted in the corresponding section.

2.1 Adaptation Models

Before introducing the analysed adaptation models, a terminology review is given taking into account the differences between adaptable and adaptive defined by Fischer (see these definitions in Section 1.2.2).

	Adaptive	**Adaptable**
Definition	Dynamic adaptation by the system itself to current tasks and current user.	User changes (with substantial system support) the functionality of the system
Knowledge	Contained in the system; projected in different ways.	Knowledge is extended.
Strengths	Little (or no) effort by the user; no knowledge of the user is special required.	User is in control; user knows her/his task best; system knowledge will fit better; success model exists.
Weaknesses	User has difficulty developing a coherent model of the system; loss of control; few (if any) success models exist (except humans).	Systems become incompatible; user must do substantial work; complexity is increased (user needs to learn the adaptation component).
Mechanisms required	Models of users, tasks, and dialogs; knowledge base of goals and plans; powerful matching capabilities; incremental update of models.	Layered architecture; domain models and domain-orientation; back-talk from the system; design rationale.
Application domains	Active help systems, critiquing systems, differential descriptions, user interface interface customization, information retrieval.	Information retrieval, end-user modifiability, tailorability, filtering, design in use.

Table 2.1: Fischer [82] comparison between adaptable and adaptive systems.

Fischer [82] defined *adaptive* and *adaptable* principles in 2001 through 6 different concepts: definition, knowledge, strengths, weaknesses, required mechanisms, and application domains. Table 2.1 shows a representation of these concepts.

As is shown in Table 2.1, an adaptive system is able to dynamically change due if a certain situation is met. This situation might act as a trigger for the adaptation. Adaptable systems, however, require the user intervention. Current smartphones provide a set of tools to allow users to adapt several elements of the user interface. For example, font sizes, colour combinations, screen magnification and dictation are several adaptable functionalities available in such devices [3][14], known as accessibility tools. Figure 2.1 shows an example of several accessibility tools in iOS, including VoiceOver on the left and Siri on the right.

These accessibility functionalities make the user interfaces *adaptable* by the users. On the contrary, *adaptive* user interfaces would modify the aspect of the shown elements without the user intervention. This means that adaptable user interfaces needs the user to change the corresponding adaptable characteristic (e.g., the font size) while an adaptive user interface would adapt without it.

Subsequently, *adaptivity* and *adaptability* are introduced. *Adaptivity* is related to the fact that a system or a service is able to learn somehow to change itself to increase the user satisfaction in the interaction process. On the other hand, *adaptability* deals with the

Figure 2.1: iOS VoiceOver [16] and Siri [15].

property of the system or service to be customized by the user [102].
Now that we have defined what adaptivity means, several significant adaptive systems are presented in the following lines.

2.1.1 Significant Adaptation Models

User interface adaptation has evolved through HCI history. First, and dealing with the concept of adaptability (and not adaptivity), user interfaces start to be malleable. Colour palettes and screen resolution tools were given to the user. Nowadays, regarding our portable devices, even automatic brightness control systems are available. Since many years, developers have attempted to offer customization tools to the end user. These tools have grown in complexity, covering a wider range of functionalities and also users. What is more, a few years ago the context was taken into account as the set of characteristics that define a situation (the context definition which is adopted in this dissertation is given by Dey [73], and its definition is introduced in Section 1.2.2).
Systems personalization and environment components adaptation have been demonstrated to benefit both users and service providers [106]. However, for achieving a satisfactory adaptation it is necessary to have several inputs, for example, a user characteristics model. Hence, the service provider will be able to apply the corresponding adaptations for the corresponding user. Besides, we believe that current context conditions [102] and user's device capabilities are also crucial within this domain.
Nilsson et al. [117] considered that designing user interfaces for mobile devices tends to be problematic for several reasons (e.g., screen size is small and interaction mechanisms

are very different from a desktop system). Besides, these devices are usually used in dynamic environments (i.e., variations in the context).

Calvary et al. [58] stated in 2002 that a user interface is *plastic* if it is capable to adapt itself taking into account context changes keeping the usability. They presented a process and a dynamic software mechanism which supports context variability. This work was supported by the idea that context changes may provoke the triggering of several reactions under a *prologue-action-epilogue* paradigm.

Using another paradigm, Lehtonen et al. [109] detailed a tool to perform dynamic adaptation on documents based on several user parameters (language, document type, and so on). The presented approach is based on several *configuration files* (Product Configuration Files) which describe the current user interface and store the user preferences. The user interfaces are designed with Bean Markup Language [144] (a language focused on user interfaces) and Java Beans.

Following a *middleware* based approach, four remarkable solutions are highlighted in the following lines. Repo [129] introduced in 2004 a model that allows the use of Web-based user interfaces (as well as the creation of new ones) that covers the environment adaptability requirements and context in the best possible way. The main problem is given by the range of devices that users typically employ, which have different capabilities and run different platforms. This situation causes the need of more flexible applications and devices. Repo identified a lack of attention in the initial adaptation process (e.g., when the capabilities of the mobile device are identified). This approach was based on a *middleware* architecture, and it was capable of detecting new devices in the current environment (context changes). It allowed services to query for devices' capabilities through a middleware architecture. Once a service identified a certain device, it sent the corresponding user interface.

Nilsson et al. [117] introduced a *middleware* solution which was able to build auto-adaptable systems. In this case, the middleware leads the adaptation process dynamically, providing several mechanisms to: detect changes in the application context, reason about these changes, and adapt to them by a dynamic reconfiguration of the current application.

Based on the same architecture introduced by Nilsson et al. [117], Hallsteinsen et al. [90] presented a system which was able to react to context changes recommending alternative configurations in each case. The benefit for the user comes from the reasonable adaptation decisions took by the middleware, being the developer responsible for describing configuration options for important variation points.

Stuerzlinger et al. [141] focused their work on desktop applications adaptations and on the adaptation, reconfiguration and combination of user interfaces using a *User Interface Facades* system. Based on the definitions laid down by Marmolin et al. [114], where differ-

ences between superficial personalization (which allows users to select different options between some predefined) and deep customization (which allows customizing deeper aspects of the system) were introduced, authors stated the following criteria for adaptive interfaces:

- Fast, simple, or just-in-time customization: users are able to customize their interfaces without advanced planning, whenever they need it, and they will be able to do it in a simple way.
- Not only big personalization, also local ones (at minor scale).
- Deep personalization: users can define new customization rules.
- Cross-application personalization: interfaces customization must enable different applications to be combined.

A *framework* based approach is presented by Almeida et al. [49] in 2011. Imhotep is a framework for user interface adaptation based on inserting preprocessor primitives within the source code. Thus, at compilation time different versions of the final application are generated due to the corresponding user and device parameters. A WURFL [40] database was used for modelling devices and their capabilities. WURFL is a DDR, a catalogue of mobile device information and a framework for adaptation of mobile user interfaces. By using it, authors ensure to have the latest devices with their capabilities. As configuration files, the platform checks both user and device capabilities and uses them to compile the corresponding solution. This approach has the limitation of being static. This means that each change in the user capabilities needs a new compilation of the whole application. As this framework has supposed our first incursion in the field of AUIs, further details of its main characteristics and architecture are given in Section 1.1.1 and Section 5.1.2. Besides, several experiments are carried out comparing Imhotep and the proposed solution, which will be depicted in the following chapters.
Evers et al. [78] tackle the problem of the need of user interaction in the adaptation process. Centred in users, authors discuss about adaptation and usability defining different types of adaptation (i.e., forward and backward) and different user ways of interaction (i.e., implicit and explicit). This perspective helps developers to take into account not only technical characteristics (e.g., users models, devices capabilities, context parameters, adaptation engines, and so on) but also some psychology to be aware of user mood or stress. In stressful situations users may not be comfortable with an adaptation engine which asks questions about the process.
Several ideas of the reviewed works are taken into account for the proposed user model in combination with the Casas et al. [63] research. For instance, the visual handicap metrics

in the sensory layer gives an idea of modelling not the user disability but the minimum needed configuration for a view component in the screen. This issue is deeply discussed in Section 2.2.

2.1.2 Physical Adaptive systems

Although it is out of the scope of this dissertation, there is a research pathway dealing with physical adaptive systems. In Tactus [36], a company which aims to redefine devices and user experiences by combining the modern and traditional interfaces and on-demand buttons on touch screens for a tactile experience[1], a novel adaptive technology has been developed. This technology allow screens to dynamically build physical buttons into a flat touch screen, depending on the current task. For example, if the user needs to send an email, the keyboard emerges from the screen as a physical interface. This interface is supported by the combination of small fluid channels which are routed throughout the Tactile Layer. This layer enables fluid to expand the top polymer layer to create the physical buttons[2] (see Figure 2.2).

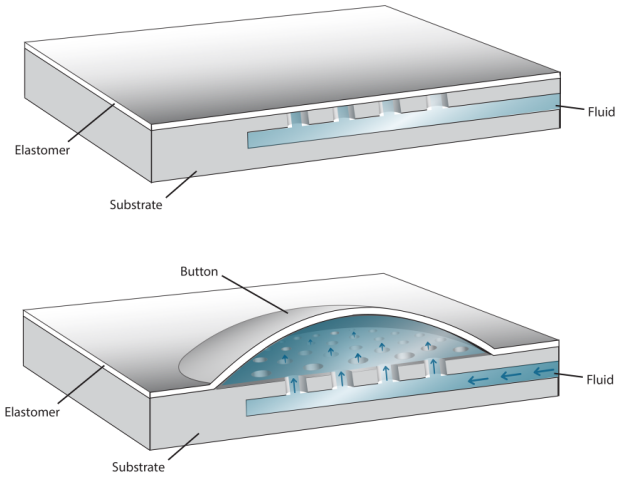

Figure 2.2: Buttons during the flat state (up) and during the raised state (down) in Tactus.

Nowadays, this technology might be surprising for the reader. Nevertheless, there are examples of physical user interfaces adaptation all around us. One of the most common and spread example is shown in cars. The cars companies deal with adaptivity for the current driver not just inside the car, but also outside. Some models adapt the driving wheel in depth and height depending on the driver who unlocks the door. Displays and

[1] https://www.linkedin.com/company/tactus-technology
[2] https://www.youtube.com/watch?v=t4eh-Cn3Pzk

controller brightness, and radio volume can be also adapted considering context light or noise. Even the car lights system adapts to the road conditions.
These are just a few examples of what adaptation technologies might be pointing at for the near future.

2.2 User Modelling

2.2.1 A Chronological Review of the Evolution of User Models

Figure 2.3 illustrates the chronological evolution and different solutions for the last 15 years. During these years different user characteristics have been taken into account considering the final purpose of the designed system.

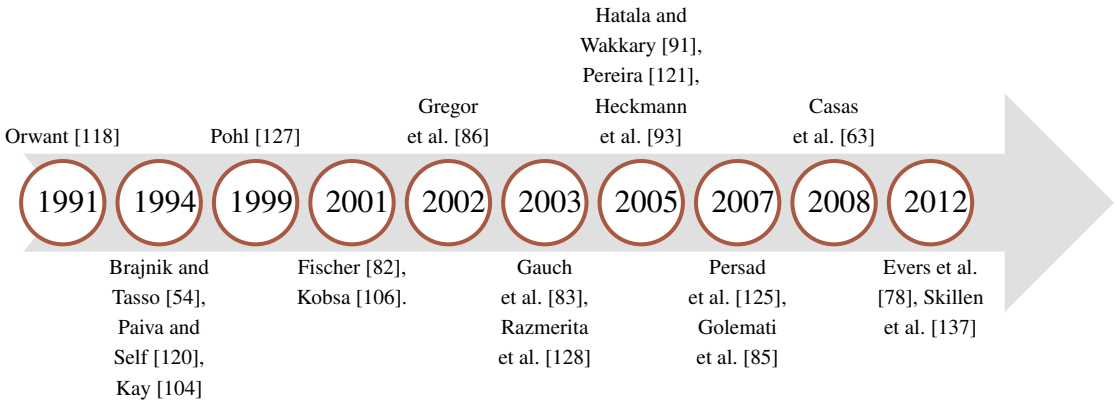

Figure 2.3: The chronological view of the evolution of remarkable user models considered in this dissertation.

First user modelling systems started in the late eighties. Allen [47], Cohen and Perrault [70], Perrault et al. [122], and Rich [130][131] are examples of researchers whose works inspired next user modelling approaches. From there, several authors started collecting different types of information about the users and exhibiting, for example, different kinds of adaptations to them [106].
Before overview the evolution of the user models, a definition of user modelling is needed. There are at least two different perspectives which might answer this question. One of these is related to AI, and it considers user modelling as the process through which systems gather information and knowledge about users and their individual characteristics. Therefore, a user model is considered a source of information about the user of the system which contains several assumptions about several relevant behaviour or adaptation data. However, in this thesis the HCI research perspective is taken, which is defined by Pohl [127] as follows:

User Model, by Pohl [127]

> *"It refers to an a-priori model of the users of a computer system that the system designer has in mind, or to the assumed models that users will probably develop of the system and the tasks they can perform using the system".*

Nevertheless, this definition and the AI perspective coincide in the idea that every system must use information about the user to be able to see and react to their different problems and needs and improve the system purpose. This section analyses the most significant user models in the past 15 years.

2.2.2 User models

2.2.2.1 1991: Jon Orwan and the Doppelgänger system

Doppelgänger is a user modelling system that performs inferences upon user data and makes this information available to applications. The system allows users to modify their models, it makes implicit generalizations about the data (which is gathered through different channels) and provides an extensible architecture [118].
User data is gathered through a continuously operative sensor network which senses users' everyday activities (see Listing 2). Hence, both long-term and short-term information about the user are stored. Sensors are integrated in users' activities. However, Doppelgänger acknowledges the imperfection of real world data. Data streams are usually incomplete or erroneous. The main objective of the Doppelgänger system is to recover from these imperfections through several learning techniques:

- The Beta distribution [76], which is used to determine the preference strength, probability of accuracy and the confidence of an estimation.
- Linear prediction, to predict a possible next event.
- Markov models, which represents user's behaviour through several states and all possible transitions between them with the corresponding probability.

The system maintains an accuracy estimation for each sensor, which helps the system to decide a confidence metric for the gathered data.

```
(object orwant location (place 344) (time 779562701)
(id active-badge))
```

Listing 2: Message from a sensor to the server [119].

The user models are represented in SPONGE, a LISP based data structure manipulated with C and Pearl programs [119] (see Listing 3). Models are stored as Unix directories consisting of domain submodels and conditional submodels. The first one contains information about the user behaviour (i.e., location, preferences, and so forth); the second group contains triggering information for deciding actions when certain situations are met. Besides, each user model is conceptually represented as a point in a high dimensional space in which the dimensions are determined by the number of sensors in the network.

```
(object orwant primary
  (object biographical_data
    (string_binding "true name" "Jon Orwant")
    (string_binding "e-mail address" orwant@media.mit.edu)
  ...)
  (object control
    (int_binding "doppelganger ID" 4))
...)
```

Listing 3: Orwant user model [119].

2.2.2.2 **2001: Gerhard Fischer**

In 2001 Fischer [82] reviews the user models of the past 10 years. He describes how using computers in HCI environments has been always modelled as a user-computer couple. These elements are modelled as an explicit connection which represents the communication between them. New and modern interfaces such as windows, menus, pointers, colours, sound and touch screens have enlarged this communication line thanks to their capabilities.

Furthermore, in addition to the possibilities of new design approaches, knowledge-based architectures in HCI explore the possibility of implicit communication channel. The required knowledge considers the problem domain, communication processes and the communication agent. Users are part of the communication agent group. Fischer defends the idea that there are many types of users. Besides, their needs change with the experience and through time. Hence, a simple user classifications (e.g., *novel*, *intermediate*

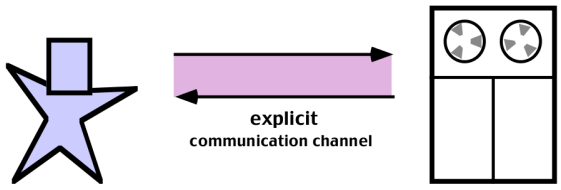

Figure 2.4: The HCI channel [82].

and *expert*) is not enough to characterize users in complex environments. Nevertheless, despite Fischer remarks the significance of each agent, he does not establish which agent capabilities are important to face the problem of modelling a user.

2.2.2.3 **2002: Gregor et al.**

In 2002, Gregor et al. [86] focus their approach on a certain groups of users: the elderly. A three group classification is presented. In the first group there are the fit older people, who do not suffer from any disability. The second group is formed by older fragile people who have one or more disabilities. Finally, the last group encompasses the older and people with disabilities whose capabilities to function depend on other people. In this case, the authors identify several user capabilities:

- Physical, sensory and cognitive capabilities.
- The ability to learn new techniques (cognitive).
- Memory problems (cognitive).
- The environment can affect several elderly capabilities.
- Elderly experience (as a positive fact).

On the other hand, Gregor et al. [86] consider that, as people grow older, their capabilities change. This process encompasses a reduction of cognitive, physical and sensory functions depending on the individual. This diversity is a significant issue for modelling users and designing computing systems.
Figure 2.5 shows an adaptable user interface which takes into account these capabilities.

2.2.2.4 **2003: Gauch et al.**

Towards the goal of personalized navigation of online information Gauch et al. [83] provide a user ontology for dynamically modelling the user browsing. The ontology is formed by several concepts which are weighted indicating the user's perceived interest in the corresponding concept. These concepts are related with surfing experience (i.e., the

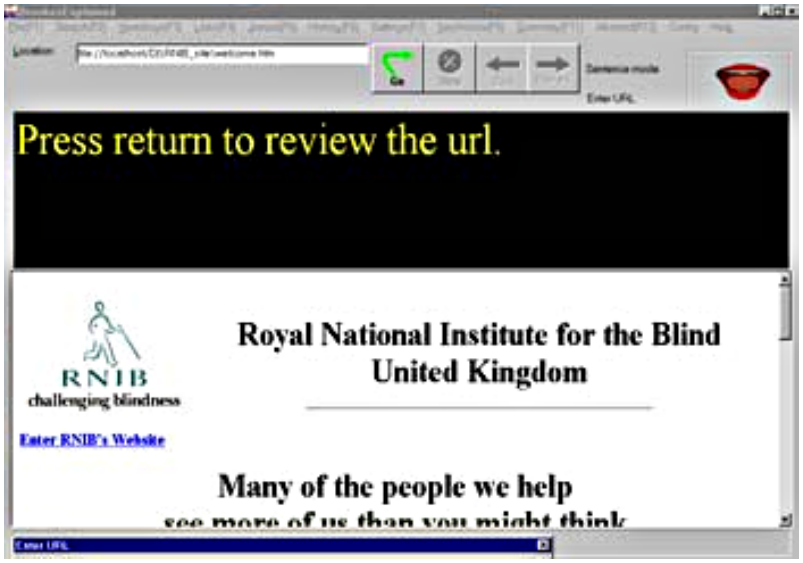

Figure 2.5: Adaptable browsing interface [86].

content, length and time spent) on each Web page and classified into the reference ontology. Hence, the user profile is created automatically. Thus, the user profile information is collected implicitly without user feedback, as the ontology's concepts are automatically weighted considering the amount of related information from the user browsing.

2.2.2.5 **2003: Razmerita et al.: The OntobUM Ontology**

Focused in the context of Knowledge Management System (KMS) Razmerita et al. [128] present OntobUM, a generic ontology-based architecture for user modelling. The model is generated through two different ways:

- *Explicitly*, using a user profile editor. Therefore, the user has to provide some information.
- *Implicitly*, information maintained by several intelligent services which maintain and update the information about the user considering the user's behaviour with the services and provide adapted services based on user's preferences.

The architecture of the presented ontology is composed of the following ontologies:

- *The User Ontology*, which structures the different characteristics and preferences of the user.
- *The Domain Ontology*, which defines several concepts about the domain.
- *The Log Ontology*, which manages the semantics of the interaction between the user and the whole system.

Authors identify several users' characteristics that are relevant for a KMS under the Behaviour concept. Nevertheless, most of the user ontology is generic and it is available to be used in other application domains.

2.2.2.6 **2005: Hatala and Wakary and the Ec(h)o system**

Ec(h)o is an ontology-based augmented audio reality system for museums which aims to maintain rich and adaptive output information. The main purpose of this work is to address the problem of supporting experience design and functionality related to museum visits through user models combined with augmented reality and tangible user interface system. Hatala and Wakkary [91] find several challenges for capturing rich context information. For the presented museum scenario, social, cultural, historical and psychological factors are significant for the user experience. In this field, the argumentation made by is remarked as relevant. Dourish states that activities and context are directly and dynamically linked Dourish [74][75]. This concept is called *embodied interaction*.
The core of the ec(h)o's reasoning module is a dynamically updated user model [143]. The ruled-based model changes as the user moves through the museum and selects several audio objects. This models enables developers to consider which inputs influence user interests. In the ec(h)o system there are two ways of updating the model: the user movement and a selection of an audio object. These actions have different effects on the model of the user interests (i.e., influence of initial interest selection, of object selection on user interest and of location change).
As it occurs with recommender systems, user's interest are vital for the concept ontology. These concepts are weighted in the ontology as concepts which represent the user's likes within the environment. Besides, an interaction history is maintained recording the way the user interacts with the museum. In addition to these characteristics the user type is also considered. Hence, the system is allowed to characterize the user experience with the environment. It classifies users into three different categories:

- The *avaricious* visitor, who wants to see as much as possible in a sequentially way.
- The *selective* visitor, who is more selective with the concepts he/she is interested in.
- The *busy* visitor, who prefers to not spend much time and get a general vision of the exhibition.

2.2.2.7 **2005: Fernando Pereira**

Within a video adaptation and quality of experience evaluation scenario, Pereira [121] studies a user characterization through three different dimensions: sensory, perceptual and emotional. First of all, Pereira establishes the difference between sensations and perceptions as follows:

- *Sensations* are monomodal, more low-level, physical and less related to the real world composition than perceptions. They regard the simple conscious experience for the corresponding physical stimulus (e.g., light variation and eyes reaction to this change). They are related to the first contact between a human and the surrounding environment.
- *Perceptions* are multimodal, and they are part of the cognition process (knowing and learning) and regard the conscious experience and identification of objects.

On the other hand, emotions are considered as central in a communication and entertainment process. Therefore, Pereira proposes a triple layered Sensation-perception-emotion (SPE) user model for the evaluation of the quality of experience in the consumption of multimedia content.

2.2.2.8 **2007: Heckmann et al.**

A different approach is implemented by Heckmann et al. [93]. Divided into four main groups (emotional state, personality, characteristics and physiological state), the authors present the General User Modelling Ontology (GUMO), an ontology model to characterize users capabilities within adaptive environments. A significant user characteristic that is taken into account in this work is the stress. In the adaptive interfaces domain it is needed to pay special attention to the consequences of each adaptation. But the stress is not only determined by this process. It is also derived from several user experiences, as the current context state (e.g. traffic, noise, surrounding people, and so forth [50]). Figure 2.6 illustrates the model presented by Heckmann et al.

2.2.2.9 **2007: Persad et al.**

Persad et al. [124][125] relate user capabilities and product demands as a tool to evaluate the product design (see Section 1.2). Persad et al. remark four main components to consider when dealing with interaction between people and technology: the user, the product, the environment or context, and the set of activities or tasks that define the interaction. The authors try to assess an adaptation degree between users and the designed

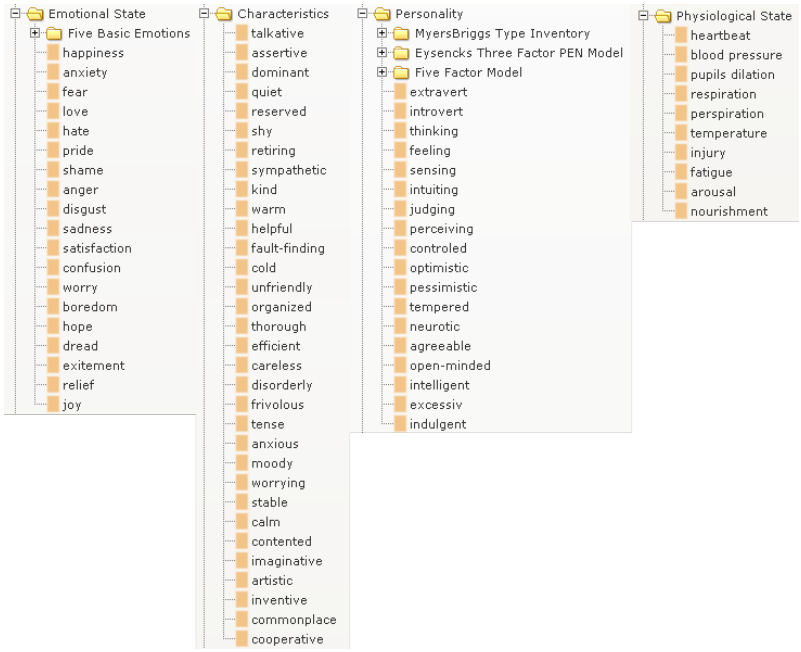

Figure 2.6: Several GUMO user model property dimensions [93].

products using different compatibility measures. These measures can be assessed on different levels of human capabilities, including sensory, cognitive and motor. The concepts of user capability and product demand provide a useful framework for analysing the user-device compatibility. The product demand levels are considered multidimensional and they are set by the interface attributes of the product itself. For example, a product's text display will be designed with a certain text size, font, and colour contrast. The combination of these attributes define the visual demand level within the user visual capabilities. Similarly, other combinations of product attributes command several cognitive and motor demands.
Persad et al. also reviewed functional classifications and experimental studies to identify the most relevant low-level skills for designing products within the cognitive, motor and sensory domain. This is highly related to the ICF functions described in Section 1.1.2.

Sensory capabilities

Visual capabilities: Several sensory capabilities are known to deteriorate with ageing [123]. Thus, Persad et al. state that the following functions seem to account for most of visual disability:

- Visual acuity.
- Contrast sensitivity.

- Colour perception.
- Useful field of view.
- Stereopsis.

Hearing capabilities: Loss of hearing capabilities may directly affect the speech interaction with the device. The main low-level hearing functions to guarantee the interaction are the following:

- Pure tone detection thresholds.
- Speech detection and recognition discrimination thresholds.
- Sound localization.

Cognitive capabilities

The product's user interface must be usable and accessible enough to guarantee that users easily understand the interaction. The following capabilities are related to the human cognitive domain:

- Working memory performance.
- Long term memory.
- Mental models, planning and problem solving.
- Language and communication capabilities.

Motor capabilities

Upper limb capabilities: There are many conditions that affect manipulating a product (e.g., arthritis, stroke, multiple sclerosis, head injury, cerebral palsy and missing or damaged limbs). These problems directly reduce grasp forces, range of motion and fatigue thresholds [125].
The following areas are highlighted within motor capabilities:

- Reach ranges for each arm.
- Grasping, dexterity and force exertion.
- Two handed actions and coordination.

Feature type	Examples
Product chassis	Handles, gripping surface.
Displays and indicators	Visual and auditory displays.
Controls and control groups	Discrete controls (button, switch) and continuous controls (slider, knob, thumb, wheel, dial, joystick). Control groups *Keypad*.
Material/media input and output	Slots (toaster slots), powered and un-powered bays and trays, doors, lids and covers.
Connectors for energy and data	Power and data connectors
Software interfaces	Navigation menus and GUI objects.

Table 2.2: Product interface classification by Persad et al. [125].

Gross body movement capabilities: Usually products require certain user mobility degree.

Besides, Persad et al. provide six general categories for product features and their interface classification. For this classification their toaster case study is considered, *"which is used to demonstrate the capability-demand interaction"* (see Table 2.2).

2.2.2.10 **2007: Golemati et al.**

Golemati et al. [85] present an ontology which, considering past literature solutions, aims to reduce the intrinsic problems of user modelling: ad-hoc modelling processes, the required amount of work to model users and the possibility of errors by omitting several user's characteristics. To this end, authors present an extensible, comprehensive and general ontology whose design is addressed through a top-down approach by firstly collecting static information about the user. Next, the ontology designers analyse the semantics of the profile models and suggest concepts that would adequately model them. It is remarkable that this work is focused on static user characteristics, although they consider the possibility of incorporating dynamic and temporal characteristics.

2.2.2.11 **2008: Casas et al.**

Another approach is the one presented by Casas et al. [63]. In this case the authors work under the *Persona* concept which is introduced to distinguish different user groups within an adaptive user interfaces domain. Originally this concept was introduced by Cooper and Saffo in 1999 by the following definition:

Personas, by Cooper and Saffo [71]

> *"Personas are not real people, but they represent them through a design process. They are hypothetical archetypes of real users".*

Casas et al. distinguish between two categories of people:

- *Primary*: those who represent the main group and use primary interfaces.
- *Secondary*: those who can use primary interfaces, but with several extra needs.

By assigning random values to several characteristics (e.g., age, education, profession, family conditions, disabilities and technological experience) authors are capable of covering a wide range of potential users. However, the most significant contribution is that, instead of being focused on users capabilities, they consider users needs. To that end they build a user profile supported by four main bases:

- The user level, which indicates the ability of the user to face the system.
- Interface, for the interaction mechanism to be used by the user.
- Audio, to indicate the audio volume levels.
- Display, which includes usual display controls (contrast, colours, brightness and so on).

This approach is focused on the solution, on the adaptation itself. It is a perspective which allows users to configure the interaction based on their capabilities. This helps applications designers because user capabilities are not directly taken into account in the model as medical or technical aspects. Hence, there is no need to be experts or have any physiological knowledge or experience about users disabilities. Another advantage is that each user can manage his/her own profile. Thus, they can configure their preferences and capabilities on their own.

2.2.2.12 **2012: Evers et al.**

Several studies, as the research presented by Evers et al. [78], recognize that it is complex to perform interfaces adaptations without bothering the user. On the one hand, adapting an interface without the participation of the user might lead to an unsatisfactory result. On the other hand, asking too much for participation might bother the user. Following this stress perspective Liao et al. [111] present an unified probabilistic decision model based on Influence Diagrams for modelling user stress levels. These levels were inferred by probabilistic inferences of several sources data (e.g., heart rate, mouth openness, head movements, or pupils monitoring).

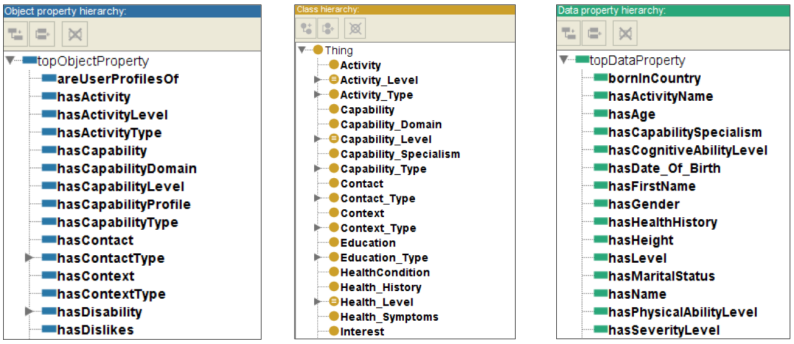

Figure 2.7: An overview of the User Profile Ontology classes, object properties and data properties [137].

2.2.2.13 2012: Skillen et al.

Within an application personalization within mobile environments Skillen et al. [137] present a User Profile Ontology which is able of modelling dynamic components. The ontology considers both static and dynamic aspects of the user mainly focused on his/her behaviour changes. The user capabilities are also taken into account for the user profile. Capabilities are defined as the extent to which the user has an ability (i.e., physical, emotional or cognitive) to carry out some activities. User's interests and several context parameters are also considered in the ontology to cover context-aware environments. Figure 2.7 overviews the User Profile Ontology classes, object properties and data properties presented by Skillen et al..

2.2.2.14 Generic User Modelling Systems

There is another generic approach for modelling users known as Shell Systems. More focused in the field of AI, these solutions consider the user model as a source of information which are built on assumptions about relevant user aspects or behaviour [127]. In the following lines several significant researches in the field are highlighted in order to take into account the difference between generic user modelling approach (user modelling shells) and the approach followed in this dissertation, which is related to HCI.

Heckmann [92] differentiates between the model, which is where the user data is collected, and the modelling system, which is the module that manages the model. Besides, he remarks the following two definitions from Wahlster and Kobsa [143] previous work:

User Model (I), by Wahlster and Kobsa [143]

> *"A user model is a knowledge source in a system which contains explicit assumptions on all aspects of the user that may be relevant to the behaviour of the system. These assumptions must be separable by the system from the rest of the system's knowledge".*

User Model (II), by Wahlster and Kobsa [143]

> *"A user modelling component is that part of a system whose function is to incrementally construct a user model; to store, update and delete entries; to maintain the consistency of the model; and to supply other components of the system with assumptions about the user".*

Assumptions are deeply studied and depicted in [127]. In this section, a quick overview of how these systems behave is performed to just take into account the difference between generic user modelling approach (user modelling shells) and the approach followed in this dissertation, which is related to Human-Computer Interaction.
In 1986 General User Modelling System (GUMS) is presented. This software allowed developers to make user-adaptive applications by defining several stereotypes, facts and rules to reason with [79]. GUMS supports the addition of new facts in runtime, manages facts inconsistencies and answers the application about assumptions about the user [106]. The following systems are several examples of generic user systems developed in the following years (chronological ordered):

- Doppelgänger, which uses several learning techniques for generalizing and extrapolating sensor data for the development of the user model [118].
- User Modelling Tool (UMT), which supports the definition of hierarchically ordered stereotypes about the user, rules for user model inferences and contradiction detection [54].
- TAGUS, which uses first-order formulas to represent assumptions about the user [120].
- The um toolkit, which models not only assumptions but beliefs, preferences and other user characteristics in attribute-value pairs [104].
- BGP-MS, which permits user and groups of users assumptions [107].

2.2.3 Users Models Comparison

In this section a description of our user modelling requirements nurtured by the earlier works is described. As mentioned before, there are several perspectives regarding the user modelling requirements. In this dissertation the HCI perspective is taken. This means, as Pohl [127] states, that the user model refers to user characteristics using a certain system. In Chapter 3 the proposed models for user, context and device are described. These models have been designed considering each of these entities as a set of characteristics that define them. Hence, the adaptation platform is able to evaluate the combination of these characteristics and perform the corresponding adaptation. Thus, the HCI interpretation of user modelling suits the main objective remarked in Section 1.3.

In the following lines the amount of different domains addressing user modelling are highlighted. From product design to multimedia and user interfaces adaptation, every approach follows the same purpose: to consider several user characteristics to improve the system and user's satisfaction and product or service usability. However, although these solutions share the same objectives, the considered characteristics differ. For ubiquitous and more context-aware domains, activities are taken into account. For example, Razmerita et al. [128] discuss an ontology based architecture, which aims to be generic by collecting user data through two different ways (*explicitly* and *implicitly*). Golemati et al. [85] also take an ontological point of view to avoid the problem of domain dependency (among others) by designing a more general, comprehensive and extensible ontology.

Gauch et al. [83] remark in the presented ontology the importance of time. Regarding the studied domain (web browsing) time is significant because it might help characterizing the user considering the spent time reading an article or visiting a website. Well known and popular recommendation systems, as YouTube, utilise this information combined with different explicit and implicit sources from the user interaction to make proper recommendations [72].

User activities have also been considered as relevant for many authors in the literature. The first example is the Doppelgänger system [118] (1991), which uses activities for sharing relevant user information to different applications. In the same way, Persad et al. [125], Heckmann et al. [93] and Skillen et al. [137] model activities to take user's behaviour and interaction into account for the proposed classifications and systems. For Hatala and Wakkary [91] activities are also vital within context-aware environments.

As occurs with context modelling (see Section 2.3) many different techniques are available for the model representation. This usually depends either on the developer, because of his/her experience, or in the system's technical characteristics. For example, if the

system is able to performs inference with the user data an ontology based representation could be more helpful than an object based one. Strang and Linnhoff-Popien [140] demonstrated that ontological modelling is more appropriate for ubiquitous computing environments.

It is also common to model physiological related characteristics of the user. For example, the works by Gregor et al. [86] and Persad et al. [125] consider physical, cognitive an sensory capabilities. Skillen et al. [137] also model several user abilities for performing different tasks and activities. The problem is that being aware of these capabilities is difficult and, in some cases, poorly practical. For example, measuring the sight capability of one individual requires physiological experience or advice. Besides, people with the same affection do not respond in the same way. A person who was born blind would interact differently with the environment than another who has been losing sight during his/her life. The precise same disability (e.g., tunnel vision) might affect different people in many different ways depending on their personal skills (e.g., adaptability, orientation, and so forth). This might lead to an idea. What if, instead of modelling physiological skills (disabilities), we were able to model user's capabilities? The first approximation for this is found in Casas et al. [63] work. Casas et al. present a user profile which abstracts from physiological aspects and lets users manage and configure their own profile. On the contrary, Skillen et al. [137] model user capabilities as a set of abilities which allow users to perform some task or activities. Although this perspective covers many user capabilities, it still needs a deeper understanding of physiological user capabilities.

On the other hand, Fischer [82] states that it not only is difficult to model users because of the wide range of different types of people that exist. He also considers that each individual changes with experience and through time. For example, old people's capabilities decrease with ageing. This idea is shared with Gregor et al. [86], whose work is centred around the elderly. Heckmann et al. [93] not only consider that users might evolve (from an AI perspective), but he also takes new context information from the inference process. This also opens a new point of view that we address in Section 1.2.1 and it is about taking context as a significant user's environment entity that might directly influence the user's capabilities. In other words, *users change through experience, time and, in concrete situations, due to the current context characteristics*. For example, an individual might not suffer from any mobility disability, but in a crowded street would be difficult to perform several daily activities (just walking could be difficult). Razmerita et al. [128] address this issue considering an implicit user information collecting process. This, of course, deals with the concept of dynamism. Finally, Evers et al. [78] consider that respecting user's interactive behaviour with applications needs to be taken into account. On the other hand,

Solution (Domain)	Ontologies	User characteristics							
		A	C	Ex	I	E	P	S	L
2002, Gregor et al. [86] (Inclusive design)			✓	✓					
2003, Gauch et al. [83] (Automatic profiling)	✓				✓				
2003, Razmerita et al. [128] (KMS)	✓	✓			✓		✓		
2005, Hatala and Wakkary [91] (Tangible interfaces)	✓				✓		✓		✓
2005, Pereira [121] (Multimedia adaptation)						✓			
2007, Persad et al. [125] (Product design demands)		✓	✓						
2007, Golemati et al. [85] (User profiling)	✓	✓		✓	✓		✓		
2007, Heckmann et al. [93] (Ubiquitous applications)	✓	✓				✓	✓	✓	
2008, Casas et al. [63] (AUI)			✓	✓					
2012, Evers et al. [78] (Adaptive applications)								✓	
2012, Skillen et al. [137] (Mobile environments)	✓	✓	✓		✓				✓

Table 2.3: Related work for the user modelling approaches. Under the user characteristics heading Activities or behaviour, Capabilities, Experience, Interests, Emotions, Personal, Stress and Location information are presented.

Pereira [121] analyses the differences between emotional and perceptional user characteristics.

Table 2.3 summarizes the analysed approaches for user modelling, emphasizing the modelled user characteristics and domains.

2.3 Context Modelling

2.3.1 What is Context?

Context is often defined according to Dey issued definition:

Context (I), by Dey [73]

> *'Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".*

In the past decades there were many definitions of context [46]. Nevertheless, the one stated by Dey is one of the most popular and extended definitions, enabling developers to easily enumerate those elements which take part in the context for a certain application domain. Dey states that:

Context (II), by Dey [73]

> *"If a piece of information can be used to characterize the situation of a participant in an interaction, then that information is context".*

A proposed example explains this definition: *"Take the canonical context-aware application, an indoor mobile tour guide. The obvious entities in this example are the user, the application and the tour sites. We will look at two pieces of information – weather and the presence of other people – and use the definition to determine whether either one is context. The weather does not affect the application because it is being used indoors. Therefore, it is not context. The presence of other people, however, can be used to characterize the user's situation. If a user is travelling with other people, then the sites they visit may be of particular interest to her. Hence the presence of other people is context because it can be used to characterize the user's situation." [73]*
By this example, it is seen how different pieces of information are analysed to determine if they belong to what Dey considers as context. These definitions are based on research experience. Section 2.3.3 shows how modelling and defining context has evolved in the past 20 years.

2.3.2 A Chronological Review of the Evolution of Context Management

Figure 2.8 shows a chronological evolution for context modelling for the last 15 years.

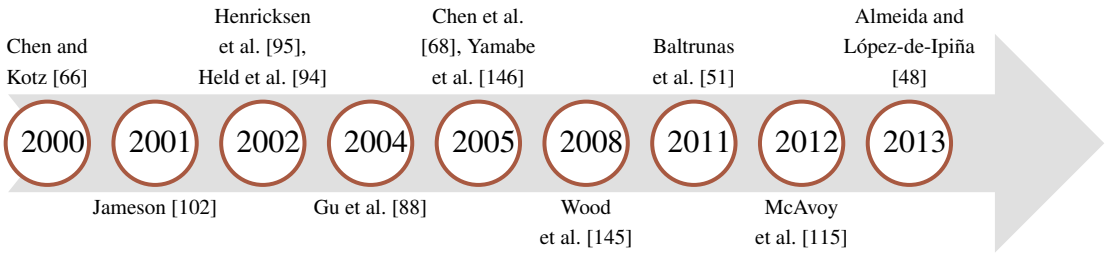

Figure 2.8: The chronological view of the evolution of remarkable context models considered in this dissertation.

2.3.3 Context Models

The first definition of *context-aware systems* is given by Schilit and Theimer [134] as "the ability of mobile user's applications to discover and react to changes in the environment they are situated in". Dey [73] also defines *context-aware systems* as those systems which, using context data, provide significant information and/or services to the user where the relevancy of the given information depends on the user task. Schmidt et al. [136] consider several issues about context modelling. Authors emphasize the excess of abstraction about context-aware systems and environments which causes a lack of models to be compared . Therefore, they present a working model for context-aware systems categorized into human and physical environment factors. In 2001 Jameson [102] studies how context-aware computing represents a challenging frontier for researchers. In this work, information about the environment, the user current state, longer term user properties and the user behaviour are compared in order to take the correct adaptation decision. Several works focused on user interface adaptation area base their processes on context changes as triggers. However, they lack a common model of context in their platforms [58][117]. In the following subsections a review of the most popular context-aware systems is presented, mainly focused on the modelled context parameters, techniques, domains and dependencies. Besides, several authors define context and context-awareness through their own experience. All these models and definitions have been considered for the context model proposed in Chapter 3. Table 2.4 shows several significant features of each approach, and a comparing analysis is performed.

2.3.3.1 2000: Chen and Kotz

Chen and Kotz [66] define context as follows:

Context, by Chen and Kotz [66]

> *"Context is the set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user".*

Besides, the definition of active and passive context-aware computing is also given. To Chen and Kotz, active context awareness is:

Active Context Awareness, by Chen and Kotz [66]

> *"(. . .) an application automatically adapts to discovered context, by the changing in the application's behaviour".*

On the contrary, they define passive context awareness as:

Passive Context Awareness, [66]

> *"(. . .) an application presents the new or updated context to an interested user or makes the context persistent for the user to retrieve later".*

Based on the work by Schilit et al. [133], Chen and Kotz [66] consider *time* as an important and natural context feature for many applications. Besides, they introduce the term *context history*, which is an extension of the time feature recorded across a time span.
A significant problem remarked in this work is the impossibility to exchange context information between the studied context-aware systems due to the way they model the environment. Furthermore, as *location* is one of the most modelled context features, Chen and Kotz provide a study of several aspects that should be taken into account when researchers face the problem of modelling it.

2.3.3.2 **2001: Anthony Jameson**

Jameson [102] analyses in 2001 how context-aware computing represents a challenging frontier for researchers in the field of AUIs. In this work information about the environment, the user's current state, longer term user properties and the user behaviour are compared in order to take the correct adaptation decision. The idea is to compare several scenarios. In the first one, only information about the environment is considered. In the following scenarios the user's current state, behaviour and long-term properties are taken into account. Thus, the results conclude that considering a wider range of user information can help context-aware systems designers.

2.3.3.3 **2002: Henricksen et al.**

The approach followed by Henricksen et al. [95] makes the following notes about context in pervasive environments:

- Context information exhibits *temporal characteristics*. Context can be static (e.g., birthday) or dynamic (e.g., user location). Besides, the persistence of the dynamic information can easily change. Thus, it is justified that the static context should be provided by the user, while the dynamic one should be gathered by sensors. Past historic and a possible forecasting of future context are also taken into account as part of the description of the whole context description.

- A second property of the context information is its *imperfection*. Information can be useless if it cannot reflect a real world state. It also can be inconsistent if it contains contradictions, or incomplete if some context aspect are unknown. There are many causes to these situations. For example, information can change so fast that it may be invalid once it is collected. This is obviously because the dynamic nature of the environment. Besides, there is a strong dependency on software and hardware infrastructures, which can fail any time.

- Context has *multiple alternative representations*. Context information usually comes from sensors which speak different languages. For example, a location sensor can use latitude and longitude physical magnitudes while the involved application works with a logical representation of location.

- *Context information* is highly *disassociated*. There are obvious connections between some context aspects (e.g., users and devices). However, other connections need to be computed with the available information.

This work also indicates the dependency between context models, the scenarios and use cases of the application domains. Authors extract several context parameters to consider:

- User activity, distinguishing between the current one and the planned one.

- Device that is being used by the user.

- Available devices and resources.

- Current relationships between people.

- Available communication channels.

2.3.3.4 **2002: Held et al.**

In 2002 Held et al. [94] discussed about the necessity of content adaptation and representation formats for context information in dynamic environments. A significant perspective is the justification of modelling not only the user but the network status and device context information as well. The following parameters are considered as relevant context information:

- *Device*: basic hardware features (e.g., CPU power, memory and so forth), user interface input (e.g., keyboard and voice recognition), output (e.g., display and audio), and other particular specifications of the device (e.g., display resolution or colour capability).
- *User*: service selection preferences, content preferences and specific information about the user.
- *Network connection*: bandwidth, delay, and bit error rate.

Authors also present several requirements concerning the representation of context information. Accordingly, they defend that a context profile should be:

- Structured, to ease the management of the amount of gathered information and remark relevant data about the context.
- Interchangeable, for components to interchange context profiles.
- Composable/decomposable, to maintain context profiles in a distributed way.
- Uniform, to ease the interpretation of the information.
- Extensible, for supporting future new attributes.
- Standardized, for context profile exchanges between different entities in the system.

2.3.3.5 **2004: Gu et al.: The SOCAM Ontology**

In 2004 Gu et al. [88] design the Service-Oriented Context-Aware Middleware (SOCAM) architecture for designing and prototyping applications in an Intelligent Environments (IE). Built on top of the Open Service Gateway initiative (OSGi) [25] architecture, such middleware consisted of the following components:

- Context providers, which abstract context information from heterogeneous sources and semantically annotate it according to the defined ontology.

- The context interpreter, which provides logic reasoning to process information about the context.
- The context database, which stores current and past context instance data.
- Context-aware applications, which adapt their behaviour according to the current context situation.
- The service-locating service, which allows context providers and interpreters to advertise their presence for users and applications to locate them.

Gu et al. use OWL to describe their context ontologies in order to support several tasks in SOCAM. As the pervasive computing domain can be divided into smaller sub-domains, the authors also divided the designed ontology into two categories:

- An *upper ontology*, which captures high-level and general context knowledge about the physical environment.
- A *low-level* ontology, which is related to each sub-domain and can be plugged and unplugged from the upper ontology when the context changes.

As a result, the upper ontology considers person, location, computational entity and activity as context concepts.

Gu et al. [89] also present an OWL based model to represent, manipulate and access context information in smart environments. The model represents contexts and their classification, dependency and quality of information using OWL to support semantic interoperability, contextual information sharing, and context reasoning. The ontology allows to associate entities' properties with quality restrictions that indicate the contextual information quality.

2.3.3.6 **2005: Chen et al.: The CoBrA Ontology**

Another work under a similar approach is the one performed by Chen et al. [68]. Authors introduce the Context Broker Architecture (CoBrA) ontology based system, which provides a set of semantic concepts for characterizing entities such as people, places or other objects within any context. The system provides support for context-aware platforms in runtime, specifically for Intelligent Meeting Rooms. The context broker is the central element of the architecture. This broker maintains and manages a shared context model between agents (applications, services, web services, and so forth) within the community. In intelligent environments participating agents often have limited resources and capabilities for managing, reasoning and sharing context. The broker's role is to help these agents to reason about the context and share its knowledge. The presented ontology relies on:

- Concepts that define *physical places* and their *spatial connections*.
- Concepts that define *agents* (humans and not humans).
- Concepts that define the *location* of an agent.
- Concepts that describe an agent *activity*.

2.3.3.7 **2005: Yamabe et al.: The Citron Framework**

In 2005 Yamabe et al. [146] present a framework for personal devices which gathers context information about the user and his/her surrounding environment. Muffin, a personal sensor-equipped device is designed. Using it, several context parameters are gathered.
Sensor information is also considered for evaluating several user high-level context information. For example, accelerometer readings might recognize a walking or running activity, shaking and rotating, an so forth. The microphone is not only used to measure the ambient noise. It is also useful for detecting the place where the user is (e.g., meeting room, restaurant, street and so on).
The context acquisition is categorized in two different groups: the *user* and the *environment*. For the user several issues are analysed. For example, *activity recognition* requires the user to use the device in specific ways (it is not the same to use it with hands or waist-mounted). Another problem they encounter is about the *time consuming* process since an event is captured, then processed and finally validated. The last contextual issue deals with the intrinsic complexity and *ambiguity* of context information. For example, the meaning of what is loud might depend on the current situation (e.g., if the user is in a meeting room, or if the user is in the street). For the environment, Muffin suffer several heat problems due to the sensors sensitivity to environment temperature. This way, sometimes the gathered measures are invalid.

2.3.3.8 **2008: Wood et al. and the AlarmNet system**

AlarmNet is an Ambient Assisted Living (AAL) Wireless Sensor Network (WSN) for pervasive adaptive healthcare in assisted living residences for patients with special needs. Wood et al. [145] contribute with several novelties:

- An extensible heterogeneous network middleware.
- Novel context-aware protocols.
- SenQ, a query protocol for efficiently streaming sensor data.

The context-aware protocol uses a two-way network information flow. On the one hand, environmental, system and residents data flow. On the other hand, Circadian Activity Rhythm (CAR) analysis goes back into the system to enable smart power management and dynamic alerts.
Several sensors are used for sensing environmental quality: light, temperature, dust, resident activities (motion sensors) and so on. The devices' queries to the system are negotiated by a Query Manager.

2.3.3.9 **2011: Baltrunas et al.: InCarMusic**

Assuming that context-aware systems adapt to the specific user situation, Baltrunas et al. [51] present a music recommendation system which takes into account the user's mood and the traffic conditions. To this end, the authors design a methodology through which users can be requested to judge several contextual factors (e.g., if current traffic conditions are relevant for a decision making task) and to rate an item when a certain contextual condition is met.
In order to take into account the user's music preference and the influence that context might have into them, context is modelled as several independent factors.

2.3.3.10 **2012: McAvoy et al.**

McAvoy et al. propose an ontology-based system for the managing of context within smart environments. One of the most significant contributions deals with the high-level information managed through the metadata and meaning which are collected by the sensor network [115]. The sensing devices within the smart environment have to be modelled in order to be semantically enriched. Data is formally represented as an ontology by using entities and the relationships which link them together. After the data is collected from the sensors it is passed to an enrichment module where it is made semantically rich. This new enriched data is stored in a semantic repository in the form of triples. The meaning of this information and the metadata are added to the data within this enrichment component.

2.3.3.11 **2013: Almeida and López-de-Ipiña: The AMBI2ONT Ontology**

Almeida and López-de-Ipiña [48] consider two common problems dealing with ambiguity in the area of context modelling: the *uncertainty* and the *vagueness*. Uncertainty models the likeness of a certain fact, while the vagueness represents the degree of membership to a fuzzy set. The uncertainty is represented by a certainty factor.
Due to the nature of the process of collecting data from the environment, the proposed ontology has been designed to support two types of uncertainty:

- Uncertain data: This kind of uncertainty is generated from the capture of data from sensors due to the imperfect nature of the devices.
- Uncertain rules: It occurs in the execution of the rules.

To reason over the ambiguous information the JFuzzy Logic Open Source fuzzy reasoner has been adapted to support uncertainty information.

2.3.3.12 **Context Models Comparison**

The previous section has reviewed several significant context-aware systems and the followed approaches for modelling relevant context parameters of the environment depending on the application domain. In fact, this is one of the main problems in context modelling: the lack of model independence from similar domains and, also, the lack of models to be compared. Despite the fact that sometimes the primary domains are similar (context-aware computing, pervasive environments and ubiquitous computing) regarding the necessity of managing context knowledge, the concrete applications and approaches' domains are different. Here, Henricksen et al. [95] realize about the lack of formality and expressiveness of previous context models.
However, to avoid this problem Gu et al. [89] present an ontology-based solution in which context information is modelled in two separated layers:

- In the upper layer there is an ontology which describes high-level knowledge about the current context and physical environment.
- Under it there is the possibility to add and remove ontologies which model low-level information of the current specific domain.

On the one hand, this approach allows developers to consider richer information, as activities, and abstract knowledge about the current global context. On the other hand, it makes possible to model specific knowledge of the current sub-domain. Besides, the possibility to plug and unplug these low-level ontologies makes this solution powerful. The solution provided by Gu et al. [88] is significant because of the following reasons:

- It considers high-level information on top of a more specific and domain dependent sub-model.
- Activities are modelled as a relevant concept for context in the upper ontology.
- The modelled entities are related. Persons are associated with locations and activities, each location is linked with indoor or outdoor entities, activities can be categorized into scheduled or deduced ones, and so forth.

Chen and Kotz [66] survey several context-aware computing solutions. The remarked context characteristics (*location* and *time*) are given more as an advice, since they do not provide a reference model. Nevertheless, they introduce several novelties like the term *context history*, which might be useful for future predictions about user's behaviour and trends.
Modelling high-level information allows to perform deeper computations taking into account behavioural characteristics, trends information, inferred knowledge from small pieces of information combinations, and so on. As can be seen in Table 2.4 many authors work with high-level data in their context-aware systems.
On the other hand, physical context parameters are frequently modelled in the analysed literature. *Location*, *time* and *environment conditions* (e.g., temperature, pressure, light and noise) are usually modelled to achieve final system's goal (e.g., adapting the user interface or recommending items or services). Besides, several approaches take user related characteristics to fulfil their purposes. For example, Schmidt et al. [136] consider not only physical environment as context information but also several human factors categorized as follows:

- User information, which gathers knowledge about user's habits, emotional stated and bio-physiological conditions.
- Social environment, made up by other user's locations, social interactions of the current user and knowledge about the behaviour of groups of people.
- Task, taking into account spontaneous activities, engaged tasks and general goals.

A user modelling approach which considers emotional issues of the user is presented by Pereira [121]. The author distinguishes between emotions and perceptions to separate both concepts for modelling processes. These approaches are found useful for recommending systems in which user mood and psychological state are relevant to filter multimedia content [51]. Following a similar perspective Evers et al. [78] analyse the user's participation in adaptive applications. Several concepts that should be taken into account regarding the user behaviour and interaction with the application are presented. Another similar approach regarding the user's more psychological aspects is presented by Liao et al. [111]. Here the authors present a model for modelling user stress levels. This is related to the concept of Considerate Computing, term that it is explained in [84].
Schmidt et al. [136] also remark the social environment as relevant for context modelling. More related to recommending environments, non explicit information about user's likes need to be computed. Similar approaches in these environments tend to avoid recommending systems intrinsic problems, like the so-called *cold start problem* [64][65].

Another relevant point remarked by Schmidt et al. [136] are the user's tasks. Several authors consider activities relevant for modelling context [95][88][45]. Activities enrich context information about the user [128]. It is common to model user activities in the user model (see Table 2.3).
Finally, sometimes the collected data might lead to misunderstandings or non-trustworthy data. Almeida and López-de-Ipiña [48] consider *ambiguity* and *uncertainty* in their work and present an ontology-based process which allows to model them within a smart environment.

Table 2.4 shows the differences between each reviewed solution for modelling context. It is difficult to gather all the approaches in a unique table. Therefore, Table 2.4 just emphasizes several differences, although these solutions have more characteristics modelled. For example, the work by Almeida and López-de-Ipiña [48] is focused on modelling context uncertainty and vagueness due to the nature of context data from sensors.
Besides and as it is deeply discussed by Strang and Linnhoff-Popien [140], modelling the context with ontologies offers the following advantages:

- Ontologies are the most expressive approach to model context.
- Composition and management of the model can be done in a distributed manner.
- It is possible to partially validate the contextual knowledge.
- One of the main strengths of ontologies is the simplicity to enact the normalization and formality of the model.

Regarding the context-aware systems, we cannot go further without remarking the issues of gathering information from different sources or sensors. This information might sometimes be unreliable. Sensors can fail in the collecting process, they also can stop working due to several reasons (e.g., power or malfunction). What is more, every sensor speaks its language (this issue is addressed applying different techniques, like data fusion). Therefore, a process to evaluate the quality and trustworthiness of the collected information should be included in every context-based system as a method to avoid undesired results.

Solution (Domain)	Ontologies	Context parameters									
		L	T	A	R	P	E	S	I	U	H
2000, Chen and Kotz [66] (Context awareness)		✓	✓								
2001, Jameson [102] (Context modelling)		✓		✓						✓	✓
2002, Henricksen et al. [95] (Pervasive computing)				✓	✓			✓	✓		✓
2002, Held et al. [94] (Context awareness)									✓	✓	
2004, Gu et al. [88] (Smart environments)	✓										✓
2005, Chen et al. [68] (Pervasive computing)	✓	✓	✓	✓	✓	✓	✓		✓	✓	
2005, Yamabe et al. [146] (Mobile computing)		✓		✓			✓			✓	✓
2008, Wood et al. [145] (AAL)				✓			✓			✓	
2011, Baltrunas et al. [51] (Recommender systems)							✓			✓	
2012, McAvoy et al. [115] (Smart environments)	✓					✓					✓
2013,Almeida and López-de-Ipiña [48] (Smart environments)	✓	✓					✓				

Table 2.4: Related work for the context modelling approaches. Under the context characteristics heading Location, Time Activity, Nearby Resources, Nearby People, Physical Environment, Infrastructure, User's parameters and High-level information are presented.

2.3.3.13 Modelling Techniques

Strang and Linnhoff-Popien [140] reviewed the context modelling literature in 2004. They differentiate among several paradigms. Here a discussion of several previously used context modelling techniques is presented:

1. Key-value: Maass [113] adopted a X.500 based solution to store location data. This approach is also used in distributed searching systems. Although it is very easy to maintain and handle its main problem is that it makes difficult to build complex structures [140]. Similar solutions follow this key-value approach to identify a context element (key), like location, with an environment variable (value) [135][142].

2. Markup scheme: Based on several derivative Standard Generalized Markup Language (SGML), for example Extensible Markup Language (XML), marking

schemes based models are widespread for modelling profiles. Some extensions are defined as CC/PP [7] standards and UAProf [38]. This kind of context modelling usually extends and completes the CC/PP and UAProf basic vocabularies. In [94] authors present an extension of this model, Comprehensive Structured Context Profiles (CSCP), which provides hierarchy to such schemes supporting the Resource Description Framework (RDF) flexibility to express natural structures of profile information.

3. Graphic models: While Bauer et al. [52] used a Unified Modelling Language (UML) tool to model the context in a air traffic domain, Henricksen et al. [96] presented a graphic model (as an extension of Object-Role Modelling [24]). UML is a widespread general purpose modelling tool with a very powerful graphic component (graphic models): the UML diagram.

4. Object oriented models: Strang and Linnhoff-Popien [140] presented an object oriented model in which context process details are embedded into object level. Data is hidden from other components.Therefore, the access to this context data is just allowed through several interfaces. This approach tries to use the object oriented programming benefits, as re-usability and encapsulation, to cover ubiquitous environment's problems about context. Another example of this approach is the one given by the GUIDE project by Cheverst et al. [69], which is focused on location. In this case the context information is also in the object as accessible states through those methods defined by the object itself and by modifying these states.

5. Ontology based models: As Strang and Linnhoff-Popien [140] discussed, modelling the context with ontologies offers the following advantages:

   - Ontologies are the most expressive approach to model context.
   - Composition and management of the model can be done in a distributed manner.
   - It is possible to partially validate the contextual knowledge.
   - One of the main strengths of ontologies is the simplicity to enact the normalization and formality of the model.

   Multiple ontology based context models have been developed in the past. In the following section several ontology-based relevant models will be discussed.

2.4 Device Models

There are many and different approaches in the literature for devices modelling. Devices capabilities might determine the boundaries of an adaptation process or the consumable resources. Lemlouma and Layaïda [110] considered that an independent device model would probably reduce adaptation process efforts for different context situations. In fact, The World Wide Web Consortium (W3C) reinforces this perspective by warning about the wide range of device capabilities and sizes [11] which define the boundaries of the content that each device can handle. Several techniques, like Device Descriptors, content transformation guides, devices APIs and CC/PP systems help developers to optimize the user experience. These modelling techniques are analysed in the following section.

2.4.1 Composite Capabilities/Preference Profiles

CC/PP [8] is a W3C standard system to express user and device capabilities. Using CC/PP a user will be able to show a specific preference or disability. For example, even though a user's device can display millions of colours, perhaps the user can just distinguish between a small set of colours. The necessity of this system stems from the wide range of web and ubiquitous devices available in the market. These devices have more and more multimedia and Web capabilities. This makes troublesome to Web content providers to service their content to these devices keeping usability and user satisfaction [110].
However, managing a large number of devices is not a new problem. Many approaches have been proposed in the literature to tackle this situation. Most of them are based on content management. This approach considers different presentation alternatives. Depending on the client device, a presentation configuration is served. Hence, in the content serving process there are two options: the server chooses which is the best configuration for the device, and the client decides what to do with the content. This approach is very easy to implement, since every device identifies itself against the server.
CC/PP is based on *profiles*. A profile contains components and attributes. Each component has, at least, one attribute, and each profile has at least one component. The main components are: the hardware platform, the software platform and single applications (e.g., a browser). Attributes can contain one or many values. For example, in case of the hardware platform component we can find the attributes *displayWidth* and *displayHeight*. These attributes have a single value. CC/PP uses RDF as formal language to build these profiles. Table 2.5 shows several advantages and drawbacks of this approach.

Advantages	Drawbacks
A good infrastructure for modelling devices	Device dependent
Content negotiation flexibility	It requires a more mature user preferences definition
Using CC/PP, Web based device developers and user agents can define accurate profiles for their products. Web servers and server proxies can use these profiles to perform the adaptation	
Open to new protocol proposals for profile exchanging	

Table 2.5: CC/PP: several advantages and drawbacks

2.4.2 UAProf

UAProf [38] is concerned with collecting wireless devices capabilities and preferences. This information is provided to content servers to easy the content format selection process. UAProf is directly related to the W3C CC/PP specification and it is also based on RDF. Hence, the document schema is extensible [56][57]. These files, usually served as application/XML mimetype, describe several mobile devices capabilities (e.g., vendor, model, screen size, multimedia capabilities, and so forth). Most recent versions have also information about Multimedia Messaging Service (MMS), video streaming and more multimedia features. UAProf profiles are voluntarily built by the vendors (e.g., Samsung and LG for GSM devices), or by several telecommunications company for CDMA/BREW devices.
The system works as follows:

1. The device sends a header containing a URL and its UAProf within an HTTP request.
2. The server side analyses the received UAProf to adapt the content to the device's display size.
3. Finally, the server takes the decision and serves the corresponding items to the device.

However, this approach has several drawbacks:

- Not every device has a UAProf.
- Not every UAProf profile is available.
- UAProf data can contain schema or data errors.

- There is no industry-wide data quality standard for the data within each field in an UAProf.

2.4.3 Device Description Repository

DDR is a concept proposed by the Device Description Working Group (DDWG), an organization within the W3C. The DDR is supported by a standard interface and an initial vocabulary core about devices' properties. Web content authors will use these repositories to adapt their content to these devices. Thus, the Web content interaction with different devices will be easier. Screen size, input mechanisms, supported colour sets, known limitations, special features, and so forth are stored in these repositories. In the following lines two of the most popular DDR systems are introduced.

2.4.3.1 WURFL

WURFL [40] is an XML based open source database which contains the characteristics and capabilities of a wide range of devices. These capabilities are classified into several groups. These groups represent a simple way to understand WURFL and its data. Its API is very easy to use and it provides a hierarchy able to infer several capabilities for devices which are still not present in the file.
The following are several capabilities used by Almeida et al. [49] for Imhotep, a framework which aims to ease the development of accessible and adaptable user interfaces:

- *display*: It contains information about the device's screen (e.g., the resolution, number of columns, orientation, and so on).
- *image_format*: Supported image formats.
- *playback*: Supported video codec.
- *streaming*: Available streaming capabilities.
- *sound_format*: Supported audio formats.

WURFL has become the de-facto standard for mobile capabilities. Nevertheless, there are several free and open source alternatives that are growing within the community very fast.

DDR	Advantages	Drawbacks
WURFL	Upgradeable to new versions	Errors in data
	A hierarchy which allows to infer values	Many empty values
	Many capabilities modelled	
	Very easy to configure	
	Powerful API	
OpenDDR	Free to use, even commercially	Limited number of capabilities
	Growing community	Default values for unknown data

Table 2.6: Analysed DDRs comparison [10].

2.4.3.2 OpenDDR

OpenDDR[1] is an open DDR alternative which also provides an API to access DDRs about devices capabilities. It has two main advantages:

- The conviction that the application will work with any W3C DDR API implementation.
- Adopting the W3C standard the Copyright of the interfaces is protected by the W3C against any intellectual property and patent claims.

Nevertheless, the OpenDDR API is complex. It does not provide an architecture approach like WURFL. Thus, it assumes default values for unknown parameters (e.g., *displayWidth* $= 800$ pixels).

2.4.3.3 DDR Solution Comparison

The main problem with these solutions is their inability to provide all the information developers usually need. For instance, in WURFL many significant fields are empty (which means that default values are used) or with error data. Another disadvantage is that, for the interface adaptation domain, sometimes we may need dynamic information about the device. For example, the battery levels, or the available memory, can be crucial pieces of information before making any adaptation process. Table 2.6 depicts each solution's drawbacks and advantages, extracted from our experience with these DDR solutions.

2.4.4 Ontologies

There are other alternatives for modelling devices. Ontologies has the ability to give some meaning to the modelled concepts and the collected information. From the reviewed

[1] https://github.com/OpenDDRdotORG/OpenDDR-Resources

ontologies we remark the Standard Ontologies for Ubiquitous and Pervasive Applications (SOUPA) ontology which, in addition to consider context information, it models many different aspects of static mobile capabilities and characteristics [67]. Another significant ontology is the one presented by Hervás et al. [97] as part of the Pervasive Information Visualization Ontology (PIVon) ontology. The Device Model Ontology describes not only the device's capabilities but its relationships with the service and user ontologies, dependencies and other features.

2.4.5 Device Discussion

Apart from all the mentioned techniques and solutions for considering devices and their capabilities, it is necessary to remark how technology evolves in a way that makes difficult to predict how we will work with smart devices in the near future. A few years ago context was understood as the result of the mixture of every physical sensor deployed in the environment. Temperature, light, noise and so forth. All these context variables have been historically collected through several sensors located strategically. Now these sensors (and more) are also embedded in these smart devices. They have become wearable, and using their capabilities of the devices they area embedded in they have improved their performance. Touchable screens, network capabilities, parallel computing resources and high storage space embedded in small devices have changed the way context and smart devices interact. This situation makes us contemplate how this interaction channel would be in the near future. Smart watches, glasses and other wearable devices are, undoubtedly, changing the way smart environments will sense, behave and react.

And the youngest of the family is moving with authority. Building castles by the sea, he dares the tardy tide to wash them all aside.

Jethro Tull

CHAPTER

3

AdaptUIOnt: An Ontology for Dynamic User Interface Adaptation

The existing literature approaches for user, context and device modelling analysed in Chapter 2 revealed several problems in systems which aim the adaptation of user interfaces, applications or services. To face these issues (regarding a user interface adaptation) a platform supported by two main bases is proposed: AdaptUIOnt, an ontology which describes the most significant concepts within a user interface adaptation domain is proposed. Secondly, a set of rules, which main purpose is to trigger different actions regarding the available knowledge of the current domain represented through the ontology.

The following sections detail the concepts illustrated through Figure 3.1. This figure presents, on the one hand, several crucial concepts represented through the AdaptUIOnt ontology. The ontology and all its characteristic are detailed in Section 3.1. On the other hand, the set of rules which concern these concepts are also depicted. These rules are described in Section 3.2. This first diagram is unfolded during this chapter and finally reviewed in Section 3.3 (specially through Figure 3.9) in order to update the illustration with all the described model and the learned concepts.

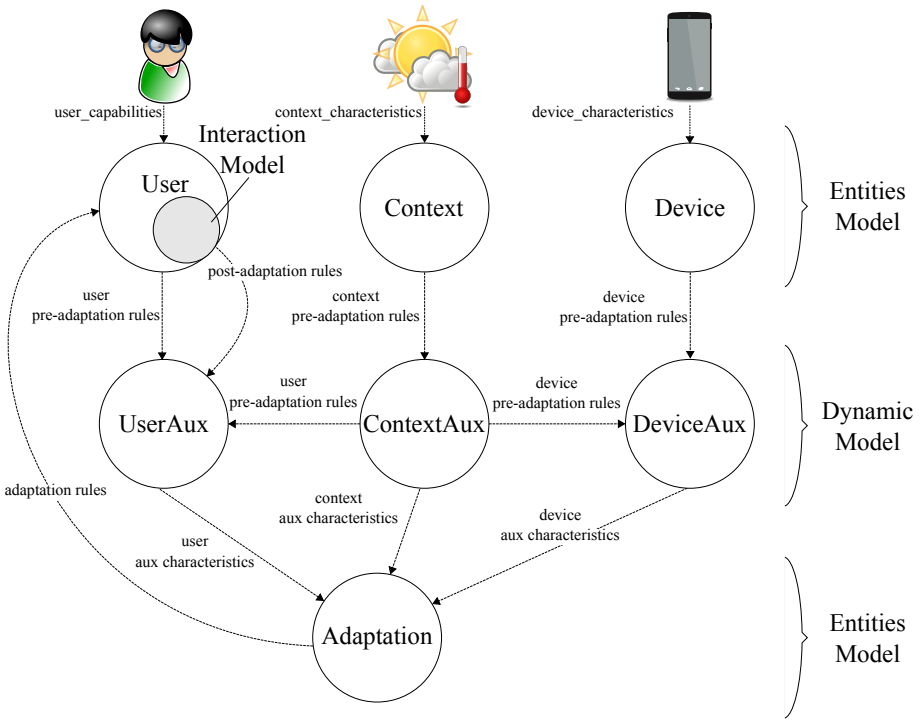

Figure 3.1: Knowledge flow through the adaptation process. The circles represent several main concepts presented in the AdaptUIOnt ontology. The arrows describe the set of rules that affect the related concepts in the circles.

3.1 The AdaptUIOnt Ontology

As said before, the proposed adaptation platform is supported by two pillars: an ontology that models several significant concepts of the domain, and a set of rules that trigger several modifications of these concepts. In this section the AdaptUIOnt ontology, which represents the existing concepts in the user interface adaptation domain, is presented.

The AdaptUIOnt ontology arises from the need of an adaptation model which gathers knowledge of the *user* capabilities, the *context* that surrounds the user, and the mobile *device* the user manipulates. The main goal is to obtain a user interface adaptation personalized exclusively for the characteristics of these three entities at the end of an adaptation process.

Through the following sections the AdaptUIOnt model is unfolded as follows: First, an introduction about the model is performed (see Section 3.1.1). This introduction answers two design questions: *how* the ontology has been designed and *why* several design decisions were made. The answers bring several distinguishing aspects from the existing models found in the literature (see Chapter 2). Next, in Section 3.1.2 an answer to the question *what* is given. The AdaptUIOnt ontology models several primary and secondary groups of entities and knowledge. The primary set of entities are grouped in the *Entities*

Model, explained in Section 3.1.3. The second group are contained in the *Dynamic Entities*, explained in Section 3.1.5. Besides, AdaptUIOnt describes several concepts and supports a set of rules which trigger different actions (see Section 3.2).

3.1.1 Introducing AdaptUIOnt: The *Hows* and the *Whys*

The literature analysis made in Chapter 2 exposes several possibilities regarding the problem of choosing the best technique to model a user interface adaptation process and all the required knowledge around it. As it will be explained later in the conclusions of this chapter, using ontologies brings several benefits to the adaptation platform. In this section *how* and *why* the AdaptUIOnt ontology has been designed is explained.

Many ontology based solutions take users as entities described by their physiological capabilities [86][128][121][125][124][137]. Moreover, facing adaptation or personalization problems, these solutions aim to cover not only capabilities but also disabilities. When AdaptUIOnt was conceived, we realised that modelling physiological abilities was not practical. Although many users may have similar preferences and capabilities, their tastes may differ. Besides, their reactions facing several problems may not be the same. In addition, we realised how difficult it is to model each user physiological characteristics without an expert's knowledge and background in the field. As scientists in the computing domain we lack this kind of physiological knowledge about individuals. Thus, we believe we cannot face modelling user capabilities and also contemplating and analysing their behaviour and responses.

Nevertheless, Casas et al. [63] described a taxonomy where user disabilities are not explicitly included. Instead of that, user preferences are classified under several concepts, as is shown in Figure 3.2. Through this taxonomy is shown how the authors avoid the modelling of physiological capabilities of the user, centring the model in several preferences. More information about the considerations of Casas et al. are given in Section 2.2.2.11.

AdaptUIOnt is built under this idea, extending it and avoiding the modelling of any explicit physiological characteristics. As it will be depicted in this chapter several concepts have been changed in the AdaptUIOnt models. Besides, several remarkable ontologies from other authors are also used to enrich the AdaptUIOnt knowledge. For example, the FOAF [12] ontology has been linked to complete the information of the user. The same happens with the GUMO [93], which also models information about the user. Although not all classes are included, several have been imported in the final version of the AdaptUIOnt model, as they might represent significant concepts for other developers. For example, in the GUMO ontology there is a class which models the user emotional state (*Basic User Dimensions* $\rightarrow$ *Emotional State*) through five basic emotions. In the current AdaptUIOnt version these emotions are not considered for the adaptations. However, we

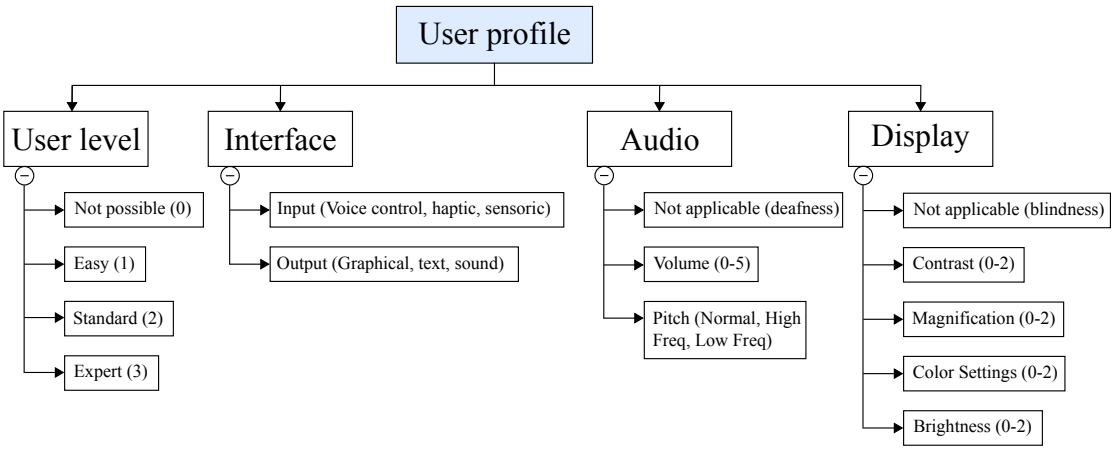

Figure 3.2: User profile taxonomy by Casas et al. [63].

believe that other developers may consider that being under an *anxiety* condition might change or modify the result or the adaptation process. Every ontology that has been used to complete the AdaptUIOnt ontology is shown in Table 3.1.

Other works have also been taken into account for the user model. For example Gregor et al. [86] consider that users (in this case, the elderly) evolve and their capabilities might change not just because of their experience but because of the context influence. For Gregor et al. elderly's capabilities decrease due to their ageing. In this dissertation, this evolution of the user capabilities is considered to be based on the context conditions and experience.

Through the AdaptUIOnt's user model we also aim to allow users to configure the adaptation process through the mentioned perspective, which is, without any required physiological knowledge. During this configuration the user interacts with a module called Capabilities Collector indicating several interaction requirements. This module translates the user indications into a capabilities model. The Capabilities Collector and all its features are detailed in Section 4.1.1.

Regarding the context, several considerations were also discussed during the design process. Context is mainly defined by its physiological conditions. Besides, the GUMO ontology has been used to enrich the model. The context model is extended in two different ways: on the one hand, there is the information collected from sensors and the combination of their measures (environment data); on the other hand, a high-level information category built from the combination of context and external pieces of information.

Devices are also modelled using different CoBrA classes. These classes represent static and dynamic concepts of the device. Device's screen resolution (*DisplayScreenResolution*) is one of the static features of a mobile device. On the contrary, the battery of the device changes with the use. This is represented through the *Battery* class, and is also

Ontology	Class	Description	Imported subclasses
GUMO [93]	*Basic User Dimensions*	It originally models different aspects of the user, as certain abilities, emotional status, and so on.	*Ability And Proficiency*, *Characteristics*, *Contact Information*, *Demographics*, *Motion*, *Role*, *Emotional State*, *Personality*, *Mental State*, *Physiological State*
	Context Information	It represents several concepts related to the environment.	*Location* and *Physical Environment* and *Social Environment*.
FOAF	*Document*		*Image* and *PersonalProfileDocument*.
	SocialInstitution		
	OnlineAccount		*Online Chat Account*, *Online E-commerce Account* and *Online Gaming Account*.
CoBrA [9]	*DeviceMemory*		
	DisplayScreen		
	DisplayScreenResolution		
	MemoryUsageType		

Table 3.1: Imported ontologies to complete several concepts of the AdaptUIOnt ontology: GUMO, FOAF and CoBrA.

used by AdaptUIOnt to represent this concept. AdaptUIOnt remarks the dynamic characteristics of these devices, which are usually not modelled and are vital for the result of any kind of adaptation. Thus, several classes representing dynamic characteristics of mobile devices are added.

3.1.2 Designing the AdaptUIOnt Ontology

The AdaptUIOnt ontology has been designed with two main considerations in mind. First, taking into account that useful and practical (and non physiological) capabilities of the user, context and device need to be represented in the model. This is carried out by reviewing the literature models and making several adaptations, modifications and contributions. For example, regarding the user model, we use FOAF and GUMO to model the user most personal characteristics. Nevertheless, the user model is not based on these ontologies. It is built under several assumptions made by Casas et al. [63] in their user profile taxonomy, with several modifications. Regarding the three entities, the ontology understands each one as a set of characteristics that define them. Thus, there is, for example, a *User* class with a relationship *definedBy* which relates the concept of a user with the characteristics that define him/her (the *UserCharacteristics* class). The same conceptualization is shared by the other two entities (see Figure 3.3). This part of the ontology has been called *Entities Model*, and it is detailed in Section 3.1.3.
Second, and based on the conclusions made by Gregor et al. [86], the proposed ontology aims to be aware of the possible dynamic variations of the environment, which may affect to the three entities. Therefore, we have implemented several auxiliary classes to collect all the temporary knowledge within the environment. These classes, shown in Figure 3.4, are linked to the main classes and complete them through several triggered rules (see Section 3.2). This part of the dynamic knowledge representation of the AdaptUIOnt ontology has been called *Dynamic Model*, and it is explained in Section 3.1.5.
As said before, the proposed adaptation engine allows users to manage the adaptation model through the Capabilities Collector module. This idea allows users to participate during the model personalization and adaptation process. To this end, several classes have an *isStatic* data property. This property enables or disables the adaptation for the corresponding class depending on its value (*true* or *false*). For example, the user might not want the display brightness to be changed. A user profile where the brightness is configured as static with the corresponding boolean value (*adaptui:userdisplayBrightnessIsStatic*) means that during the adaptation process the corresponding rule (see the adaptation rules set in Section 3.2) will evaluate this property and finally avoid any change for the brightness parameter. The set of AdaptUIOnt classes, object and datatype properties are shown through Figure 3.5, Figure 3.6 and Figure 3.7 respectively.

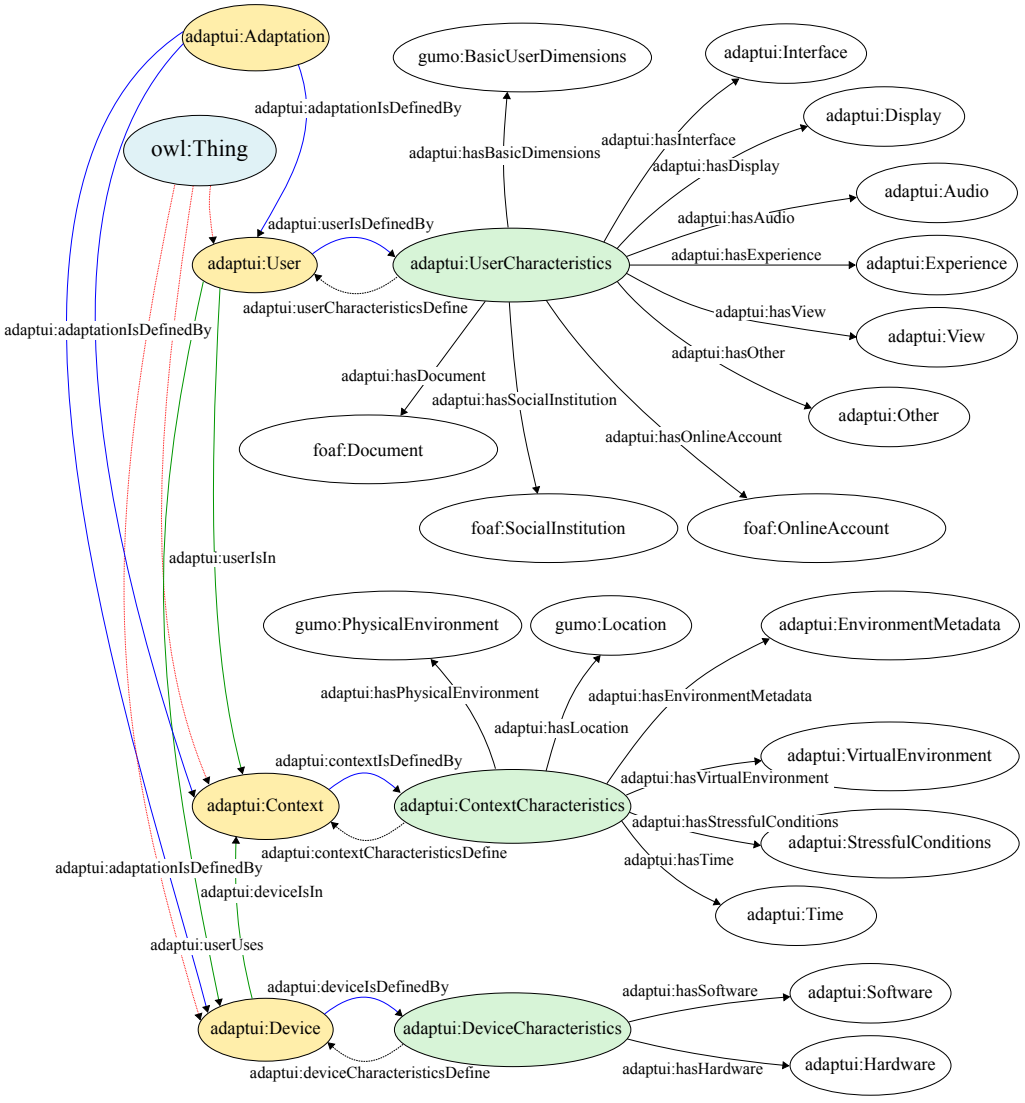

Figure 3.3: *User*, *Context* and *Device* classes (in yellow) and their main object relationships. As is shown, these classes are defined by their corresponding characteristics class (in green).

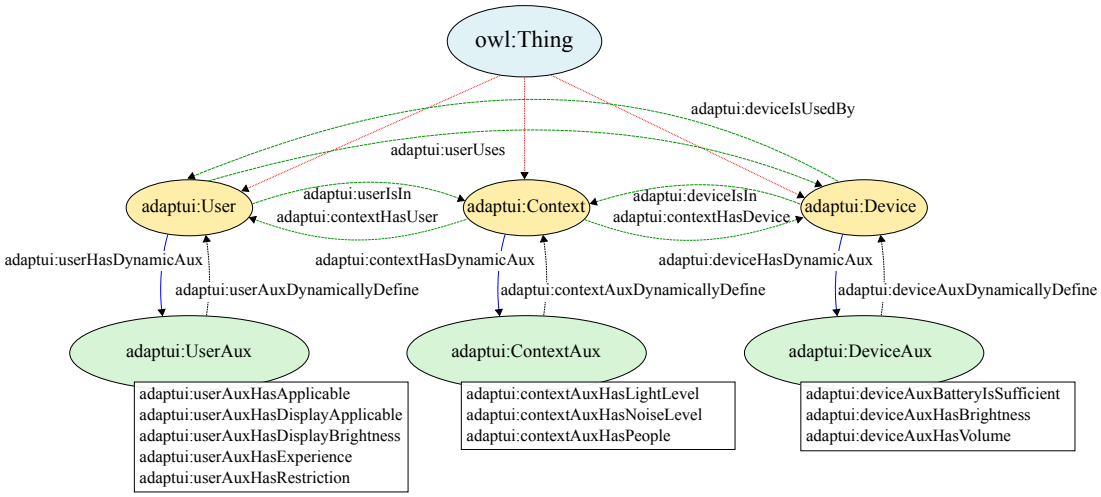

Figure 3.4: *UserAux*, *ContextAux* and *DeviceAux* classes (in yellow) and their main datatype relationships.

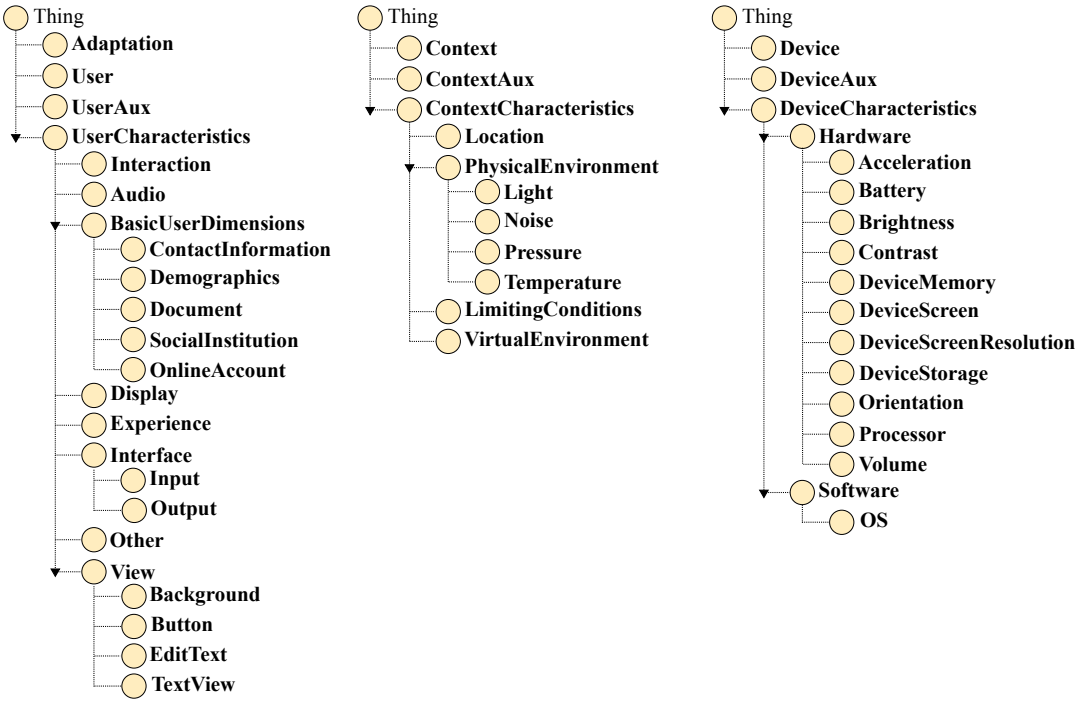

Figure 3.5: *User* (left), *Context* (centre) and *Device* (right) classes of the AdaptUI ontology.

Figure 3.6: AdaptUI object properties.

Figure 3.7: AdaptUI datatype properties, not considering other ontologies. The left group of properties are those related to the *UserCharacteristics* class.

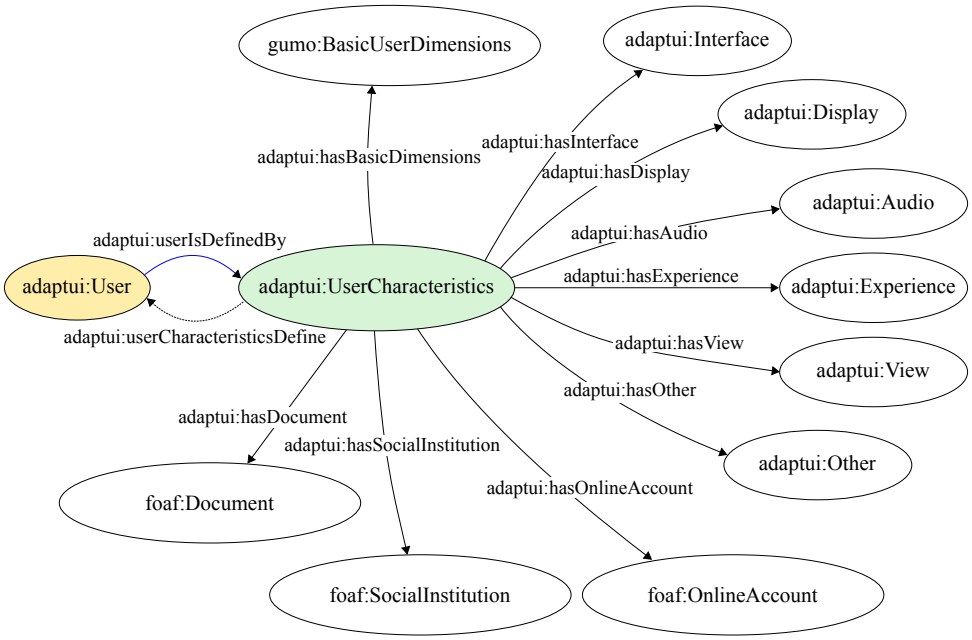

Figure 3.8: The *User* and the *UserCharacteristics* classes.

3.1.3 The *Entities Model*

After reviewing the literature in Chapter 2 we found that, for an appropriate adaptation of a user interface, there are three main entities which must be represented in the domain: the user, the context and the device. These concepts are semantically represented through the *User*, *Context* and *Device* classes in the AdaptUIOnt ontology. These classes are defined by their corresponding characteristics. These characteristics are represented through the *UserCharacteristics*, *ContextCharacteristics* and *DeviceCharacteristics* classes (see Figure 3.8). Together with the *Adaptation* class, these seven classes form the Entities Model, which is a conceptual group within the whole AdaptUIOnt model. Figure 3.3 shows the set of classes which form the Entities Model.

Through the following sections the classes which belong to the Entities Model are detailed. First, the *UserCharacteristics* class is described in Section 3.1.3.1. Next, the *ContextCharacteristics* class is detailed (see Section 3.1.3.2). Third, a description of the *DeviceCharacteristics* class is given in Section 3.1.3.3. Finally, the *Adaptation* class is detailed in Section 3.1.3.4.

3.1.3.1 The *UserCharacteristics* Class

The *UserCharacteristics* class is one of the seven classes included in the conceptual Entities Model. This class includes a series of subclasses which build the knowledge about the user. Several user modelling ontologies are used (i.e., FOAF and GUMO). However, the strength of this class lies not on the representable knowledge through these classes, but

in the way the knowledge about the user capabilities (and disabilities) is represented. As explained in the introduction of this chapter, modelling physiological capabilities of users is troublesome and not practical. Several authors have tried to model user physiological characteristics [86][128][121][125][124][137]. Although their systems may behave properly, there is still the issue of facing a coherent user modelling process including user's capabilities. Not only because we lack physiological background in the area, but also because different users may behave in different ways even if they suffer from the same disability.
To avoid this problem and to provide an ontology able to model user's capabilities, AdaptUIOnt's user's capabilities and disabilities knowledge is represented through the following classes (see the details of each class given in Table 3.2):

- *Interface*: This category gathers input and output information about the user interaction preferences. Instead of modelling physiological capabilities regarding user's interaction abilities (e.g., sight capabilities), the model focuses in taking into account the user needs. For example, a user with a sight problem will not have to model a specific sight disability. This would require to consider several sight conditions and, for each one, different measures, ranges and classifications. Instead of that, there is the possibility for the user to model the Input/Output as voice and audio based interaction. This means that, by specifying these parameters, the system will understand that the user might suffer from a hearing loss condition or a sight disability.

- *Audio* and *Display*: These two classes model aspects about the use of the audio commands and volume controls, as well as different presentation parameters for the display, as brightness, contrast, orientation and magnification. Thus, sight disabilities or, simply, sight problems due to context conditions (e.g., direct sunlight on the device screen) and hearing problems are faced by allowing the user to establish the default (or minimum) brightness, magnification, contrast, pitch, frequency and volume of the adaptation.

- *Experience*: User experience with technology might be useful when executing adaptation rules. For examples, inexperienced users may need more instructions to run the profile (model) configuration process to add the corresponding knowledge about their capabilities.

- *Other*: Several extra parameters (for example, the property *has_TTS*, which means that the user needs a tool which reads the text from the display). It also includes language and vibration related aspects.

- *View*: User interfaces are composed of different views (GUI elements) that are combined for representing different applications and services. For each of these views (e.g., a button) the model considers several properties that the user can configure (i.e., size and colours). For example, users with colour blindness can change the colours of the components before any adaptation to specify the set of colours they are able to distinguish, or adjust a minimum size for a button so the user is able to interact with it. Thus, in the future adaptations, the adaptation rules will be aware of this user particular need.

- *Basic User Dimensions*: Extracted from the GUMO ontology, this class models information about the user contact information. See Table 3.1 in order to see the whole set of classes included under this class.

- *Document*, *SocialInstitution* and *OnlineAccount*: Extracted from FOAF, these classes complete several personal and social characteristics of the user.

Subclass	**Property name**	**Description**
Interface	*interfaceHasInput*	This property models the possible input methods for the user with the following possibilities: *gestures*, *haptic*, *voice_control*, *sensory*, *only_haptic*, *only_voice_control*.
	interfaceHasLanguage	Describes to define the desired language.
	interfaceHasOutput	This property models the possible output methods for the user with the following possibilities: *standard* (images, text, video and sound), *only_audio*, *only_text*.
Audio	*audioHasApplicable*	Describes if audio interaction is applicable for the user. If *false*, we infer that the user might suffer from some hearing problems.
	audioHasFrequency	Represents the preferred audio frequency.
	audioHasPitch	Describes the preferred pitch.
	audioHasVolume	Represents value which points out the desired audio volume.
Display	*displayHasApplicable*	Describes the capability of the user to interact with the display. If *false* we might understand that the user suffers from a sight disability.
	displayHasBrightness	Represents value representing the brightness value.
	displayHasContrast	Describes indicating the contrast value.
	displayHasMagnification	Represents value which represents the magnification degree needed for the user to properly see the display.
	displayHasOrientation	Describes two options: *landscape* or *portrait*. This parameter indicates the preferred display orientation for the user.
Experience	*userHasExperience*	A set of possible values to describe user's experience qith technology: *standard*, *advanced*, *not_applicable*.

View	*viewHasColor*	Describes the colour for the view background.
	viewHasWidth	Represents value to determine the width of the view.
	viewHasHeight	Describes the height of the view.
	viewHasTextColor	Represents value representing the colour for the text.
	viewHasTextSize	Represents value representing the text size.

Table 3.2: UserCharacteristics data properties

3.1.3.2 The *ContextCharacteristics* Class

Usually, context is modelled considering that the data collected by sensors is enough to characterize it. In AdaptUIOnt this kind of context is also modelled, since sensor information is significantly relevant for being aware of the context environment conditions. Thus, the information collected by sensors is represented under the *PhysicalEnvironment* and *Location* classes, which are classes from the GUMO ontology. Nevertheless, the AdaptUIOnt ontology does not only model sensor related knowledge. There are three extra classes which aim modelling high-level information about the environment: *LimitingConditions*, *VirtualEnvironment* and *EnvironmentMetadata*. In the following lines a description of each class is presented:

- *PhysicalEnvironment*: Environment information collected from sensors (e.g., absolute location, available resources, light and noise conditions).
- *Time* and *Location*: Both classes are used to characterize each user and device in a temporal and location context. Both entities are linked to these context variables through the *userIsIn* and *deviceIsIn* object properties (see Figure 3.6).
- *LimitingConditions*: Instead of modelling a set of particular activities we consider several context situations that might impede the interaction with the user. These situations are modelled as different groups regarding the capabilities that they might affect (see Section 3.1.4).
- *VirtualEnvironment*: Combining the knowledge of the categories above, it is possible to extract high-level information. For example, sensing that there is a light at the office, it is possible to infer that there are people working. Thus, we avoid the use of other sensors to indicate this activity.
- *EnvironmentMetadata*: Environment knowledge is associated to sensors. A sensor can provide information about the temperature (23 ^{o}C). But this information by itself is poor in a context-aware system. Environment metadata can describe and

Subclass	Property name	Description
Location	*hasRelativeLocation*	Represents relative locations.
	hasAbsoluteLocation	Describes *longitude* and *latitude*.
Light	*contextHasLight*	Describes the amount of luxes (lx)[112] in the environment.
Noise	*contextHasNoise*	Represents the amount of decibels (dBs) in the environment.
Time	*hasTimeValue*	The current time.
LimitingConditions	*hasMovementRestriction*	Includes several abstract activities.

Table 3.3: ContextCharacteristics data properties

enrich this knowledge, providing time and location data. For example, "the current temperature at 12:00 AM in Bilbao is 13 ^{o}C". This information contains small pieces of low level information which, combined, form a high level information about the current environment situation.

Table 3.3 shows the main datatype properties of the *ContextCharacteristics* class.

3.1.3.3 The *DeviceCharacteristics* Class

The device model is also built over several useful classes from other ontologies. The most important characteristic of this class consists in modelling dynamic information about the device regarding the adaptation. For example, low battery levels might be considered risky by the adaptation engine, as there is the possibility that the device turns off during the process. Device capabilities are categorized as follows:

- *Software*, which encompasses different software aspects of the device (i.e., the Operative System (OS) platform).
- *Hardware*, designed to model information about the current status of different capabilities (i.e., available battery and memory).

Table 3.4 shows the main datatype properties of the *DeviceCharacteristics* class.

3.1.3.4 The *Adaptation* Class

The last class included in the Entities Model is the *Adaptation* class. This class represents the final stage of the whole adaptation process. This means that, after the adaptation process the results will be represented through an individual (or instance) of this class. Therefore, the proposed adaptation platform will semantically request the corresponding adaptation for the user interface to this class. As can be seen in Figure 3.3, the *Adaptation* class is linked to the other classes of the Entities Model (*User*, *Context* and *User*)

Subclass	Property name	Description
Brightness	*deviceHasBrightness*	Describes the current device's brightness.
Contrast	*deviceHasContrast*	Represents the current device's contrast.
Volume	*deviceHasVolume*	Describes the current device's volume.
Processor	*deviceHasHWCores*	Models the number of the device's cores.
OS	*deviceOSHasVersion*	Indicates the current OS version.
Acceleration	*deviceHasAcceleration*	Represents the current X, Y, Z acceleration.
DeviceScreen, Processor, Orientation, DeviceMemory, Battery, DeviceScreenResolution, DeviceStorage	*hasFactoryValue*	Maximum values for the elements under the Sub-class column.

Table 3.4: DeviceCharacteristics data properties.

through the *adaptationIsDefinedBy* object property. Thus, future semantic requests are allowed searching for a specific user capability, context situation or device characteristics. Table 3.5 shows the datatype properties of the *Adaptation* class.

3.1.4 Incoherence and Activities

During the design of the AdaptUIOnt ontology several considerations were studied in order to fully understand how the three main entities (user, context and device) interact with each other. Modelling these entities and being aware of their interactions among them lead the platform to deduce the best adaptation for the user in each case. However, there are several situations where the adaptation process is not so obvious. These situations are defined by the activities that are being carried out within the environment.
Persad et al. [125][124] consider that activities need to be taken into account. They describe Human Factors and Ergonomic theory as four components: the user, the product, the context and the activities over time that constitutes the interaction. Hong et al. [99] classify context conflicts into several categories:

- Sensing conflict: Not matching results from several physical data sources.
- Service resource conflict: The lack of resources in a service offering process may provoke several conflicts.

Datatype property	Description
adaptationBrightnessHasValue	Describes the final brightness value to be configured in the device.
adaptationVolumeHasValue	Represents the final volume value to be configured in the device.
adaptationButtonHasSize	Describes the final size value for a button.
adaptationButtonHasBackgroundColor	Represents the final background colour for a button.
adaptationButtonHasTextSize	Describes the final text size for a button.
adaptationButtonHasTextColor	Represents the final text colour for a button.
adaptationEditTextHasSize	Describes the final size value for a edit text.
adaptationEditTextHasBackgroundColor	Represents the final background colour for a edit text.
adaptationEditTextHasTextSize	Describes the final text size for a edit text.
adaptationEditTextHasTextColor	Represents the final text colour for a edit text.
adaptationTextViewHasSize	Describes the final size value for a text view.
adaptationTextViewHasBackgroundColor	Represents the final background colour for a text view.
adaptationTextViewHasTextSize	Represents the final text size for a text view.
adaptationTextViewHasTextColor	Describes the final text colour for a text view.

Table 3.5: *Adaptation* class datatype properties.

- User preference conflict: Users whose profiles or preferences are different but having the same context situation may also result in context conflict.

During the designing process of the adaptation platform, we asked ourselves if it would be enough to consider just the user and his/her capabilities within the current context. Thus, several hypothetical scenarios were studied:

- A user suffers from a visual impairment. This disability obstructs the user from seeing the application content properly. Then the adaptation engine will intercede to facilitate another interaction channel for the user, e.g., by voice recognition and control. The problem is that there are situations in which a common adaptation from the system will not be appropriate. For example, if the user is in a place where silence is essential (i.e., a library, a hospital or in an exam), an audio interaction based communication could not be appropriate.

- A user who sees perfectly well interacts with the application's default user interface. If we avoid a situation in which the user is driving a visual/touch based adaptation could put the user in risk, as he/she would need to look at the display and use his/her hands.

- Another user is at home, and he/she does not suffer from any severe disability. At 01:00 pm he/she starts cooking. The application requests user attention for several tasks. This situation might be risky if the adaptation requests the user attention while he/she has, for example, oil in the pot, or he/she is manipulating knives.

These examples present several situations where users are involved in tasks that contradict the current context and user capabilities. An *adaptation incoherence* is defined by several environment parameters that induce the platform to perform a certain adaptation for the current conditions. However, the result of this adaptation, although it can be aligned with the context characteristics, can be incoherent.
Therefore, we need something more to characterize the current situation that involves these three entities: activities. Activities help to understand the current user, context and device situation. In other words, it enriches the environment information.
Requiring the use of the hands, being at a certain location (like a library), or demanding the user attention are several examples of situations which may represent some risks that have to be taken into account when dealing with a context modelling problem. For example, driving or cooking restricts user capabilities momentarily. Hence, there are activities that impede the user. As is difficult to model every possible activity that the user performs, AdaptUIOnt includes a class (*LimitingConditions*) that models abstract groups of activities:

- Activities that limit the use of the hands.
- Activities that limit the use of the voice.
- Activities that limit the user sight capability.
- Activities that limit the user attention.
- Activities that limit the user movement.
- Combinations of these activities.

3.1.5 The *Dynamic Model*

The proposed solution addresses several issues of several modelling approaches found in the literature. Following the perspective of Fischer [82], the adaptation platform and the presented ontology consider that the modelled entities are not static. They change through time due to their interaction. To express this concept of *dynamic model* several auxiliary classes have been included:

- *UserAux*: This class is helpful when a certain context situation impedes a user capability. Updated by the pre-adaptation rules (explained in Section 3.2.1) this class is used as intermediary between the *Adaptation* class and the *ContextAux* class, where the classification of the context collected information is modelled. As is shown in Table 3.6, it models several user dynamic capabilities.

- *ContextAux*: Context is considered in a different way. As context information comes from sensors, the different incoming data needs to be translated to a more descriptive language. Thus, if a value of 35,000 lx is collected (as a brightness value of the *ContextCharacteristics* class) the *contextAuxHasLightLevel* data property is modified with the *direct_sunlight* value (see Table 3.6). Therefore, it is possible to work with more meaningful data. Table 3.7 and Table 3.8 show the different classifications for light and noise levels.

- *DeviceAux*: Following the same approach, this class deals with the dynamic capabilities of the device.

Working with sensor data opens new fronts regarding the AdaptUI platform. For example, fuzzy reasoning would help to refine the collected information. This is discussed in the Future Work section (see Section 6.4).

Class	**Property name**	**Description**
UserAux	*userAuxDIsplayHasApplicable*	Models the applicability of display adaptations for the user. There are two different scenarios: in the first one, the user might suffer from a disability that makes impossible for him/her to interact with a display. In this case *Display* is not applicable for this user; on the contrary, the user can specify that he/she does not want the display to be adapted. In any case the value of this property will determine if the rules need to consider *Display*.
	userAuxAudioHasApplicable	Similar to the previous property, this one has the same effect for audio adaptations.
	userAuxHasDisplayBrightness	Depending on the value of the *UserCharacteristics*: *userDisplayBrightnessIsStatic* property, the corresponding rule will update this value indicating if the brightness should be considered for the adaptation process.
	userAuxHasExperience	Represents user's experience with technology: *easy*, *expert*, *not_possible*, *standard*.
	userAuxHasRestriction	User's activities are considered for adaptation.

		Hence, a Boolean value is modelled in this property to indicate so.
ContextAux	*contextAuxHasLightLevel*	Represents several light classifications: *clear_night*, *dark_overcast*, *daylight*, *direct_sunlight*, *living_room*, *moonless_clear*, *moonless_overcast*, *office_hallway*, *office_lightning*, *overcast_day*, *sunrise*, *twilight*.
	contextAuxHasNoiseLevel	Represents several noise classifications: *absolute_threshold_of_hearing*, *breathing*, *building_work*, *conversation*, *factory*, *gig*, *jackhammer*, *leaves_murmuring*, *library*, *office*, *traffic*, *train*, *truck*, *whispering*.
DeviceAux	*deviceAuxBatteryIsSufficient*	Based on Table 3.9, indicates whether the adaptation should be performed considering the current battery level.
	deviceAuxHasBrightness	Describes the current brightness level of the device's screen.

Table 3.6: Auxiliary classes' data properties.

Brightness (measured) in lx	**Surfaces illuminated by**	**Ontology value**
0.0001	Moonless, overcast night sky (starlight)	*moonless_overcast*
0.002	Moonless clear night sky with air-glow	*moonless_clear*
0.27-1.0	Full moon on a clear night	*full_moon*
1.0-3.4	Dark limit of civil twilight under a clear sky	*twilight*
3.4-50	Family living room lights	*living_room*
50-80	Office building hallway/toilet lighting	*office_hallway*
80-100	Very dark overcast day	*dark_overcast*
320-500	Office lighting	*office_lightning*
500-1,000	Overcast day; typical TV studio lighting	*overcast*
1,000-25,000	Full daylight (not direct sun)	*daylight*
25,000-130,000	Direct sunlight (latter figure is above atmosphere)	*direct_sunlight*

Table 3.7: Luminance provided under various conditions [21].

In order to describe the whole detailed model and how the main classes are related to each other, Figure 3.1 shows how the knowledge flows through the AdaptUI platform. First, the three main classes of the *Entities Model* are populated with the information about the user, the context and the device. Then, the pre-adaptation rules are triggered and updates classifies and updates the collected knowledge into intermediate knowledge. This knowledge is represented in the *UserCharacteristics*, *ContextCharacteristics* and *DeviceCharacteristics* classes. Finally, the *Adaptation* class requests the processed knowledge and through

the adaptation rules the final adaptation is sent to the user. Additionally, within the user model the interaction model collects information about the interaction, updating the *UserAux* class if needed. This brings the execution of the rules again, which means that a new result is generated.

Ontology value	dB
absolute_threshold_of_hearing	0
breathing	10
leaves_murmuring	20
library	40
office	50
conversation	60
traffic	70
factory	80
truck	90
train	100
construction	110
rock_gig	120
jackhammer	130

Table 3.8: Most common sound intensity levels modelled by default in AdaptUI.

Battery (%)	Ontology value
$x \leq 15$	*not_sufficient*
$15 < x \leq 50$	*sufficient*
$50 < x \leq 100$	*optimal*

Table 3.9: Battery percentage and ontology values for AdaptUI.

3.1.6 Conclusions

During the previous sections the AdaptUIOnt ontology, which forms part of the two main bases of the adaptation platform (detailed in Chapter 4), has been described. The most significant characteristic of the ontology is that allows the representation of the knowledge about what we consider the three main entities in a user interface adaptation domain: the user, the context, and the device.

The first trouble when designing the ontology appeared when the user capabilities modelling was needed. As we lack the required physiological knowledge in this area, we believe that using physiological information of the user capabilities would not be practical. Consequently, an abstraction of the conceptualization of the model was performed. Instead of considering the physiological factors that allow a user to perform several interactions, the AdaptUIOnt model centres its focus in the needs of the user to carry out these

interactions. This is significant, as it allows to avoid modelling these capabilities explicitly. Besides, the presented ontologies use several extendedly used ontologies to enrich the information about the three entities. Table 3.1 details the most important classes imported in AdaptUIOnt.

The ontology has been conceptually divided into two different parts. The first one gathers the main classes, those which directly represent knowledge about the main entities. In this case, this part of the ontology has been called the Entities Model, and its composed by the *User*, *Context*, *Device*, *UserCharacteristics*, *ContextCharacteristics*, *DeviceCharacteristics* and *Adaptation*. Each main entity is defined by a class which models its characteristics. Thus, the *User* class is directly related to the *UserCharacteristics* class through a *isDefinedBy* object property. The same procedure is followed by the other two main entities. The *Adaptation* class, however, just represents the final stage of the whole adaptation process. Therefore, after the corresponding reasoning (detailed in the following sections) the adaptation platform will semantically requests the adaptation information represented by it. The second part of the ontology encompasses the classes used for adding a dynamic characteristic to the model. The *UserAux*, *ContextAux* and *DeviceAux* are part of this Dynamic Model. Their purpose is to adapt the collected pieces of information to a high level information to be used by the platform. Hence, developers are able to manage richer data for the corresponding adaptations.

3.2 The AdaptUI Rules Set

As said in the introduction of this chapter, the developed adaptation platform is supported by two bases. The first one, the AdaptUIOnt ontology, described in the previous sections. The second one, a set of rules that uses the knowledge represented by the ontology and triggers several actions aiming the user interface adaptation. In this dissertation the adaptation process is understood as a three step procedure:

- First, the knowledge of the main entities needs to be collected. Once it is gathered, a normalization process is performed. This process classifies the collected information so it can be easily read. For example, a noise sensor might sense a 100 dBs sound. Initially, the platform might not understand this value. Thus, using several ranges explained during this chapter (see Table 3.7, Table 3.8 and Table 3.9) a high level classification is carried out.

- Secondly, and taking into account the transformed knowledge in the previous step, several actions are performed to adapt the user interface to the current situation (represented through the ontology in the previous stage).

- Finally, a refinement of the adapted user interface is available to improve the final results.

This conceptual division of the adaptation process in the adaptation platform results in a three set of different rules, which are detailed below.

3.2.1 The Pre-adaptation Rules

This group of rules are designed to check those values which come from each entity of the Entities Model in order to translate them to the auxiliary models. For example, a 35,000 lx context brightness indicates the amount of lx of the current environment. This value is then classified to a more verbose subgroup (in this case, *direct_sunlight*); a 10 device battery value represents the percentage of the remaining battery. This is translated as a *non_sufficient* battery level in the auxiliary device class. Thus, the corresponding adaptation rule checks these new parameters and decides whether the adaptation should follow one path or the other. In this dissertation we consider that 10% of the total battery levels for a device might not be enough to run an adaptation process. Of course this is not accurate. The user might be able to use a nearby power adaptor. Besides, 10% battery level might not be enough if the user aims to browse the web, make a video call, and so forth, but it might be sufficient carrying out one adaptation. This means that these values are established by the developer regarding the just the adaptation, but they directly depend on the user context. This set of rules directly affects the *UserAux*, *ContextAux* and *DeviceAux* classes, being the main reason for these classes to belong to the Dynamic Model introduced in Section 3.1.5.
Table 3.10 describe the default pre-adaptation rules included in AdaptUIOnt.
Equation 3.1 shows an example of the *checkNoiseLevelTraffic* pre-adaptation rule.

$$\begin{aligned} &Context(?context)\ \&\ Noise(?noise)\ \&\ ContextAux(?caux)\ \&\\ &contextHasNoise(?noise, ?value)\ \&\ lessThanOrEqual(?value, 70)\ \&\\ &greaterThan(?value, 60)\\ &\Rightarrow\\ &contextAuxHasNoiseLevel(?caux, \text{“}traffic\text{”}) \end{aligned} \tag{3.1}$$

3.2.2 The Adaptation Rules

Depending on the checked rules and the auxiliary classes' status, different rules are triggered. These rules result in different values for the *Adaptation* class, which is the class queried in the device platform for bringing the final adaptation to the device. Table 3.11 details the default adaptation rules in the AdaptUIOnt ontology.

Pre-adaptation rule	**Involved ontologies and classes**	**Description**
checkBatteryLevelIsSufficient	*soupa:Battery* *adaptui:Device* *adaptui:DeviceAux*	Considering Table 3.9 this rule evaluates if the current battery level is enough to perform any adaptation.
checkLightLevelDarkOvercast *checkLightLevelTwilight* *checkLightLevelSunrise* *checkLightLevelOvercast* *checkLightLevelOfficeLightning* *checkLightLevelOfficeHallway* *checkLightLevelMoonlessClear* *checkLightLevelDayLight* *checkLightLevelDirectSunlight* *checkLightLevelFullMoon* *checkLightLevelLivingRoom*	*adaptui:Context* *adaptui:ContextAux* *gumo:Light*	These rules evaluate the context light input through sensors by using the classification shown in Table 3.7. The result of this evaluation would be a verbose concept (e.g., *sunrise*) which is stored in the ontology using the *contextAuxHasLightLevel* datatype property.
checkNoiseLevelBreathing *checkNoiseLevelTruck* *checkNoiseLevelTrain* *checkNoiseLevelTraffic* *checkNoiseLevelOffice* *checkNoiseLevelLibrary* *checkNoiseLevelLeavesMurmuring* *checkNoiseLevelBreathing* *checkNoiseLevelBuildingWork* *checkNoiseLevelConversation* *checkNoiseLevelFactory* *checkNoiseLevelGig* *checkNoiseLevelJackhammer*	*adaptui:Context* *adaptui:ContextAux* *gumo:Noise*	These rules evaluate the context noise input through sensors by using the classification shown in Table 3.8. The result of this evaluation would be a verbose concept (e.g., *sunrise*) which is stored in the ontology using the *contextAuxHasNoiseLevel* datatype property.
checkUserHasAttentionRestriction *checkUserHasNoRestriction* *checkUserHasSightRestriction* *checkUserHasMovementRestriction* *checkUserHasHandsRestriction* *checkUserHasHearingRestriction*	*adaptui:Activity* *adaptui:UserAux*	If the current activity impedes the user, these rules would store the corresponding boolean value in the *userAuxHasRestriction* datatype property.

Table 3.10: The AdaptUIOnt pre-adaptation rules.

Adaptation rule	**Involved ontologies and classes**	**Description**
adaptBrightness and *adaptVolume*	*adaptui:Adaptation* *adaptui:DeviceAux* *adaptui:UserAux* *adaptui:Brightness* *adaptui:Volume* *adaptui:Display*	These rules consider the values stored in the *UserAux* class to adapt the brightness and volume accordingly.
adaptButtonSize *adaptButtonColor* *adaptButtonBackgroundColor* *adaptButtonTextColor*	*adaptui:Adaptation* *adaptui:DeviceAux* *adaptui:UserAux* *adaptui:Button*	These rules take into account the current button configuration and the sensed context disabilities.
adaptEditTextSize *adaptEditTextBackgroundColor* *adaptEditTextTextSize* *adaptEditTextTextColor*	*adaptui:Adaptation* *adaptui:DeviceAux* *adaptui:UserAux* *adaptui:EditText*	These rules take into account the current edit text configuration and the sensed context disabilities.
adaptTextViewtSize *adaptTextViewtBackgroundColor* *adaptTextViewtTextSize* *adaptTextViewtTextColor*	*adaptui:Adaptation* *adaptui:DeviceAux* *adaptui:UserAux* *adaptui:TextView*	These rules take into account the current text view configuration and the sensed context disabilities.

Table 3.11: The AdaptUIOnt adaptation rules.

Usability rule	Involved ontologies and classes	Description
checkTaskEffectiveness	*adaptui:UserAux* *adaptui:Effectiveness* *adaptui:Polisher*	It measures the proportion of goals of the task achieved correctly.
checkTaskCompletion		It measures the proportion of the task that is completed.
checkErrorFrequency		It measures the frequency of errors.
checkTaskTime	*adaptui:UserAux* *adaptui:Productivity*	It measures the required time to complete the current task.
checkTaskEfficiency	*adaptui:Polisher*	It measures the efficiency of the user.
checkEconomicProductivity		It measures the cost-effectiveness of the user.
checkProductiveProportion		It measures the proportion of the time the user is performing productive actions.
checkRelativeUserEfficiency		It compares the efficiency of the user compared to an expert.

Table 3.12: The AdaptUIOnt usability rules. The metrics mentioned in this table are detailed in Table 4.3 and Table 4.4.

Equation 3.2 shows an example of the *adaptVolume* adaptation rule.

$$\begin{aligned} &Adaptation(?adaptation)\ \&\ DeviceAux(?device)\ \&\ ContextAux(?context)\ \&\\ &deviceAuxBatteryIsSufficient(?device, ?battery)\ \&\ equal(?battery, true)\ \&\\ &contextAuxHasNoise(?context, ?noise)\ \&\ equal(?noise, \text{“}traffic\text{”}) \\ &\Rightarrow \\ &adaptationVolumeHasValue(?adaptation, 7) \end{aligned} \tag{3.2}$$

3.2.3 The Usability Rules

In order to check the usability satisfaction of the user with the provided adapted user interface, a usability rules set is provided. By checking several usability metrics (detailed in Section 4.2.2.1) these rules determine if the interaction with the adapted user interface might be considered enough. Table 3.12 introduced the usability rules included in AdaptUIOnt.

Adaptation rule	Involved ontologies and classes	Description
incrementBrightness	*adaptui:UserAux*	It increments the brightness of
	adaptui:Polisher	the device in 1F.
decrementBrightness	*adaptui:Adaptation*	It decrements the brightness of
		the device in 1F.
incrementVolume		It increments the volume of
		the device in 1 unit.
decrementVolume		It decrements the volume of
		the device in 1 unit.
incrementButtonSize	*adaptui:UserAux*	It increments the size of the
	adaptui:Polisher	button adding 10 dpis.
decrementButtonSize	*adaptui:Adaptation*	It decrements the size of the
		button in 10 dpis.
darkenButtonBackgroundColor		It darkens the colour.
lightenButtonBackgroundColor		It lightens the colour.
darkenButtonTextColor		It darkens the text colour.
lightenButtonTextColor		It lightens the text colour.

Table 3.13: The AdaptUIOnt post-adaptation rules. For edit texts and text views the same rules that are applied for the buttons are provided.

Equation 4.1 shows an example of the *checkRelativeEfficiency* adaptation rule.

$$\begin{aligned} UserAux(?user)\ \&\ Productivity(?productivity)\ \&\ Polisher(?polisher)\ \&\\ userAuxHasProductivityMetrics(?user, ?productivity)\ \&\\ hasRelativeEfficiency(?productivity, ?efficiency)\ \&\\ lessThanOrEqual(?efficiency, 0.5)\ \&\\ \Rightarrow\\ launchPolisherRules(?polisher, true) \end{aligned} \tag{3.3}$$

3.2.4 The Post-adaptation Rules

Once the user interface is adapted, a concrete architecture module (see Section 4.2.2) monitors the user activity. Hence, through a series of usability metrics the adaptation is considered satisfactory by the adaptation platform. If it detects that the usability level is inadequate, these rules are triggered, changing the user model.

Equation 3.4 shows an example of the *incrementButtonSize* post-adaptation rule.

$$\begin{aligned} Polisher(?polisher)\ \&\ launchPolisherRules(?polisher, true)\ \&\\ UserAux(?user)\ \&\ userAuxHasEffectivenessMetrics(?user, ?effectiveness)\ \&\\ effectivenessMetricHasErrorFrequency(?effectiveness, ?freq?)\ \&\\ greaterThan(?freq, 0.5)\ \&\ Adaptation(?adaptation)\ \&\\ adaptationHasButtonSize(?adaptation, ?size) \Rightarrow\\ adaptationButtonHasSize(?adaptation, ?size + 10) \end{aligned} \tag{3.4}$$

The given set of rules by the adaptation platform have been included as they are understood as vital for a precise adaptation. Nevertheless, and as it will be explained in Chapter 4, these rules are not final. This means that they are modifiable. To this end, a series of tools for developers that allow them to change the knowledge managed by the platform are provided.

3.2.5 Conclusions

In this second part of the chapter the second main basis of the adaptation platform has been presented: the set of rules. Divided into three different groups (considering that the adaptation process is understood as a three step procedure) these rules aim to finally generate a user interface adaptation under the following steps:

- Firstly, a classification and normalization of the collected knowledge of the main entities is needed. This might be low level knowledge. Thus, a higher level information is obtained during this process. To reach this goal, the pre-adaptation rules set is needed. These rules affect the *UserAux*, *ContextAux* and *DeviceAux* classes.
- Next, once the knowledge has been normalized, the adaptation rules set are triggered, generating the corresponding changes in the *Adaptation* class in the AdaptUIOnt ontology.
- Finally, as the adaptation might be improvable, several rules are executed regarding a refinement of the last adapted user interface. This is detailed in Section 4.2.2.

3.3 AdaptUIOnt Conclusions

Inspired by the existing literature approaches for user, context and device modelling in this chapter a model supported by two main bases has been described. On the one hand, we have presented AdaptUIOnt, an ontology which deals with the main problems analysed in Chapter 2. On the other hand, a set of adaptation rules (divided into three groups: pre-adaptation, adaptation and post-adaptation rules) have been also introduced. These two bases are illustrated through Figure 3.9.

The decision of using ontologies is based on the extracted conclusions from the analysis of the literature. Specially, the analysis by Strang and Linnhoff-Popien [140] is significantly remarked in this dissertation. In this work the authors compare different context modelling techniques to finally conclude that ontologies represent the most promising asset with respect to the other analysed techniques (i.e., key-value models, markup scheme models, graphical models, object oriented models and logic based models). As the authors state, ontologies are strong regarding distributed composition, partial validation, content validation and facing incompleteness and the quality of the information.
Besides, after studying the benefits extracted by Strang and Linnhoff-Popien conclusions and several significant works in the literature several more benefits regarding the AdaptUIOnt model are detailed below:

- It makes the solution more extendible. Knowledge is easily represented through semantic models and the models themselves are also easily reusable. A context model built by others can be not only use, but for example, extended with ambiguity and incompleteness support. In fact, the AdaptUIOnt has been built over several contrasted models found in the literature (see Section 3.1). These models are highly spread and tested and due to the nature of ontologies allow us to extend them with our designs.

- Reasoning is allowed over the knowledge represented in the ontology. Ontologies represent knowledge, and one of the benefits of using domain concepts is the ability to learn, classify and infer new knowledge from the existing one. In the developer platform there is an architecture module which deals with reasoning using the knowledge stored in the AdaptUIOnt ontology (see Section 4.1.2.2).

However, the decision of using ontologies as a modelling technique did not only bring benefits. The first drawback, and probably the most significant when dealing with semantics in mobile devices, is the lack of efficiency. Examining several approaches along the literature we encountered that, usually, the reasoning tasks are delegated to an external infrastructure. Until the *smartphones boom* (see Figure 3.10) it was quite difficult to find mobile devices with the appropriate hardware specifications for managing heavy processes. Thus, the analysed literature approaches usually delegate this processing tasks to external services (when talking about UI adaptation in mobile devices). Then, these services send back the corresponding results after performing the corresponding actions.
This drawback is significant, since the platform aims to not only provide an ontology model for the adaptation process, but also a series of tools to allow developers to adapt the ontology and the knowledge represented through it (see Section 4.3.2).
Another problem deals with the granularity of the model. A generic model might not be useful for specific scenarios in the same domain, as it would lack specific parameters.

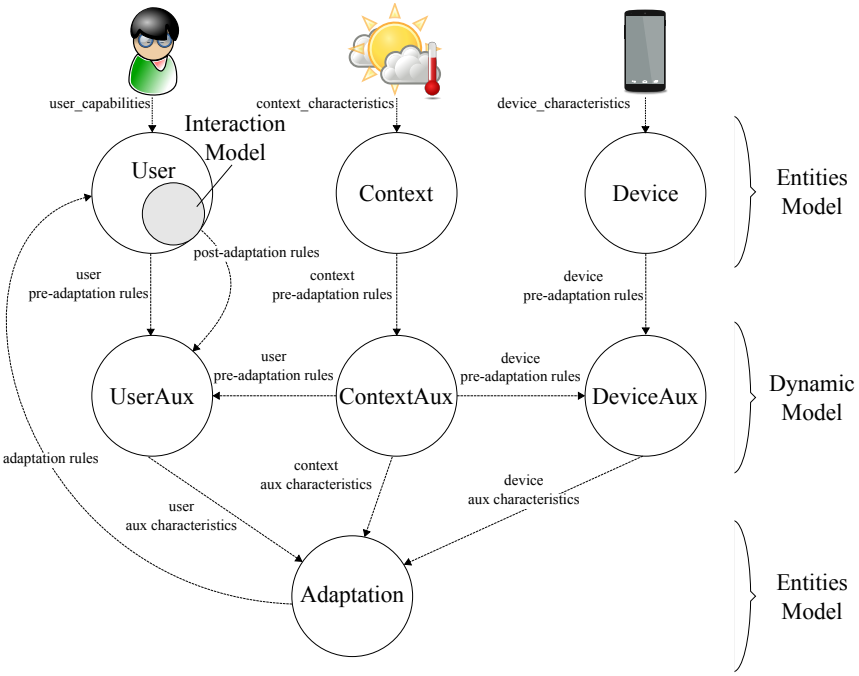

Figure 3.9: Knowledge flow through the adaptation process. The circles represent several main concepts presented in the AdaptUIOnt ontology. The arrows represent the set of rules that affect the related concepts in the circles.

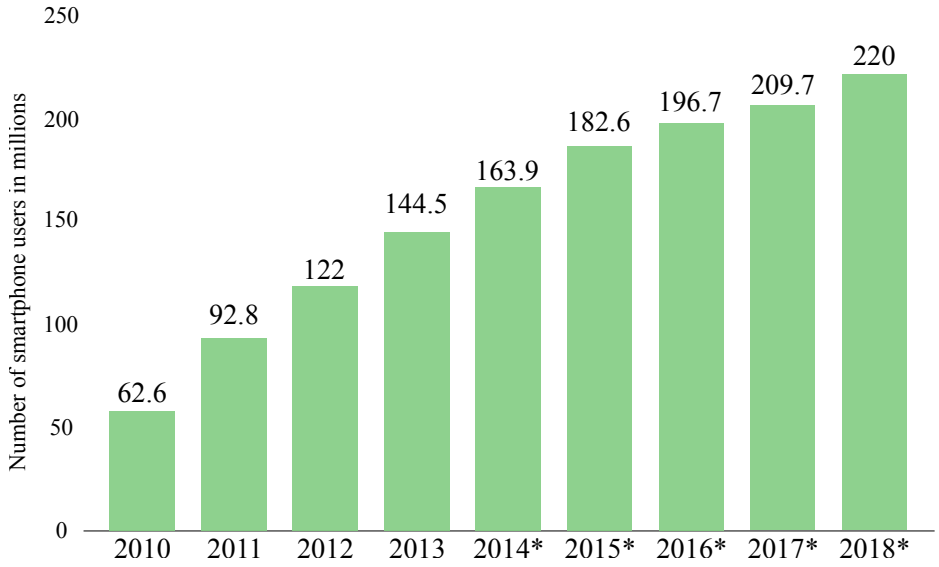

Figure 3.10: Number of smartphone users in the U.S. from 2010 to 2018 (in millions) [33]. This forecast shows the anticipated number of smartphone users in the U.S. from 2014 to 2018, based on figures from 2010 to 2013. The source estimates that there will be more than 196 million smartphone users in the U.S. by the year 2016.

On the other hand, if it is too specific they may not apply well to other domains. The AdaptUIOnt model tries to address this second aspect by being abstract regarding the user disabilities.

Regarding the presented set of rules, it is significant how they represent the three step adaptation understood in this dissertation. This is translated into a three subset of rules which aim to modify and generate new knowledge through a reasoning process. These subsets, called pre-adaptation, adaptation and post-adaptation rules, are characterized not only for gathering several rules, but also for being modifiable by developers through a series of tools provided by the platform. These tools and their characteristics are detailed in Chapter 4.

Sounds come crashing and I hear laughing, all those lights just blaze away. I feel a little strange inside, a little bit of Jekyll, a little Mr. Hyde.

Rory Gallagher

CHAPTER

4

The AdaptUI System Architecture

During this dissertation several problems of adaptive systems have been remarked. Special attention has been paid to their lack of dynamism, the problems of modelling (*what* and *how* to model, the *depth* of the model and the *required knowledge* in the domain) and the external services and platforms dependency to perform computational complex operations. Addressing these issues, this dissertation aims a fully operational mobile platform which also acts as a demonstrator of the models described in Chapter 3. This chapter describes the three-layered architecture designed for AdaptUI, the adaptation platform. The architecture layers and their modules are depicted in Figure 4.1.

Each layer of the AdaptUI adaptation platform is composed of several modules; each module aiming to solve a concrete requisite. First, there is the Modelling Layer which collects information of the user's (non physiological) capabilities and context and device characteristics to finally build a semantic representation of this knowledge. Two main modules form the Modelling Layer: the Capabilities Collector and the Semantic Modeller. On the one hand, the Capabilities Collector's main goal is to obtain information about each entity. On the other hand, the Semantic Modeller leads the semantic representation and storage of the information collected by the Capabilities Collector.

Next, there is the Adaptation Layer, which manages the adaptation process of the user interface components through the Adaptation Engine. Together with the Adaptation Engine there is the Adaptation Polisher module. This module aims to refine the adapted user interface by analysing several usability metrics of the user interaction.

Finally, the Application Layer provides several tools for developers to be able to use AdaptUI. Through the provided API developers are allowed not only to make their appli-

cations adaptive, but also to add, edit and delete knowledge and rules.

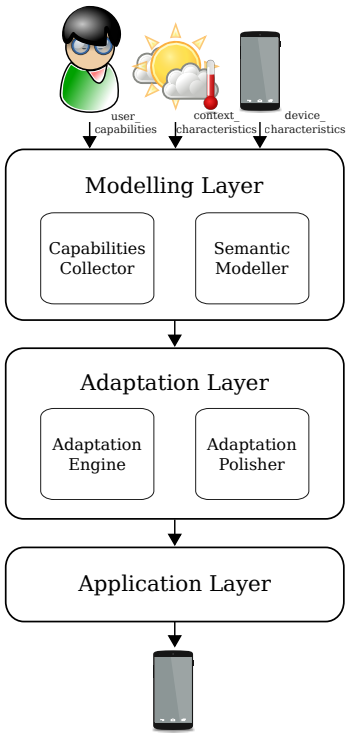

Figure 4.1: AdaptUI's three-layered global architecture.

In the next sections of this chapter each layer and module is more concretely explained. In Section 4.1 the Modelling Layer and its two modules are introduced. The Capabilities Collector is described in Section 4.1.1, while the Semantic Modeller is detailed in Section 4.1.2. The Adaptation Layer is presented in Section 4.2. This section also details the Adaptation Polisher (see Section 4.2.2). Next, Section 4.3 presents the characteristics of the Application Layer.
In addition, in Section 4.4, the relationships among the mentioned layers and modules and the flow of the information and knowledge within the AdaptUI platform is depicted. Finally, a detailed example is described in Section 4.5 following the information flow mentioned in Section 4.4, highlighting the concrete tasks performed by each layer and module.

4.1 The Modelling Layer

The first layer to be described of the AdaptUI architecture is the Modelling Layer. This layer aims to generate the different profiles (semantic models) for the three main entities: the user, the current context and the device. In order to do this, the Modelling Layer combines the results of two different modules: the Capabilities Collector and the Semantic Modeller. The Capabilities Collector collects information about the three main entities.

This module deals with the current capabilities of the user, the environment current situation, and with several characteristics of the device, storing the collected information for further usage. On the other hand, the Semantic Modeller represents the knowledge gathered and stored by the Capabilities Collector in a semantic model specified at the AdaptUIOnt ontology (which is fully described in Chapter 3).
The two modules which form the Modelling Layer are described in the following sections. First, the Capabilities Collector is introduced in Section 4.1.1. Next, the Semantic Modeller is presented in Section 4.1.2.

4.1.1 The Capabilities Collector

As mentioned in the literature, Fischer [82] defended that the user evolves through time. More concretely, time and experience are two of the reasons for the evolution of user's characteristics. In AdaptUI users evolve, but under different assumptions. Fischer considered long terms parameters as the keys for the evolution of the user. On the contrary, AdaptUI takes into account temporary and concrete context conditions, limited to a specific instant. Consequently, the user model is built contemplating a dynamic user whose capabilities change due to context conditions.
Considering this dynamic user perspective based on several aspects related to context terms, we discussed how this could be applicable to mobile devices. Mobile devices have several characteristics that make them susceptible to change (i.e., battery consumption, location awareness, preferences of the screen or sound). Thus, in AdaptUI mobile devices are also considered as a dynamic entity.
Finally, regarding the surrounding environment, we consider that it also has the propensity to change (i.e., temperature, light or noise). Therefore, within the Modelling Layer a concrete module to collect these characteristics has been designed: the Capabilities Collector.
The Capabilities Collector is a software module that allows the system gathering non physiological capabilities of the user, collect different variables of the current environment situation, and identify several device characteristics in order to be aware of the whole domain limitations. Figure 4.2 illustrates how the Capabilities Collector uses several activities to collect the information about the three entities. This information is transferred to the Semantic Modeller.

4.1.1.1 Android Activity

In Android, activities [2] are application components that provide a screen with which the user can interact. Each activity is given a window in which to draw its user interface. Android applications usually consist of multiple activities that bound to each other.
Android applications have to declare a Main activity. This activity is always presented

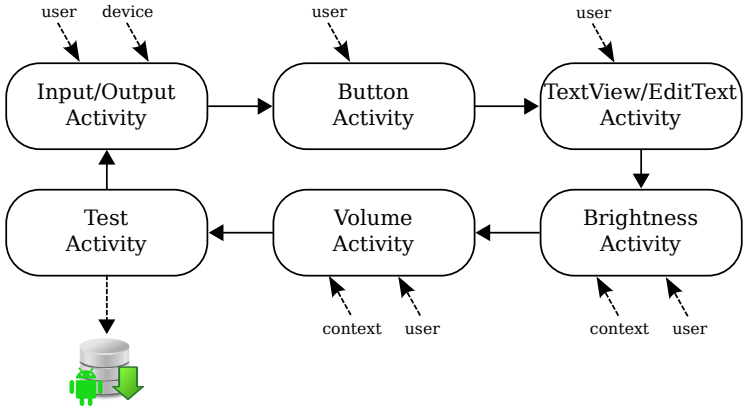

Figure 4.2: The Capabilities Collector activities and their relationships with the three main entities in AdaptUI.

to the user when launching the application for the first time. Besides, activities can start other activities storing their current status (if needed). An activity lifecycle is illustrated in Figure 4.3.

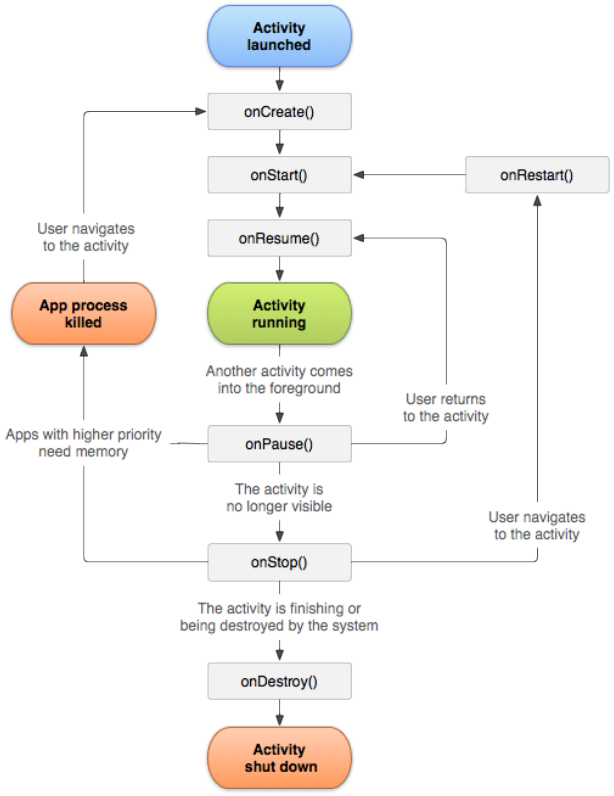

Figure 4.3: The activity lifecycle.

To create an activity the developer has to extend the Activity class. A main callback method is needed to be implemented: the *onCreate()* method. The system always calls this method when creating an activity. In it, the essential components of the activity

should be initialized. Besides, and before any initialization, the *setContentView()* method has to be called. This method defines the layout for the activity's user interface, which is defined as an XML file in the *layout* folder of the Android project.
Listing 4 shows an example of an activity initialization. The layout of the activity is shown in Listing 5. Next, Figure 4.4 illustrates the result of such activity.

```
1  public class ExampleActivity extends Activity {
2    private Button button;
3
4    @Override
5    public void onCreate(Bundle savedInstanceState) {
6      super.onCreate(savedInstanceState);
7      setContentView(R.layout.activity_main);
8
9      // Here the developer can initialize components for
10     // this activity, such as buttons, text views, etc.
11     button = (Button) findViewById(R.id.button_example);
12     // ...
13     }
14 }
```

Listing 4: Example of an activity initialized with a button.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent"
5      android:orientation="vertical" >
6      <!--Elements to be shown in the layout-->
7      <Button
8          android:id="@+id/button_example"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:text="Button" />
12     <!--...-->
13 </LinearLayout>
```

Listing 5: An activity layout declaring a button.

It is also important to remember defining the activity in the *AndroidManifest.xml* file. This file gathers the main specification of the declared activities, filters, services and allowed permissions, along with the identification of the application. Listing 6 shows an example of a typical manifest file.

Figure 4.4: The resulting activity from the combination of Listing 4, Listing 5 and Listing 6.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="es.android.example"
android:versionCode="1"
android:versionName="1.0" >

<uses-sdk android:minSdkVersion="16" />
<!--Permissions and services declaration-->

<application
  android:icon="@drawable/ic_launcher"
  android:label="@string/app_name" >
  <!--List of activities-->
  <activity
    android:label="@string/app_name"
    android:name=".MyAndroidAppActivity" >
    <intent-filter >
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
</application>
</manifest>
```

Listing 6: Application manifest file.

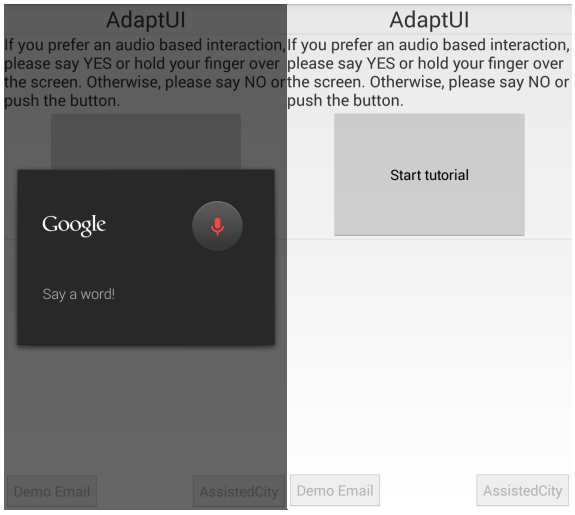

Figure 4.5: Capabilities Collector's input activity.

4.1.1.2 Collecting the User's Capabilities

AdaptUI requires the user's capabilities (together with the current context situation and the characteristics of the mobile device) as an input for the adaptation process. As explained in Section 3.1.3, the user model of the AdaptUIOnt model does not consider physiological knowledge about the user. This is due to the lack of physiological and medical background that users (and developers) of AdaptUI might have, which makes the capabilities representation inaccurate and impractical. Instead of this, the user model within the AdaptUIOnt ontology provides a layer of abstraction, focusing on the user interaction needs. For example, a user who suffers from a hearing disability in AdaptUI can explicitly configure a minimum sound level, so the system will not reduce it in any future adaptation. Thus, we avoid modelling specific physiological problems related to the user's hearing capability. To collect this kind of information about the user, the Capabilities Collector shows several screens (known as activities in Android) to the user in which different interactions are required.

The first capability that the Capabilities Collector deals with is the input method or type of interaction. In the AdaptUIOnt ontology this is represented through the *Interface* class. Through several basic instructions (presented in text mode and as quick audio question) the user selects his/her input and output preferences. For example, Figure 4.5 shows the first activity of the Capabilities Collector module. The instructions shown in this activity can be read by the user and by the application itself (for example, by using the Android TextToSpeech (TTS) [37] service). Every decision the user makes is stored in the mobile phone as a profile for a future *semantization* by the Semantic Modeller.

Once the input interaction type has been determined the output is configured accordingly. Next activities show several Android view [4] components which are configurable by the user. Views represent the basic building block for user interface components in Android.

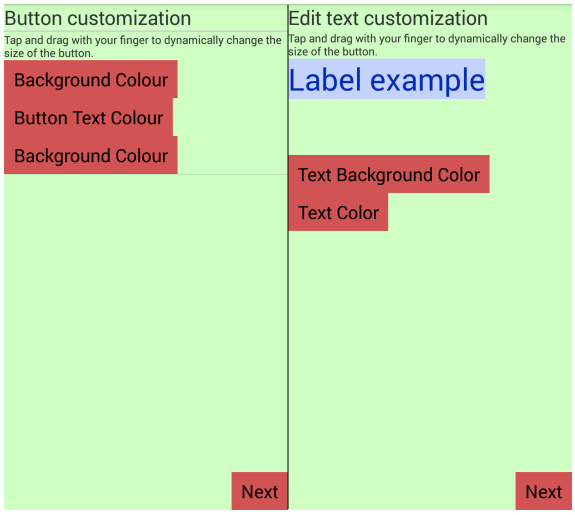

Figure 4.6: Different Android view components personalization: on the left, button; on the right, text view and edit text.

A view occupies a particular area on the screen and is responsible for drawing and event handling. In this case, a button, a text view and an edit text (which is a particular case of a text view) are presented (see Figure 4.6). The Capabilities Collector allows the user to modify their size, component colour, and also the text size and colour.
Finally, the display brightness and the system volume are also allowed to be customized. The surrounding light and noise are monitored using the available device sensors, which are listed by the Capabilities Collector when it is first launched. Thus, AdaptUI is aware of the context conditions and is able to adjust these parameters to the user's preferences.

4.1.1.3 The Context Situation

Current smartphones are equipped with several sensors that allow devices to collect different environment parameters. Light sensors, microphones, proximity sensors for disabling the screen, even Bluetooth and infra-red transceivers are just several examples of the hardware available in such devices.
Taking advantage of the current situation, in which smartphones are able to sense the environment, the Capabilities Collector collects knowledge of the user context. Light conditions and noise are classified by the Capabilities Collector as follows:

- Light conditions are measured in lx, which is the International System of Units (SI) unit of luminance [112].

- The current noise is collected using the mobile available microphones. It is measured in dB, which is a logarithmic unit that expresses the ratio between two values of a physical quantity (often power or intensity).

Once all this features have been collected, the profile is temporarily stored in the device. This storage is carried out using the Android SharedPreferences [32]. This class provides a general persistence framework for developers to save and retrieve key-value pairs of primitive data types. After this temporary storage in the device, the Semantic Modeller is the module which deals with the task of the semantic representation of the model. Listing 7 shows how developers deal with the SharedPreferences to store and retrieve data in Android. If complex objects are used, the process is different (see Listing 8).

```
//Creating a shared preference
SharedPreferences  mPrefs = getPreferences(MODE_PRIVATE);

//Saving data in the editor
Editor editor = mPrefs.edit();
editor.putBoolean("key_name", true);
editor.putString("key_name", "string value");
editor.commit();

//Retreiving data
mPrefs.getBoolean("key_name", null);
mPrefs.getString("key_name", null);

//Removing data
editor.remove("key_name"); // will delete key name
editor.commit(); // commit changes
```

Listing 7: Using Android SharedPreferences. By default SharedPreferences allows to store and retrieve primitive data. Implementing Parcelable allows complex objects to be persistent.

```
public class CustomImplementation implements Parcelable {
  private int intAttribute;
  private float floatAttribute;

  public UserMinimumPreferences(int intAttr, float floatAttr) {
    super();
    this.intAttribute = intAttr;
    this.floatAttribute = floatAttr;
  }

  public CustomImplementation(Parcel in) {
    readFromParcel(in);
  }

```

```
15   public static Parcelable.Creator<CustomImplementation> getCreator() {
16     return CREATOR;
17   }
18
19   // Writing in the Parcelable object
20   @Override
21   public void writeToParcel(Parcel dest, int flags) {
22     dest.writeInt(intAttribute);
23     dest.writeFloat(floatAttribute);
24   }
25
26   // Reading from the Parcelable object
27   private void readFromParcel(Parcel in) {
28     intAttribute = in.readInt();
29     floatAttribute = in.readFloat();
30   }
```

Listing 8: Using Android SharedPreferences to store non-primitive objects.

Figure 4.7 shows how the brightness and the volume of the application is configured by a user which is aware of the light and noise of the environment.

4.1.1.4 The Device Characteristics

Device characteristics are significant within the AdaptUI platform. These characteristics limit the possible adaptations of the user interface. For example, not all the smartphones have the same maximum brightness or sound levels, nor they have the same connectivity capabilities. Thus, being aware of each device's capabilities becomes essential in AdaptUI.

The device characteristics are collected by the Capabilities Collector by using the *Settings.System* class in Android. This class contains global system-level device preferences. Listing 9 shows a piece of code which requests the maximum values for the brightness. To allow the user to change the brightness of the device's screen a NumberPicker element is shown. In its initialization, it is needed to specify the minimum and the maximum values of the range. Thus, the user is able to configure the brightness to 0. For the maximum brightness value it is needed to state a default value of $1F$.

```
static final int MAX_BRIGHTNESS = 1F;
static final int MIN_BRIGHTNESS = 0;

brightnessPicker = (NumberPicker) findViewById(R.id.brightness_picker);
brightnessPicker.setMinValue(MIN_BRIGHTNESS);

WindowManager.LayoutParams layoutParams = getWindow().getAttributes();
layoutParams.screenBrightness = MAX_BRIGHTNESS;
getWindow().setAttributes(layoutParams);

currentBrightness = android.provider.Settings.System.
                        getInt(getContentResolver(),
                        android.provider.Settings.System.
                        SCREEN_BRIGHTNESS,-1);

brightnessPicker.setMaxValue(currentBrightness);
```

Listing 9: Android NumberPicker initialization with a minimum value of 0 and a maximum value requested to the *Settings.System class*.

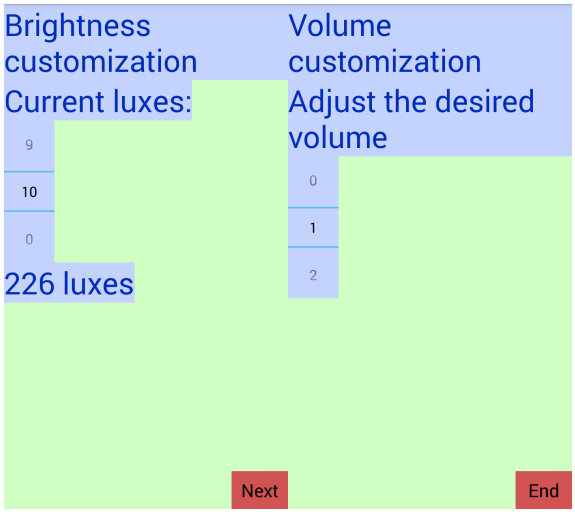

Figure 4.7: Brightness (left) and Volume (right) adjustment sensing the surrounding light and noise.

On the contrary, to get values related to audio, the *AudioManager* class is required. Before asking the user about the way of interaction, the Capabilities Collector performs an analysis about several device's capabilities. Several hardware details are collected: processor maximum speed, processor availability, battery current levels, maximum and current available memory, available input modes, and so forth. All the characteristics are

shown in Table 4.1. These are stored during the process for the Semantic Modeller to store in the final model.

Characteristic	Description
Brightness	Describes the current device's brightness.
Contrast	Represents the current device's contrast.
Volume	Describes the current device's volume.
Processor	Models the number of the device's cores.
OS version	Indicates the current OS version.
Acceleration	Represents the current X, Y, Z acceleration.
Orientation	Current orientation of the device.
Device memory	Maximum device memory.
Available memory	Current available device memory.
Battery	Current device battery levels.
Screen resolution	Maximum screen resolution.
Storage	Maximum available storage.
Available storage	Current available storage.
Processor	Maximum processor of the device.
Available processor	Current processor availability.

Table 4.1: Requested device characteristics.

4.1.2 The Semantic Modeller

Once the Capabilities Collector has finished its task, all the collected knowledge is stored in the mobile. The Semantic Modeller's goal is to build a semantic representation of this knowledge through the AdaptUIOnt ontology and store it in the mobile device. This task requires from a semantic reasoner. However, there are no available and remarkable reasoners written in Java and compatible with Android. Thus, an Android port of the Pellet [28] reasoning engine has been developed: *Pellet4Android*. As Pellet is available for open source applications under the terms of the Affero General Public License (AGPL) version 3 license, the provided *Pellet4Android* is also available in Github [29] under the same license regulations.

During the following subsections both versions of the Pellet reasoning engine are presented. First, in Section 4.1.2.1, the Java based Pellet reasoning for desktop is introduced. Next, Section 4.1.2.2 describes the process followed to port Pellet to Android.

As the Semantic Modeller finishes storing the corresponding knowledge in the AdaptUIOnt ontology, several rules are triggered. The main consequence of the execution of these rules is shown through the *Adaptation* class. This class is filled with the adaptation values that will be requested by the Adaptation Layer, which is the next layer of the AdaptUI architecture.

4.1.2.1 **Pellet**

Developed by Clarkparsia, Pellet is a free and open source OWL-DL reasoner written in Java. As a DL reasoner, it provides several standard inference services:

- *OWL-DL consistency checking*: Pellet assures that the knowledge represented in the ontology does not contain contradictory facts. Carroll and De Roo [62] define that *"an OWL consistency checker takes a document as input, and returns one word being Consistent, Inconsistent, or Unknown."* In DL terminology this is the operation to check the consistency of an ABox with respect to a TBox (see the description of this DL terminology in Table 4.2).
- *Concept satisfiability*: Determines whether it is possible for a class to have any instances. This avoids cases in which unsatisfiable classes might define instances, which will cause the ontology to be inconsistent.
- *Classification*: Pellet is able to create the complete class hierarchy from the computation of the subclasses relations between every named class. This allows performing future queries (i.e., getting all the direct subclasses of a class).
- *Realization*: It is able to find the most specific classes that an individual belongs to. Realization directly depends on classification, and it cannot be done without it.

Component	**Description**
ABox (Assertional Box)	Contains assertions about individuals (i.e., OWL facts such as type, property -value, equality or inequality assertions).
TBox (Terminological Box)	Contains axioms about classes, i.e., OWL axioms such as subclass, equivalent class or disjointness axioms.
Knowledge Database	A combination of ABox and TBox (i.e., a complete ontology).

Table 4.2: Several DL terminology.

The reasoning capabilities of Pellet are accessible through different interfaces. For example, Pellet is directly integrated with the Protégé ontology editor [30]. Thus, reasoning capabilities are allowed through the user interface of this editor. For developers, there are several tools provided as APIs which allow the utilisation of Pellet. The most significant ones are Apache Jena [19] and Manchester OWL-API [26]:

- Jena: Apache Jena (or Jena) is a free and open source framework written in Java which allows building semantic web and Linked Data applications. The framework is composed of different APIs interacting together to process RDF data.

- OWL-API: Available under The GNU Lesser General Public License (LGPL) or Apache Licenses, the OWL-API is an open source high level API maintained by the University of Manchester for working with OWL ontologies. Closely aligned with the OWL 2 structural specification, the OWL-API supports parsing and rendering in the syntaxes defined in the W3C specification (i.e., Functional Syntax, RDF/XML, OWL/XML and the Manchester OWL Syntax), the manipulation of ontological structures, and the use of reasoning engines (i.e., Chainsaw, FaCT++, JFact, HermiT, Pellet and RacerPro).

4.1.2.2 **Reasoning with Pellet in Android: Pellet4Android**

The AdaptUI platform was conceived to support reasoning and semantic knowledge representation due to the benefits that ontologies bring. Nevertheless, after analysing the possibilities and solutions provided by the community (i.e., mobile reasoning platforms) we found that, although nowadays mobile capabilities allow heavier processing, there are no remarkable efforts in the area.
Yus et al. [147] noticed this issue and made great efforts porting several Java based reasoners to Android. Although the authors do not provide the developed ports to the community, they provide several instructions to make these reasoners available for Android devices. Thus, following these instructions a port of Pellet has been developed for AdaptUI: *Pellet4Android*.
As Pellet, *Pellet4Android* also supports full OWL 2 and DL rules. To make Pellet run in Android the following steps were needed:

- Pellet uses Jena, which is a free and open source framework written in Java which allows building semantic web and Linked Data applications. However, Jena is not directly importable to Android. Thus, it is necessary to replace it with Androjena [6]. Androjena is a porting of Hewlett-Packard's Jena semantic web framework to Android.

- Pellet also imports the Java Architecture for XML Binding (JAXB) library [18] for XML parsing. JAXB is able to translate XML Schemas building a set of classes that correspond to that schema. It originally comes with the Java 1.6 release, but its easily importable to previous Java version integrating the JAXB jar files. The problem is that JAXB uses classes that are not compatible with Android. Thus, this package has been removed.

- As the official Oracle documentation states, the *javax.xml.bind* package *"provides a runtime binding framework for client applications including unmarshalling, marshalling, and validation capabilities"* [17]. Nevertheless, this package is also in-

compatible with Android, and we should remove it from the package list and the build path.

- Also related with the way Java manages XML files, the *org.apache.xerces* [41] package is used for the creation and maintenance of XML parsers and related software components. Several classes under this package are not importable by Android, so the package has to be removed.

- Finally, under the *com.clarkparsia.pellet* package there are several classes that are not directly importable by Android. Thus, without removing them, we have to search the specific troublesome lines of code (usually related with concrete data types and exceptions) and comment them.

Figure 4.8 shows the package structure of the *Pellet4Android* port and the imported libraries in the Android project.

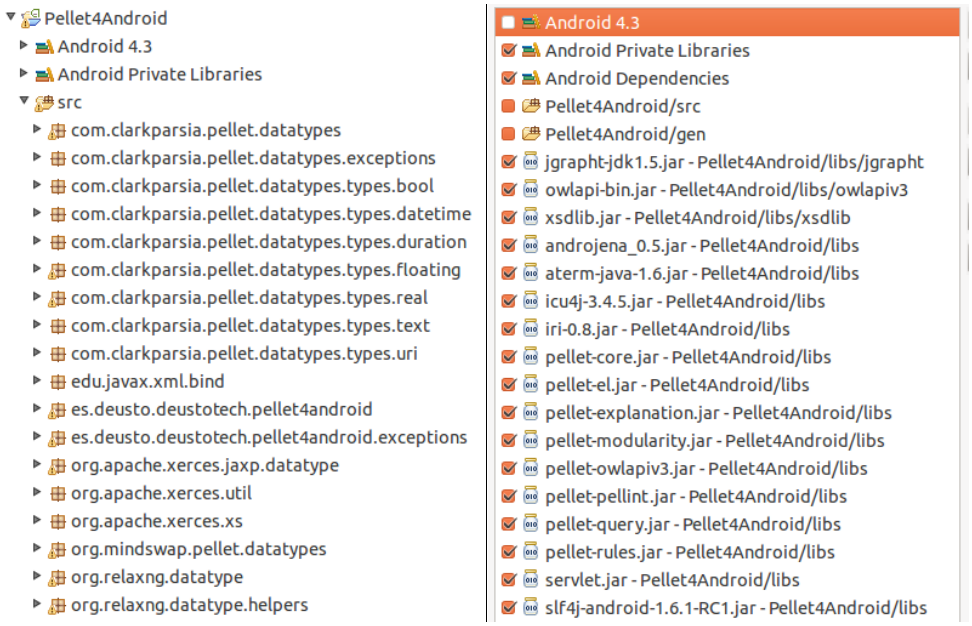

Figure 4.8: Package structure (left) and needed libraries (right) for Pellet4Android.

As Yus et al. [147] remarked, it is important to provide a larger heap size. To do this, the *android:largeHeap* tag of the AndroidManifest.xml file of the project has to be changed to *true*.
A first evaluation of the port was carried out by Yus et al. [147]. Nevertheless, Bobed et al. [53] performed a second evaluation of the Pellet version for Android obtaining promising results in terms of efficiency and time responsiveness. Although in this dissertation a similar approach to port Pellet has been followed, it is impossible to be a 100% sure if the results are the same (as the version by Yus et al. [147] was not available for testing). Thus, in Chapter 5 an evaluation of *Pellet4Android* is included.

4.2 The Adaptation Layer

The second layer of the AdaptUI architecture is the Adaptation Layer. After the processes performed by the Modelling Layer, the Adaptation Layer aims to lead the dynamic adaptation of the elements presented in the user interface. It is formed by two different modules: The Adaptation Engine, whose purpose is to adapt the currently interface shown by the device, and the Adaptation Polisher, which aims the refinement of the user interface basing its task in several usability metrics.

4.2.1 The Adaptation Engine

After the storage of the domain knowledge in the AdaptUIOnt ontology by the Semantic Modeller, several rules are triggered. These rules are grouped in three different subsets: the pre-adaptation rules, the adaptation rules and the post-adaptation rules. More concretely, the adaptation rules subset is the one which modifies the knowledge represented by the *Adaptation* class in the AdaptUIOnt ontology.
Once these rules have been executed the Adaptation Engine requests these results to the *Adaptation* class. Then, it launches several methods to dynamically change the aspect of the current user interface. These methods basically redraw and refresh the different components shown in the current activity, sharing their new characteristics to the rest of activities.
Listing 10 shows an example of how several views are redrawn. First, the elements that are part of the activity (i.e., buttons and textviews) are initialized (similarly to other applications). Next, several methods regarding the adaptation are called. The *redrawViews()* method takes into account the user's configured profile through the Capabilities Collector and adapts the components of the activity accordingly. Every activity overwrites this method (and others), as they extend from a parent abstract class called *AbstractActivity*. This class mainly manages the services initialization (e.g., TextToSpeech) and the ontology.

```
1  @Override
2  protected void onCreate(Bundle savedInstanceState) {
3    super.onCreate(savedInstanceState);
4    setContentView(R.layout.activity);
5
6    //Get the stored parameters in the mobile
7    Bundle bundle = getIntent().getExtras();
8    userPrefs = bundle.getParcelable(getResources().
9                          getString(R.string.view_params));
10
11   //Initializing an interface button
```

```
12    btnNext = (Button) findViewById(R.id.button_next);
13
14    //Calling different needed methods
15    redrawViews();
16    initializeServices(TAG);
17    addListeners();
18  }
19
20  //Modifying the button
21  @Override
22  public void redrawViews() {
23    btnNext.setMinimumWidth(userPrefs.getButtonWidth());
24    btnNext.setMinimumHeight(userPrefs.getButtonHeight());
25    btnNext.setTextColor(userPrefs.getButtonTextColor());
26    btnNext.setTextSize(userPrefs.getButtonTextSize());
27    //...
28  }
```

Listing 10: Example of the creation and adaptation of an activity. In this case the example is centred in the adaptation of a button.

Listing 11 shows a piece of source code where the developer uses the OWL-API to store the user profile in the AdaptUIOnt model.

```
1  private void storeUserPreferencesIntoOntology() {
2      ontManager.addDataTypePropertyValue(
3          getAudios().get(0), "audioHasVolume", volumeLevel);
4      ontManager.addDataTypePropertyValue(
5          getDisplays().get(0), "displayHasBrightness",
6          userPrefs.getBrightness());
7      ontManager.addDataTypePropertyValue(
8          getLights().get(0), "contextHasLight", userPrefs.getLuxes());
9      //...and so forth
10 }
```

Listing 11: Inserting values in the corresponding classes of the AdaptUIOnt ontology.

Finally, Figure 4.9 shows two activities with the same user interface. The difference is that the activity on the left is showing the default user interface defined in the activity's layout. On the contrary, the activity on the right shows adapted components.

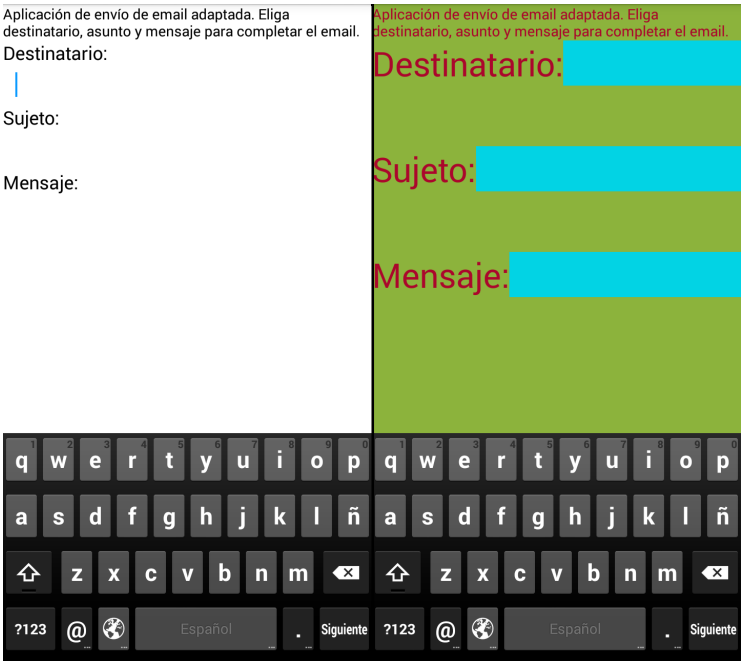

Figure 4.9: User interface adaptation performed by the Adaptation Engine. On the left, a default activity with no adaptation. On the right, the same activity after the adaptation process. As is shown, the colours sets and sizes of each component of the adapted user interface are different from the non adapted one.

4.2.2 The Adaptation Polisher

Although the adaptations for the user follow the instructions detailed by him/her, there is still the possibility that the Adaptation Engine's results lead to a unsuccessful interaction/adaptation. One of the main reasons for this is that the classification of the model, considering the context aspects, is just an approximation of the reality. For example, considering that between 1,000 lx and 25,000 lx the reasoner determines that the ontology value for the luminance is *daylight*, the difference might be significant in real scenarios (see Table 3.7). In other words, the user might easily interact with a 1,000 lx context light but tediously in a 24,999 lx light environment. In order to tackle this problem, there are two possible solutions: to consider an extensive set of classification rules including more possible situations (e.g., dividing the luminance table into more categories), or to design a specific module which evaluates the user interaction results: the Adaptation Polisher.
The first case is difficult to implement. Nevertheless, AdaptUI provides an API which aims to help in this issue allowing developers to create and modify the ontology knowledge. This means that it is possible to try to model every tiny context variation to capture different context characteristics and adapt the user interface accordingly. However, this is not practical, and AdaptUI covers the second case with the Adaptation Polisher. Therefore developers do not have to model these small variations in the environment.

The Adaptation Polisher is a software module, part of the Adaptation Layer, which monitors the effectiveness and responsiveness of the adapted interfaces. Collecting different usability and productivity metrics of the interaction carried out by the user, this module is able to make small but specific adaptations to improve the ongoing interaction. This module has been designed considering the relative user efficiency productivity metrics. These metrics compare the efficiency of the user compared to an expert. But it has no sense to compare the user with others in AdaptUI. The system cannot generalize and apply adaptations based on other users' preferences. To solve this, in AdaptUI we propose to maintain a base adaptation, which thanks to the Adaptation Polisher and its interaction results will be improved. Consequently, an interaction model is built for each adaptation. Therefore, we can determine the efficiency of the user when he/she is manipulating an adaptation made by the system by comparing the last adaptation to the previous one.

4.2.2.1 **The Usability Metrics**

For this module, several usability metrics have been studied and implemented. These metrics are classified into two different groups: *effectiveness* metrics and *productivity* metrics. Table 4.3 and Table 4.4 detail the usability metrics implemented in AdaptUI. The following information is given for each metric in the cited tables:

a) Purpose of the metric: Expresses the main goal of the current metric.

b) Measurement, formula and data element computations: Provide the formulas to compute the current metric and the meaning of the used data elements.

c) Interpretation of the measured value: Details the range and preferred values.

d) Metric scale type: Provides the type of scale used by the current metrics. The possible scale types are: Nominal, Ordinal, Interval, Ratio and Absolute scale.

e) Measure type: Provides the type of the measure. The possible measure types are: Size, Time and Count.

In the original International Organization for Standardization (ISO) document [101] there are 4 extra columns that have not been included in Table 4.3 and Table 4.4. This is because the values for each metric under these columns are the same:

a) Method of application: Provides an outline of the application.

b) Input to measurement: Details the source of data used in the measurement. In this case there are two inputs that each metric shares: Operation (test) report and User monitoring record.

Metric name	Purpose of the metrics	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type
Task effectiveness	To measure the proportion of the goals of the task achieved correctly.	$M1 = \|1 - \Sigma A_i\|$ A_i = proportional value of each missing or incorrect component in the task output	$0 \leq M1 \leq 1$ The closer to 1.0 the better.	—	A = proportion
Task completion	To measure the proportion of the task that is completed.	$X = A/B$ A = number of tasks completed B = total number of tasks attempted	$0 \leq X \leq 1$ The closer to 1.0 the better.	Ratio	A = Count B = Count X = Count/Count
Error frequency	To measure the frequency of errors.	$X = A/T$ A = number of errors made by the user T = time or number of tasks	$0 \leq X$ The closer to 0 the better.	Absolute	A = Count

Table 4.3: The effectiveness metrics used in the Adaptation Polisher, as it appears in [101].

c) Reference: Identifies software life cycle processes where the metric is applicable. There are three processes that each metric shares: 6.5 Validation, 5.3 Qualification testing and 5.4 Operation.

d) Target audience: Identifies the user(s) of the measurement results. Again, the metrics share User and Human interface designer as their audiences.

4.2.2.1.1 Effectiveness Metrics

Effectiveness metrics, as detailed in the ISO/International Electrotechnical Commission (IEC) 9126-4 [101], evaluate whether the current task achieves a specific goal considering the accuracy and completeness of the corresponding task. These metrics are shown in Table 4.3.

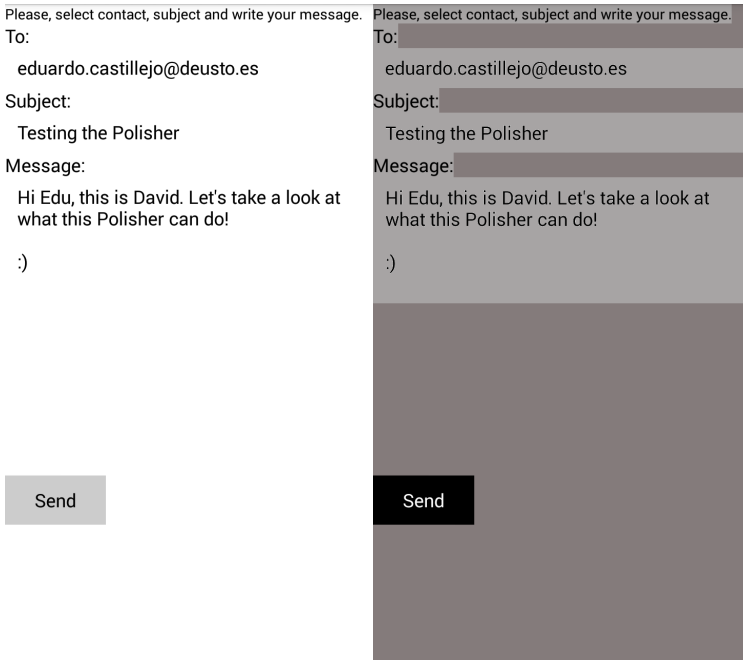

Figure 4.10: User interface adaptation performed by the Adaptation Engine. On the left, the default version, without adaptations. On the right, the same application adapted by AdaptUI.

4.2.2.1.2 Productivity Metrics

Productivity metrics evaluate the resources consumed by the users in relation to the effectiveness achieved in the current task [101]. These metrics are shown in Table 4.4.

4.2.2.2 Adaptation Polisher Scenario

In the following lines a scenario describing step by step the actions performed by the Adaptation Polisher is presented. Table 4.6 shows the inferred adaptation for the user, context and device characteristics described in Table 4.5.

As Table 4.5 shows, David does not suffer from any disability. Nevertheless, the context situation presents characteristics that might trouble David during the interaction process. The cold temperature and the lack of sufficient light requires a user interface adaptation. Thus, AdaptUI increases the device's brightness and the views' sizes. Figure 4.10 illustrates the differences between the default user interface (left) and the adapted one (right). Thus, David uses his device through the adapted user interface. At this point, the corresponding interaction model is built by the Adaptation Layer, collecting the usability metrics shown in Section 4.2.2.1 (see Table 4.3 and Table 4.4). Table 4.7 shows the used metrics and the computed values for both the default and the adapted interaction models. As is shown in Table 4.7, the resulting adapted user interface provided by AdaptUI does not improve the interaction of the user. The required time for performing the same task

Metric name	**Purpose of the metrics**	**Measurement, formula and data element computations**	**Interpretation of measured value**	**Metric scale type**	**Measure type**
Task time	To measure the required time to complete the task.	$X = Ta$ Ta = Task time	$0 \leq X$ The closer to 1.0 the better.	Interval	T = Time
Task efficiency	To measure how efficient the users are.	$X = M1/T$ $M1$ = task effectiveness T = task time	$0 \leq X \leq 1$ The larger the better.	—	T = Time X = proportion/ time
Economic productivity	To measure the cost-effectiveness of the user.	$X = M1/C$ $M1$ = task effectiveness C = total cost of the tasks	$0 \leq X$ The larger the better.	—	C = Value
Productive proportion	To measure the proportion of time the user is performing productive actions.	$X = Ta/Tb$ Ta = productive time = task time - help time - error time - search time Tb = task time	$0 \leq X \leq 1$ The closer to 1.0 the better.	Absolute	Ta = Time Tb = Time X = Time/ Time
Relative user efficiency	To measure the efficiency of the user compared to an expert.	Relative user efficiency $X = A/B$ A = ordinary user's task efficiency B = expert user's task efficiency	$0 \leq X \leq 1$ The closer to 1.0 the better.	Absolute	C = proportion/ time

Table 4.4: The productivity metrics used in the Adaptation Polisher, as it appears in [101].

	Scenario
User	
- Personal data	David, 23 years old, Spanish
- Activity	-
- Known disabilities	-
Context	
- Location	Relative: Vitoria, Spain
- Time	06:30
- Brightness	600 lx
- Temperature	-5 ºC
Device	Motorola Moto G
Task	Send an email

Table 4.5: Scenario situation summary.

Adaptation	**Value**
hasColourSet	-
hasViewSize	10
hasResponse	vibration
hasInput	Default
hasOutput	Visual and audio
hasVolume	5

Table 4.6: Final adaptation for the presented scenario.

Metric	**Value for the default interaction model**	**Value for the adapted interaction model**
Task effectiveness ($M1 = \|1 - \Sigma A_i\|$)	0.7	0.350
Task completion ($X = A/B$)	1	0
Error frequency ($X = A/T$)	0.2	0.562
Task time ($X = Ta$)	15	32
Task efficiency ($X = M1/T$)	0.046	0.010
Relative user efficiency ($X = A/B$)	1.0	0.5

Table 4.7: The interaction model computed by the Adaptation Layer. Time (T) has been measured in seconds.

(sending an email) is 32 seconds, while by default David uses 15 seconds approximately. Thus, the user interface does not fit the user needs.
Once the interaction model has been built, the Adaptation Polisher checks the usability rules set. As detailed in Section 4.2.2, these rules trigger the polisher rules if certain usability ranges are exceeded. Equation 4.1 shows a usability rule checking the relative user efficiency of the interaction model.

$$\begin{aligned} UserAux(?user)\ \&\ Productivity(?productivity)\ \&\ Polisher(?polisher)\ \&\\ userAuxHasProductivityMetrics(?user, ?productivity)\ \&\\ hasRelativeEfficiency(?productivity, ?efficiency)\ \&\\ lessThanOrEqual(?efficiency, 0.5)\ \&\\ \Rightarrow\\ launchPolisherRules(?polisher, true) \end{aligned} \tag{4.1}$$

On the other hand, Equation 4.2 is triggered by Equation 4.1. In the consequent of this rule there is a value of 1.10. This value means that the size of the views presented in the previous adapted version of the user interface should be increased in a 10% Hence, the next rule polishes the adapted user interface.

$$\begin{aligned} Polisher(?polisher)\ \&\ launchPolisherRules(?polisher, true)\ \&\\ UserAux(?user)\ \&\ userAuxHasEffectivenessMetrics(?user, ?effectiveness)\ \&\\ effectivenessMetricHasErrorFreequency(?effectiveness, ?freq?)\ \&\\ greaterThan(?freq, 0.5)\\ \Rightarrow\\ setViewSize(1.10) \end{aligned} \tag{4.2}$$

Thus, the resulting polished user interface is shown in Figure 4.11.

4.3 The Application Layer

The last layer of the AdaptUI architecture is the Application Layer. This layer aims to ease the use of the AdaptUI platform to developers. To do this, several tools have been provided. These tools have been designed not only to integrate AdaptUI within developers' applications (to adapt their user interfaces) but also to leave in their hands the decision of how to manage the knowledge of the AdaptUI platform. Thus, through the AdaptUI provided API, developers are allowed to:

a) Initialize a Pellet reasoner and connect the application to it, loading the AdaptUIOnt ontology and its rules.

b) Launch queries about the knowledge stored in the ontology in order to adapt the corresponding user interface elements.

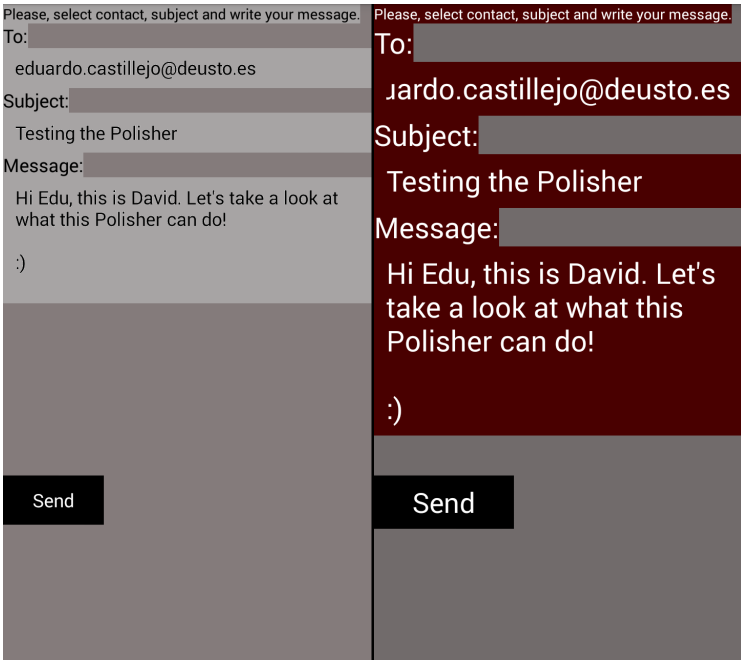

Figure 4.11: Polished user interface. On the left, the adapted version. On the right, the polished one.

c) Generate, change and adapt the knowledge contained in AdaptUI. Classes, properties and relationships described in the AdaptUIOnt ontology are fully customizable by developers to cover the domain of their adaptation problem.

d) Customize the provided set of rules. The AdaptUI API provides a set of methods to edit different rules.

The AdaptUI's API can be internally divided into two separated APIs: the adaptation API and the knowledge API. The first one, aiming to solve developer's adaptation issues, is described in Section 4.3.1; The second one's purpose is to allow developers for adapting the knowledge of the domain.
Nevertheless, this division of the API methods is a conceptual split. The provided methods are available under the same API. Table 4.8 contains the most significant methods from AdaptUI, regarding the loading of the ontology and its initialization.

Method	Description
AdaptUI()	Constructor with no parameters
AdaptUI(namespace, views)	This method requires the main ontology namespace and a Collection with the views to adapt.
loadOntologyFromFile(path, fileLocation)	It loads an ontology file from a specific Android path.
loadOntologyFromUri(uri)	Loads the ontology from a specified Uniform Resource Identifier (URI).
getExternalDirectory()	Returns the root storage folder in the current Android device.
mapOntology(namespace, fileName)	It loads the stored ontology to avoid Internet connection when managing the namespace.
getOntologyManager()	It returns the *OntologyManager* instance, which gives access to further operations (i.e., *getIndividualOfClass* and *createClass*).

Table 4.8: AdaptUI API methods.

4.3.1 The Adaptation API

The adaptation API aims to provide developers a set of methods to include user interface adaptation in their applications. As the adaptation is lead by Android views, the goal of the adaptation API is to be easy enough for any Android developer to use it. Thus, with a simple view initialization it would be possible to change the view's aspect by calling several simple AdaptUI methods. Table 4.9 details the most useful methods available in the AdaptUI's adaptation API.

Method	Description
adaptLoadedViews()	It returns a Map object which keys are the property names, and its values the stored values in the ontology. It requires the use of the constructor with parameters.
adaptViewBackgroundColor(namespace, className)	It adapts the view's background colour with the corresponding value.
adaptViewBackgroundColor(fullClassName)	
adaptViewTextColor(namespace, className)	It adapts the view's text colour with the corresponding value.
adaptViewTextColor(fullClassName)	

adaptViewTextSize(namespace, className)	It adapts the view's text size with the corresponding value.
adaptViewTextsize(fullClassName)	
adaptBrightness(namespace, className)	It adapts the display's brightness with the corresponding value.
adaptBrightness(fullClassName)	
adaptVolume(namespace, className)	It adapts the volume with the corresponding value.
adaptVolume(fullClassName)	

Table 4.9: Adaptation related AdaptUI API methods.

Listing 21 shows an example of an Android activity using AdaptUI. This example presents a simple button initialized with the values stored in the AdaptUIOnt ontology.

```
1  private static final NAMESPACE =
2          "http://www.morelab.deusto.es/ontologies/adaptui.owl#"
3
4  @Override
5  protected void onCreate(Bundle savedInstanceState) {
6    super.onCreate(savedInstanceState);
7    setContentView(R.layout.activity);
8
9    // Fetch the desired view
10   Button button = (Button) findViewById(R.id.button);
11   // Initialize an AdaptUI instance
12   AdaptUI adaptUI = new AdaptUI();
13   // Fetch the desired views instances in the ontology, in
14   // this case we want to adapt a button.
15   List<String> buttons = adaptUI.getIndividualsOfClass(
16                                          NAMESPACE, "Button");
17   // Use Android default View methods to adapt the user
18   // interface through retrieving the corresponding value
19   // from the ontology
20   button.setBackgroundColor(adaptUI.adaptViewBackgroundColor(
21                                          buttons.get(0));
22 }
```

Listing 12: An example of a button background colour adaptation.

4.3.2 The Knowledge API

Ontologies are formal specification of concepts that represent knowledge within a specific domain. This knowledge is provided through a vocabulary which describes the types and

relationships of the concepts represented in that specific domain.
The knowledge conceptualization is mainly described through classes, attributes, relations, individuals and axioms:

- Classes represent concepts within the domain. In other words, classes describe the type or kind of the members of the class.
- Each class can have a set of properties or characteristics which describe it. These properties are represented through attributes. These attributes are also called datatype properties.
- Relations detail the relationships that the concepts of the ontology might share. They are referred as object properties.
- Instances are the representation of the concepts of the ontology.
- Finally, axioms, including rules, are assertions that together comprise the overall theory that the ontology describes in the current domain.

These concepts together build ontologies to represent the knowledge of a domain. In AdaptUI,the knowledge of a domain is not considered static. Besides, the solutions provided in the literature usually do not apply well when changing the domain. Therefore, AdaptUI aims to ease the adaptation of the domain knowledge through the customization of these concepts. Through a set of methods within the AdaptUI API, developers are allowed to insert, edit and delete classes, attributes, instances and rules. Table 4.10 shows the available methods for developers to modify the knowledge contained in the AdaptUIOnt ontology.
AdaptUI is designed to support several user capabilities, context status and device characteristics. Nevertheless, regarding the rules set it is impossible to assume every possible situation and react accordingly. Thus, a set of methods to create, edit and delete SWRL rules has been provided. This allows developers to adapt the AdaptUI platform to new and unexpected situations in the domain where the platform might not behave properly.

Method	**Description**
createClass(namespace, className)	Inserts a new class.
removeClass(namespace, className)	Erases an existing class, its individuals, attributes and relationships with other classes.
createObjectProperty (namespace, indName1, indName2, objectPropName)	Inserts a new object property connecting two individuals.
removeObjectProperty(namespace, objectPropName)	Erases an existing datatype property.

createDatatypeProperty(namespace, individual, dataPropName)	Inserts a new datatype property assigning a value to a class. If it does not exist, it is created.
removeDatatypeProperty(namespace, dataTypeName)	Erases an existing datatype property.
createIndividualMembership(namespace, indName, className)	Inserts a new individual of a concrete class.
removeIndividual(name)	Removes an individual.
createSWRLRule(name, ant_classes, ant_dataprop, ant_objectprop, cons_classes, cons_dataprop, cons_objprop)	Inserts a new rule, including the set of antecedent and consequent classes and properties.
removeSWRLRule(name)	Removes a rule if the rule has a name associated to it.

Table 4.10: Knowledge related AdaptUI API methods.

Listing 13 shows an example of how AdaptUI manages the knowledge modification through this API.

```
1  private void modifyOntologyKnowldege() {
2    AdaptUI adaptUI = new AdaptUI();
3
4    List<String> classes = (List<String>) adaptUI.getClassList();
5    System.out.println("Classes: " + classes.size());
6    // Prints 'Classes: 18'
7
8    // Any modification of the ontology knowledge is
9    // performed through the OntologyManager instance,
10   // which can be accessed through AdaptUI
11   adaptUI.createClass(NAMESPACE + "Test");
12
13   classes = (List<String>) adaptUI.getClassList();
14   System.out.println("Classes: " + classes.size());
15   // Prints 'Classes: 19'
16 }
17
18 // OntologyManager class method
19 public void createClass(final String clz){
20   OWLDataFactory factory = ontology.getOWLOntologyManager()
21         .getOWLDataFactory();
22   // Creating the class
```

```
23    OWLClass clazz = factory.getOWLClass(IRI.create(clz));
24
25    // This is necessary to make the ontology use
26    // the new class, otherwise it would not refresh
27    // the ontology
28    OWLAxiom declareC = factory.getOWLDeclarationAxiom(clazz);
29    manager.addAxiom(ontology, declareC);
30  }
```

Listing 13: Modifying the ontology knowledge with the AdaptUI's API.

4.4 The Information Flow

Once the AdaptUI's layers and modules have been described, in this section the interaction among them and the information flow are reviewed. Figure 4.12 shows a diagram with the whole architecture and data flows.

First, and within the Modelling Layer, the user capabilities and context and device characteristics are collected by the Capabilities Collector. This module stores the collected information about these entities in Android using key-value sets. While the capabilities of the device and several context parameters are gathered transparently for the user, to collect his/her capabilities the Capabilities Collector needs the user to complete a profile personalization process.

Once the Capabilities Collector finishes its task, the Semantic Modeller retrieves the stored information and begins to transform it into semantics using the AdaptUIOnt ontology. To be able to use semantics and to manipulate the model and the available knowledge about the entities the Semantic Modeller uses *Pellet4Android*, which is an Android based port of Pellet.

Next, the Adaptation Engine (within the Adaptation Layer) semantically requests the last adaptation instructions. These instructions are defined by the rules triggered when the Semantic Modeller stores the knowledge in the AdaptUIOnt ontology. Once it is stored, the corresponding rules result into a series of adaptations. These adaptations, represented through the *Adaptation* class, are collected by the Adaptation Engine and represented in the user's display.

Once the user receives the corresponding adapted user interface, another Adaptation Layer's module works in background: the Adaptation Polisher. Actually, this module starts working from the beginning. Its purpose is to be aware of the user interaction with the applications. This includes not only the adapted applications but also the rest of them. Thus, it is able to establish several usability boundaries under which the user might no be comfortable with. Comparing the user interaction profile (generated during the usage of default applications under normal circumstances) and the adapted interaction model

(generated during the interaction with an adapted user interface) this module triggers several instructions to re-adapt the user interface. These instructions are ruled through the post-adaptation set of rules.
Finally, within the Application Layer several tools (provided through an API) are available for developers to make their applications adaptive and modify the knowledge which will represent the domain (including concepts, relationships and rules).

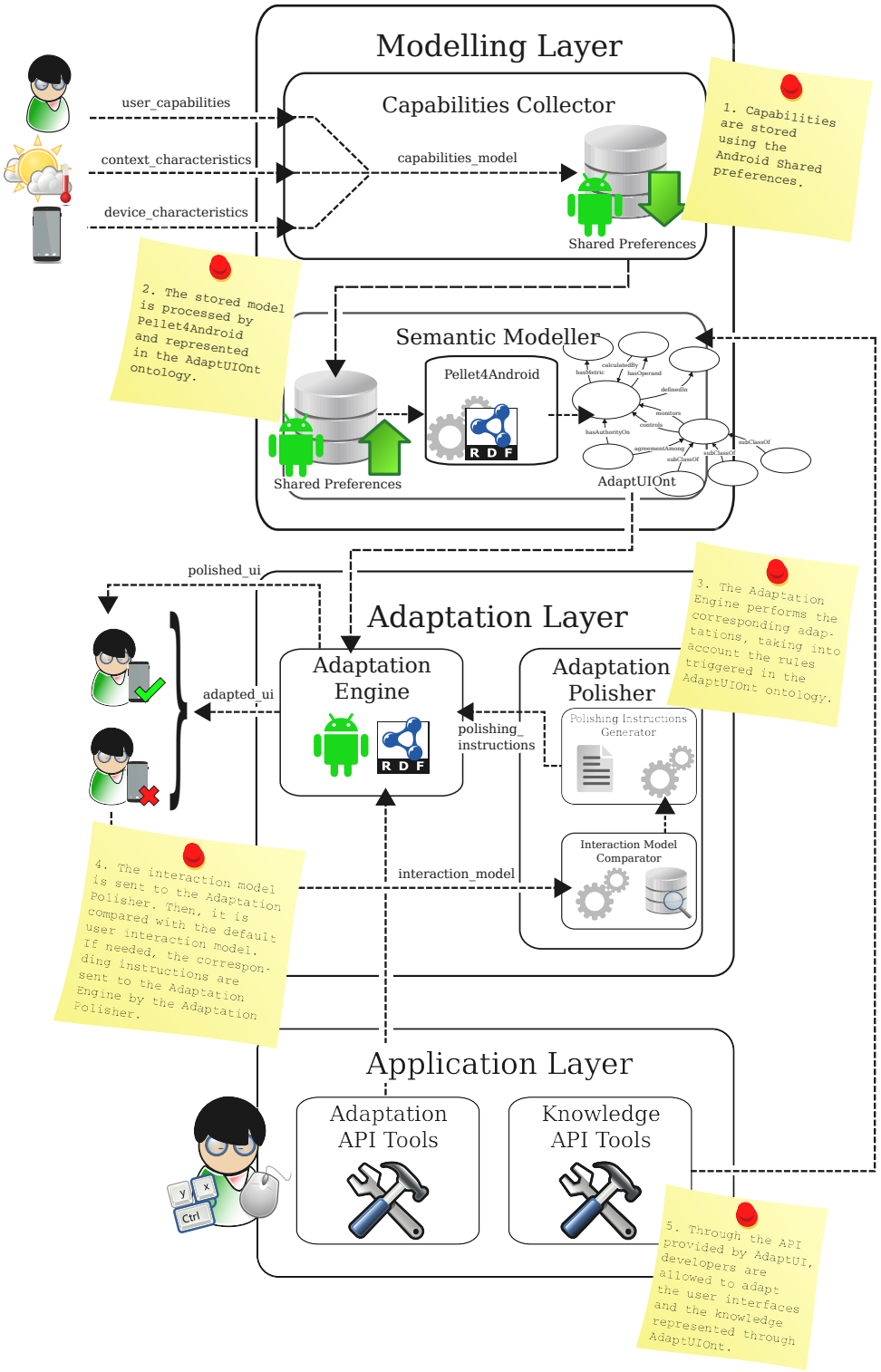

Figure 4.12: The information flow within the AdaptUI platform.

4.5 A Complete Example

The modules described in the previous sections of this chapter work together to provide an adapted user interface for the sensed user's context disabilities. The following section presents an example which details the whole process aiming to demonstrate the adaptation process for a concrete user interface. This demonstration is carried out through the introduction of particular scenario. This scenario introduces Molly, a user who is trying to make a phone call from the beach with her Android mobile device. The characteristics of Molly's context cause her problems to interact with the phone and achieve her goal. The main identified context issue is the amount of luxes in the current situation, which might induce temporary sight disabilities. The details of this scenario are shown in Table 4.11.

	Scenario
User	
- Personal data	Molly, 39 years old, United States
- Activity	Phone call
- Known disabilities	-
Context	
- Location	Relative: Plentzia, Spain
- Time	14:35
- Brightness	30,000 lx
- Noise	80 dBs
- Temperature	28ºC
Device	Samsung Galaxy SII
	Battery: 55%

Table 4.11: Scenario summary.

To make the phone call, Molly is using the application whose user interface is shown in Figure 4.13. Thus, the shown user interface should be adapted by AdaptUI if the scenario presents characteristics that might trouble Molly during the phone call process.

Figure 4.13: A simple phone call application user interface.

First, the proper functioning of the AdaptUI platform highly depends on the interaction capabilities represented in the AdaptUIOnt ontology. These capabilities are first gathered by the Capabilities Collector. This module also collects context and device characteristics to build the corresponding semantic model. Hence, the Capabilities Collector has to be run by Molly in her device. The first task this module performs is collecting several characteristics of the used device. This is made by calling several Android system methods which allow developers know the current capabilities of the device (e.g., available memory and battery). Next, several activities are presented through which Molly is asked to calibrate and configure different visual elements, such as buttons, edit texts and text views. This has to be performed considering the current context situation. Otherwise, it would not be useful for future adaptations within a similar context. As the context might vary during the capabilities collecting process, the module uses a native pre-semantic representation of the gathered knowledge. Using a simple key-value representation each interaction is stored in a SharedPreferences based model. This allows Molly to specify the needed interaction preferences rapidly. The process ends with the translation of this model into a semantic one, represented by the AdaptUIOnt ontology. This takes several seconds, and it is performed by the Semantic Modeller.

The Capabilities Collector represents the device's characteristics as Table 4.12 shows.

The most significant characteristics collected by the Capabilities Collector are the battery level and the maximum brightness and volume available levels. The first one might determine if the adaptation can lead to an excessive power consumption, while the last two determine the adaptation boundaries for brightness and volume.

Next, the activities which collect the interaction capabilities of the user represent the gathered information as shown in Table 4.13.

Due to the context situation Molly specifies a minimum brightness value of 9. Taking

Class(*)	Key	Value
OS	*osVersion*	4.1.2
Battery	*battery*	55
Volume	*maxVolume*	12
Brightness	*maxBrightness*	15
DeviceScreenResolution	*resolution*	480×800

Table 4.12: The device's characteristics represented by the Capabilities Collector.

Class(*)	Key	Value
Interface	*input*	default
	output	default
View	*colourSet(*)*	default
Display	*minBrightness*	9
	maxBrightness	max
Audio	*minVolume*	4
	maxVolume	max
Other	*language*	English
	vibration	true

Table 4.13: The user's characteristics represented by the Capabilities Collector. The colour set (*) represents the whole colour configuration that the user specifies for the corresponding views, including the background colours, text colours and so on.

into account that the maximum available value is 15, it represents a high value within the brightness range. This is due to the context brightness that she is suffering, which impedes a proper use of the display. More concretely, the current amount of luxes (30,000 lx) causes trouble when using the display. The described context situation is detailed in Table 4.14.

As mentioned before, in order to allow AdaptUI to properly adapt the user interface the user is asked to use the Capabilities Collector. Figure 4.14 shows one of the Capabilities Collector's activities in which Molly is choosing different colours for the corresponding views.

Class(*)	Key	Value
Brightness	*brightness*	30,000
Noise	*noise*	80
Time	*time*	14:35
Location	*city*	Plentzia
	country	Spain
	longitude	43.414353
	latitude	-2.944183

Table 4.14: The context's characteristics represented by the Capabilities Collector.

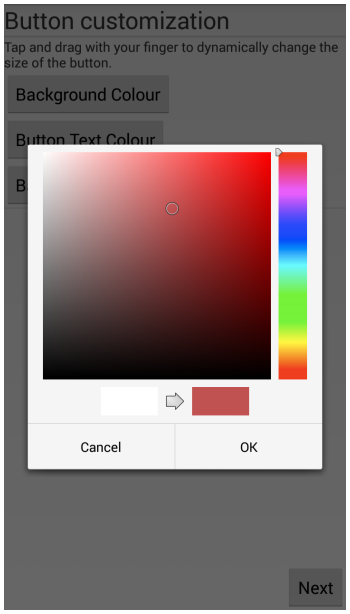

Figure 4.14: A user configuring the buttons colour set through the Capabilities Collector.

Once the interaction with the Capabilities Collector is finished, the Semantic Modeller uses the SharedPreferences stored model and stores it as a semantic representation of this model in the AdaptUIOnt ontology. The semantic representation of the knowledge represented in Table 4.13, Table 4.12 and Table 4.14 is shown in Table 4.15.
Consequently, the first rules of the AdaptUIOnt ontology are triggered. These rules form the pre-adaptation rules set, and their goal is to compute possible interaction disabilities for the user tanking into account the input user interaction capabilities, context sensed characteristics, and device features. For example, Equation 4.3 updates the *userAuxDisplayHasApplicable* datatype property by checking if the user interacts with the Capabilities Collector in a *default* manner.

$$\begin{aligned} & UserAux(?uaux)\ \&\ UserCharacteristics(?user)\ \& \\ & Interface(?interf)\ \&\ Input(?input)\ \& \\ & userHasInterface(?user, ?interf)\ \&\ interfaceHasInput(?interf, ?input)\ \& \\ & swrlb:equal(\text{“}default\text{”}) \\ & \Rightarrow \\ & userAuxDisplayHasApplicable(?uaux, true) \end{aligned} \tag{4.3}$$

As this is an intermediate and internal process Molly still does not perceive any adaptation or change. It is in the next step when the adaptation rules determine the new configuration of the presented views shown in Figure 4.13. The Adaptation Engine loads the stored AdaptUIOnt model, and it executes the corresponding adaptation rules defined by the developer, whose goal is to make the corresponding modifications on the mentioned views. Once the rules have been executed, the resulting user interface is dynamically updated

Entity	Class	Property	Value
User	Display	*userDisplayIsApplicable*	true
		userDisplayBrightnessIsStatic	false
	Audio	*userDisplayApplicableIsStatic*	false
		userAudioHasApplicable	true
		userAudioApplicableIsStatic	false
		userAudioHasVolume	4
	Interface	*userInterfaceInput*	default
		userInterfaceOutput	default
	View	*userViewIsStatic*	false
	Other	*userHasLanguage*	English
		vibration	true
Context	Brightness	*contextHasBrightness*	30,000
	Noise	*contextHasNoise*	80
	Time	*contextHasTime*	14:35
	Location	*contextHasRelativeLocationCity*	Plentzia
		contextHasRelativeLocationCountry	Spain
		contextHasAbsoluteLocationLongitude	43.414353
		contextHasAbsoluteLocationLatitude	-2.944183
Device	Software (OS)	*deviceHasOSVersion*	4.1.2
	Hardware (Battery)	*deviceHasBattery*	55
	(Volume)	*deviceHasMaxVolume*	12
	(DeviceScreenResolution)	*deviceHasResolution*	480×800

Table 4.15: User, context and device profiles as represented in the AdaptUIOnt ontology.

Figure 4.15: The default user interface (left) and the adapted user interface (right).

and presented to the user. Equation 4.4 shows how the views are updated due to the context brightness. The resulting adapted user interface is then presented to Molly, and its appearance is shown in Figure 4.15.

$$\begin{aligned} &Button(?b)\ \&\ ContextCharacteristics(?c)\ \&\ gumo : Light(?light)\ \& \\ &gumo : PhysicalEnvironment(?pe)\ \&\ contextHasPhysicalEnvironment(?c, ?pe)\ \& \\ &lightHasBrightness(?light, ?value)\ \&\ physicalEnvironmentHasLight(?pe, ?light)\ \& \\ &swrlb : greaterThanOrEqual(?value, 20000) \\ &\Rightarrow \\ &buttonHasBackgroundColor(?b, -16711936) \end{aligned} \tag{4.4}$$

Now the user interface has been adapted there are two possibilities. On the one hand, if Molly is satisfied with the provided adaptation, AdaptUI finishes its process. On the other hand, there is the possibility of rejection. In any case, the system checks the satisfaction of the user by computing several usability metrics which determine the usability comfort with the presented user interface. The Adaptation Polisher builds and compares two interaction models. The first one is built with the usability metrics detailed in Section 4.2.2.1 under default circumstances. This means that the model is built when Molly interacts with the device in a scenario in which there are no context disabilities. On the contrary, the second model, is built from the same usability metrics but taking into account the adapted user interface. The Interaction Model Comparator evaluate the usability metrics differences between the two interaction models. Once these rules are executed, their results update the last adaptation of the user interface if needed. The polished user interface is compared with the adapted one in Figure 4.16. In the presented scenario, the Equation 4.5

Figure 4.16: The adapted user interface (left) and the polished user interface (right).

updates polishes the button's text as follows:

$$\begin{aligned} &Button(?b)\ \&\ Usability(?u)\ \&\ EfficiencyMetric(?em)\ \&\\ &usabilityHasEfficiencyMetric(?u,?em)\ \&\ swrlb:greaterThanOrEqual(?value,0.5)\ \&\\ &efficiencyMetricHasTaskEffectiveness(?em,?value)\\ &\Rightarrow\\ &buttonHasTextColor(?b,16777215) \end{aligned} \tag{4.5}$$

As this example demonstrates, the polisher rules allow the developer to keep improving the adapted user interfaces. This is carried out by maintaining both default and adapted interaction models which are based on the usability metrics detailed in Section 4.2.2.1.

Hey Baby, can I step into your world a while? 'Yes you can', she said, 'come on back with me for a while, we're gonna go cross the Jupiter's sands, and see all your people one by one.'

Jimi Hendrix

CHAPTER

5

Evaluation

In the previous chapters several contributions have been presented. All these contributions together define AdaptUI, a mobile platform which tackles adaptive and dynamic user interfacing for mobile devices. AdaptUI combines mobile semantics and reasoning, supported by an ontology which models user's characteristics, context current conditions and device's hardware and software characteristics. In this chapter an evaluation of the designed ontology and the semantic based platform which manages it is performed. This evaluation is addressed through several quantitative experiments regarding system performance with specific operations (e.g., reasoning over a controlled set of rules), other alternatives to adaptive user interfacing and, finally, analysing users' acceptance and feedback. Consequently, this chapter is divided in two different parts considering the experiments' goals. On one hand, an evaluation of the technical performance of the AdaptUI system is presented. This evaluation includes quantitative and qualitative experiments. It has been developed as follows:

a) First, the developed mobile adaptation of the Pellet reasoning engine is tested through several scenarios. In these scenarios different modifications of the AdaptUIOnt ontology are presented in order to evaluate the performance differences between *Pellet4Android* and Pellet when dealing with different designed ontologies.

b) Secondly, as the system's goal is to adapt the user interface, a comparison with another user interface adaptation solution has been performed: the Imhotep framework. Imhotep is a user interface adaptation framework which aims to ease the development of adaptable and more accessible user interfaces.

c) Next, several scenarios are presented in which the adaptation process of the AdaptUI system is described step by step, from the scenario conditions to the adaptation results. These scenarios have been divided into three different groups, each group aiming to cover different types of adaptation and environments. This classification includes:

   - A scenario in which the limitations suffered by the user are caused by the current context conditions.
   - A second scenario introducing several limitations due to the activities the user performs.
   - A final case in which several disabilities imped the user to interact properly with the user interface.

d) Finally, the APIs that AdaptUI provides are tested with Android developers. The participating developers have to deal with the twofold purpose of the AdaptUI API. On the one hand, developers are requested to modify the knowledge represented in a test version of the AdaptUIOnt ontology. On the other hand, they have to implement an adaptive user interface in Android.

Besides, an evaluation including actual users, without technical knowledge requisites, is performed. This part of the evaluation is based on their user satisfaction when using AdaptUI. The scenario for this experiment includes the user and an Android based tutorial through which the user calibrates several interaction parameters. Taking into account different context situations, several applications are adapted consequently. After the adaptation, the user is requested to complete a questionnaire of usability. More concretely, the SUS questionnaire is presented (among several extra questions to classify the results).
Hence, the reminder of this chapter has been structured as follows:

- A technical evaluation, whose experiments are detailed in Section 5.1. These experiments assess the reasoning performance comparing Pellet for desktop versus Pellet for mobile devices. The provided APIs have been also evaluated in an experiment including developers. AdaptUI aims to perform all the adaptations in the mobile device. Besides, the designed models have been developed using semantics, together with a set of rules which manages the process. Thus, a remarkable part of the whole AdaptUI architecture is *Pellet4Android*, the Android version of Pellet included in the Adaptation Engine (see Section 4.1.2). Hence, this part of the evaluation includes the corresponding comparison of both Pellet versions.

- A usability evaluation of the system, whose details are accordingly described in Section 5.2. This evaluation includes users interacting with the AdaptUI system.

In order to evaluate the usability of AdaptUI the SUS, a usability questionnaire, is presented to the users. Section 5.2.4 presents Table 5.30 and Table 5.31, which shows the responses from all the users participating in the evaluation of AdaptUI.

- Finally, Section 5.3 and Section 5.4 summarize and analyse the results obtained and detailed in the previous sections, regarding both technical and usability aspects.

5.1 Technical Evaluation

The technical evaluation of this chapter includes several experiments that are carried out aiming to test the performance of the adaptation process. One of the most important developed modules of AdaptUI is the Pellet reasoning engine port for Android. This module requires special emphasis in the evaluation, as leads the whole adaptation process and manages the knowledge represented through the AdaptUIOnt ontology. Besides, several other experiments are presented. Thus, in the following sections these experiments and their results are detailed:

a) First, Section 5.1.1 presents a series of experiments regarding the performance of *Pellet4Android* in comparison with Pellet for Java. These experiments include the default AdaptUIOnt ontology and different versions of AdaptUIOnt with several modifications of the axioms sets. For more details of the *Pellet4Android* mobile reasoning engine see Section 4.1.2.2.

b) Secondly, AdaptUI is compared to another user adaptation solution: Imhotep. This framework is detailed in Section 5.1.2.

c) Next, several scenarios are presented to evaluate the adaptation process of AdaptUI. Besides, a comparison with Imhotep user adaptation framework is also performed. The scenarios and their details are given in Section 5.1.3.

d) Finally, 5 developers with experience in developing Android based applications have evaluated the provided API. This experiment is detailed in Section 5.1.4.

5.1.1 Performance Evaluation: Pellet and Pellet4Android

During this section different experiments are presented in order to discuss the results of using mobile reasoning engines. In this dissertation an evaluation of the reasoning performance of *Pellet4Android*, the Android based version of Pellet ported for AdaptUI, is performed. To do so, this evaluation considers Pellet, the desktop Java based reasoning engine, to make a comparison between the performance results of both solutions. This evaluation has been divided into three different experiments:

1. The first experiment evaluates the AdaptUIOnt ontology version with both reasoning engines. When loading the ontology several rules are processed and triggered. As a result of this experiment the corresponding performance results are compared.

2. For the second experiment the set of ABox axioms of the AdaptUIOnt ontology is increased. The ABox gathers the knowledge about the individuals, including concepts and roles assertions, and individuals equality and inequality [108]. In other words, it describes the attributes of instances (or individuals), the roles between instances, and other assertions about instances regarding their class membership with the TBox concepts [1]. The ABox, RBox and TBox belong to the OWL 2 Description Logic (DL). As for these experiments several modifications have been carried out in these axiom sets, a more concrete description of the concepts represented by each set is detailed in Table 5.1. Hence, increasing the number of individuals might result into difference performance results comparing both platforms.

3. Next, the set of SWRL axioms of the AdaptUIOnt ontology is modified by increasing its amount of axioms. This axiom set collects axioms related to the rules included in the ontology. Therefore, this experiment focuses on the reasoning capabilities of each version of Pellet.

The cited experiments have been performed using two different types of devices. The ones running Pellet have been launched using a desktop environment. On the contrary, as *Pellet4Android* is an Android based version of the reasoning engine, several mobile devices have been tested. The characteristics of all the devices used for these experiments are detailed in Table 5.2.
The mobile devices shown in the previous table have not been chosen arbitrarily. As can be seen in Figure 5.1, at February 2014 the global spread of Samsung devices represent the 65% share of all Android devices.
According to the explanation of this section, in the following lines each experiment is described and evaluated. First, the default AdaptUIOnt reasoning performance comparing Pellet and *Pellet4Android* is introduced in Section 5.1.1.1. Next, these reasoning engines are tested using a modification of the default ontology, increasing the ABox axioms set. This experiment is described in Section 5.1.1.2. Finally, the AdaptUIOnt default ontology's SWRL axioms set is increased to evaluate the performance of both reasoning engines when dealing with larger sets of rules (see Section 5.1.1.3).

5.1.1.1 Using the Default AdaptUIOnt Ontology

In this experiment the default version of the AdaptUIOnt ontology is used as input of the reasoning engines in order to evaluate their performance. As AdaptUIOnt has been

ABox	TBox
Membership assertions, either as concepts or as roles. Attributes assertions. Linkages assertions that capture the above but also assert the external sources for these assignments. Consistency checking of instances. Satisfiability checks, which imply that the conditions of instance memberships are met.	Definitions of the concepts and properties of the controlled vocabulary. Declarations of concept axioms or roles. Inferencing of relationships, be they transitive, symmetric, functional or inverse to another property. Equivalence testing as to whether two classes or properties are equivalent to one another. Subsumption, which is checking whether one concept is more general than another. Satisfiability, which is the problem of checking whether a concept has been defined (is not an empty concept). Classification, which places a new concept in the proper place in a taxonomic hierarchy of concepts. Logical implication, which is whether a generic relationship is a logical consequence of the declarations in the TBox. Infer property assertions implicit through the transitive property.

Table 5.1: TBox and ABox components purposes [1].

Device	Hardware		Software
	RAM	CPU	OS
Acer TravelMate 8481	8.0	Quad-core 1.60 Intel® Core™ i5-2467M	Ubuntu 13.10 (64 bits)
Samsung Galaxy SIII Mini	1.0	Dual-core 1.0, Cortex-A9	Android 4.1.2
Samsung Galaxy SIII	1.0	Quad-core 1.4, Cortex-A9	Android 4.3
Samsung Nexus 10	2.0	Dual-core 1.7, Cortex-A15	Android 4.4.2

Table 5.2: Execution platforms software and hardware main specifications. The RAM memory is measured in GB and the CPU processor in GHz.

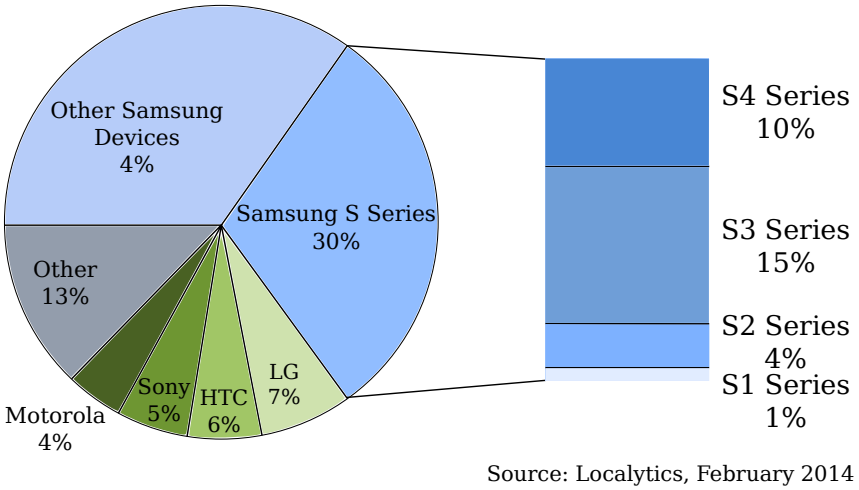

Figure 5.1: Global Android share [98]. As this pie chart illustrates, Samsung devices represent the 65% of the Android worldwide market share.

		Axioms		Results		
Device	**Triples**	**ABox**	**SWRL**	**Mean**	**Median**	**Deviation**
Acer laptop (Pellet)	2,779	37	13	0.946	0.951	0.017
Galaxy SIII Mini (*Pellet4Android*)	2,779	37	13	2.764	2.737	0.127
Galaxy SIII (*Pellet4Android*)	2,779	37	13	1.649	1.652	0.076
Nexus 10 (*Pellet4Android*)	2,779	37	13	5.147	5.122	0.205

Table 5.3: Pellet and *Pellet4Android* comparison loading the default AdaptUIOnt ontology.

designed to be a lightweight ontology to be available to be used with mobile reasoning engines, this experiment shows how *Pellet4Android* performs in comparison with the results obtained by Pellet in a desktop environment. These results are shown in Table 5.3.
Surprisingly Table 5.3 reveals that the Nexus 10 performs worse than the other two mobile devices. This is usually due to the available and allocated memory that Java estimates for launching the required reasoning tasks. Table 5.4 for instance shows a more usual behaviour of the Nexus 10, which matches its hardware specifications.
Figure 5.2 illustrates the differences of using Pellet or *Pellet4Android* when loading the default ABox and SWRL axiom sets in AdaptUIOnt. As shown in Figure 5.2 and in Table 5.3, the performance of Pellet for Java environments is better in any case if we compare it with the Android based version. However, and although there are several differences depending on the used mobile device, the differences are small. This is specially remarkable in the case of the Samsung Galaxy SIII. This device performs a remarkable 1.649 seconds, which is just approximately 0.7 seconds slower than Pellet for Java.

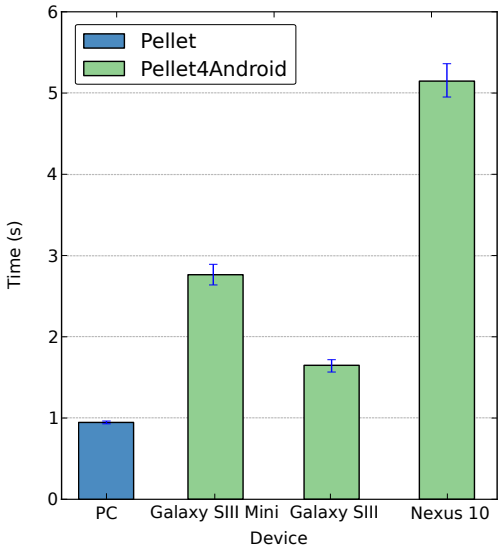

Figure 5.2: Pellet and *Pellet4Android* performance comparison using the default AdaptUIOnt ontology. See Table 5.3.

5.1.1.2 Incrementing the ABox Axioms Set

In this case the experiment consists in incrementing the ABox axioms set of the AdaptUIOnt ontology. As stated before, the ABox describes the attributes of instances, the roles between instances, and other assertions about instances regarding their class membership with the TBox concepts [1]. The modification of the ABox has been carried out by incrementing the amount of instances in 5,000, 10,000, 15,000 and finally reaching 20,000 instances. Thus, with this experiment we want to evaluate how increasing the number of individuals might result into a performance penalization, specially in the case of *Pellet4Android*. Table 5.4 shows the results of this experiment.

Figure 5.3 illustrates the differences between Pellet and *Pellet4Android* when loading the AdaptUIOnt ontology with an increment applied to the ABox axioms set. As shown in the chart (accordingly with Table 5.4) the performance of Pellet for Java environments keeps being better in comparison with the Android based version. While Pellet maintains its performance under 5 seconds, Android devices require much more time to perform the same reasoning. In fact, specially the Samsung Galaxy SIII Mini mobile device requires more than 180 seconds to evaluate the corresponding 20,000 instances, while the Samsung Galaxy SIII needs approximately 97 seconds and the Nexus 10 not more than 32.

5.1.1.3 Incrementing the SWRL Axioms Set

Finally, this last experiment consists on incrementing the SWRL axioms set of the AdaptUIOnt ontology, which collects the axioms related to the rules included in the on-

Device (Reasoner)	Axioms			Results		
	Triples	ABox	SWRL	Mean	Median	Deviation
Acer laptop	12,779	5,000	13	3.014	3.012	0.034
(Pellet)	22,779	10,000	13	3.903	3.905	0.052
	32,779	15,000	13	4.228	4.231	0.036
	42,779	20,000	13	4.539	4.541	0.042
Galaxy SIII Mini	12,779	5,000	13	59.412	59.508	0.708
(*Pellet4Android*)	22,779	10,000	13	30.321	30.327	0.347
	32,779	15,000	13	87.957	87.018	1.108
	42,779	20,000	13	183.882	183.879	2.101
Galaxy SIII	12,779	5,000	13	36.336	36.194	0.668
(*Pellet4Android*)	22,779	10,000	13	16.471	16.439	0.288
	32,779	15,000	13	45.387	45.593	0.729
	42,779	20,000	13	97.151	97.440	1.120
Nexus 10	12,779	5,000	13	14.171	14.428	0.525
(*Pellet4Android*)	22,779	10,000	13	9.024	9.065	0.291
	32,779	15,000	13	17.944	17.969	0.496
	42,779	20,000	13	32.070	32.019	0.588

Table 5.4: Pellet and *Pellet4Android* comparison loading the AdaptUIOnt ontology with an increment in the ABox axiom set.

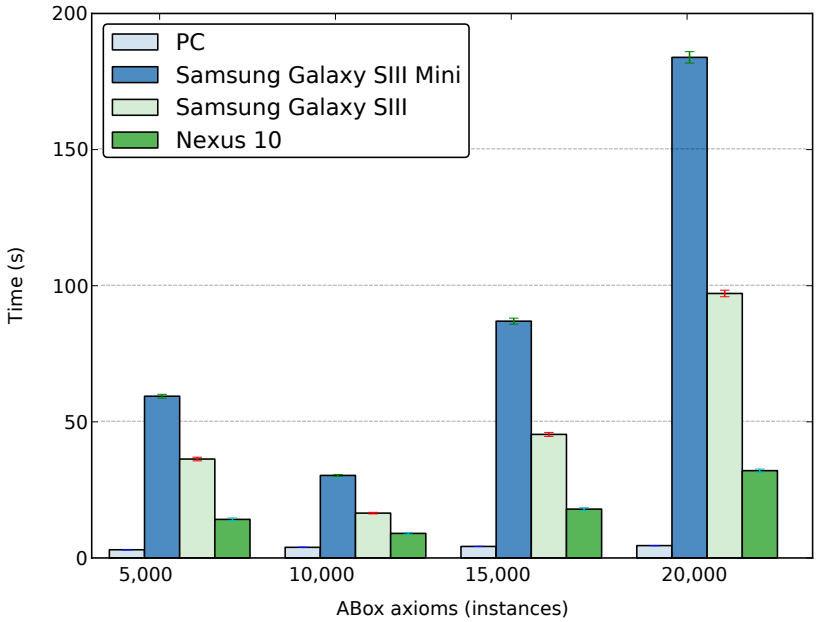

Figure 5.3: Pellet and *Pellet4Android* performance comparison using the AdaptUIOnt ontology increasing the ABox axioms set. See Table 5.4.

		Axioms		Results		
Device	**Triples**	**ABox**	**SWRL**	**Mean**	**Median**	**Deviation**
Acer laptop	82,779	37	5,013	4.770	4.732	0.141
(Pellet)	162,779	37	10,013	6.327	6.296	0.164
	242,779	37	15,013	7.427	7.194	0.444
	322,779	37	20,013	8.147	8.117	0.105
Galaxy SIII Mini	82,779	37	5,013	96.878	96.988	0.109
(*Pellet4Android*)	162,779	37	10,013	101.656	101.899	0.322
	242,779	37	15,013	243.981	244.011	0.298
	322,779	37	20,013	331.433	331.894	0.110
Galaxy SIII	82,779	37	5,013	84.121	84.121	0.869
(*Pellet4Android*)	162,779	37	10,013	74.248	74.103	0.250
	242,779	37	15,013	209.431	208.628	1.699
	322,779	37	20,013	216.005	216.077	1.202
Nexus 10	82,779	37	5,013	22.317	22.471	0.333
(*Pellet4Android*)	162,779	37	10,013	45.193	44.736	1.312
	242,779	37	15,013	85.543	88.134	5.490
	322,779	37	20,013	107.151	106.131	2.749

Table 5.5: Pellet and *Pellet4Android* comparison loading the AdaptUIOnt ontology with an increment in the SWRL axiom set.

tology. Using an amount of 5,000, 10,000, 15,000 and finally 20,000 rules we aim to evaluate how increasing the number of rules penalize the performance of Pellet, specially in the case of *Pellet4Android*. Table 5.5 shows the results of this experiment.
Figure 5.4 illustrates the differences between Pellet and *Pellet4Android* when running the different sets of SWRL axioms of the AdaptUIOnt ontology. As shown in the chart, it takes more time to evaluate the set of rules than the instances. In fact, the scale of the Y axis reaches 350 seconds, while in Figure 5.3 the maximum mean value reaches less than 200 seconds.

5.1.1.4 **Discussion**

During these experiments, and as shown in Table 5.3, Table 5.4 and Table 5.5, neither the TBox or RBox axioms sets have been modified. These collections belong to what is called the terminology knowledge in OWL 2, and it does not affect the performance of the reasoning process. Hence, the TBox and the RBox contain 266 and 22 axioms respectively during the experiments.
On the contrary, sets of 5,000, 10,000, 15,000 and 20,000 axioms have been added to the ABox and SWRL axioms sets to demonstrate the performance penalization when using large axioms sets with the tested reasoning engine. These modifications of the AdaptUIOnt ontology instance set have been performed through several Python scripts, which have modified the corresponding default *adaptui.owl* file.

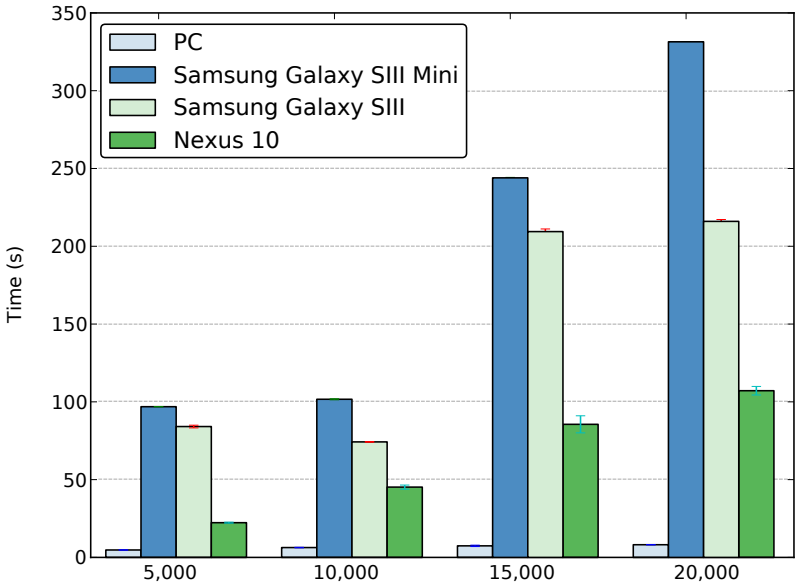

Figure 5.4: Pellet and *Pellet4Android* performance comparison using the AdaptUIOnt ontology increasing the SWRL axioms set. See Table 5.5.

5.1.1.5 Conclusions

These results demonstrate that, although managing semantics in mobile devices is possible, they are still far from the results obtained in a desktop environment when dealing with complex reasoning tasks. Nonetheless, *Pellet4Android* is a good approximation of a reasoner to run on mobile devices. A native Pellet for Android port written in C++ would probably improve these results. But one of the AdaptUIOnt ontology's benefits is its lightness. Using less than 400 axioms, this ontology is able to model a full adaptation domain, considering the user and his/her capabilities, the surrounding environment and the device characteristics. The results shown in Table 5.3 show how *Pellet4Android* responses properly when dealing with the AdaptUIOnt ontology. However, more efforts in mobile reasoners would benefit these systems.

Regarding the ABox collection of axioms, the first conclusion that is extracted from the results shown in Table 5.4 is that Pellet running in a PC has a very optimal response. For example, increasing the number of instances to almost 20,000 instances just reduces the response time in around 4 seconds (see Figure 5.3). Increasing the ABox slows down the final performance, incrementing the number of seconds for loading the ontology. On the contrary, running this experiment in the mobile devices reveals a lack of efficiency. First, each device's hardware capabilities are taken into account. The first mobile device, the Samsung Galaxy SIII Mini needs around 2.764 seconds for loading the default AdaptUI ontology, which consists of 37 ABox and 13 SWRL axioms. This same case takes 1.649

seconds in the Samsung Galaxy SIII and 5.147 seconds in the Nexus 10. Although these figures might appear to be enough considering we are dealing with limited hardware, by increasing them in the same way that it has been done with the Acer laptop results into unmanageable time responses. For instance, loading 5,032 ABox axioms takes 3.014 seconds for the laptop and Pellet, for the Samsung Galaxy SIII Mini it takes 59.412 seconds, 36.336 seconds for the Samsung Galaxy SIII and 14.171 seconds for the Nexus 10.
Nevertheless, these differences are more significant when dealing with the SWRL axioms set. In this case, although Pellet's performance does not exceed 9 seconds, the differences with the ABox axioms set are much bigger regarding *Pellet4Android*. In the best case, the Nexus 10 needs around 22 seconds to reason over 5,000 rules, while the Samsung Galaxy SIII Mini needs more than 90. This depicts the existing differences of performance depending on the Android device we choose.
As a final conclusion, taking into account the presented results, *Pellet4Android* presents promising performance results when dealing with the AdaptUIOnt. The lightness of the ontology (considering the amount of axioms) assures a more than acceptable performance.

5.1.2 Comparing AdaptUI with other AUI Solutions

As a second part of the technical evaluation, in this section a comparison between AdaptUI and Imhotep is performed. The purpose of this section is to demonstrate that the capabilities of the AdaptUI platform improve the ones provided by the Imhotep framework. Hence, this section is organized as follows: First, in Section 5.1.2.1 the Imhotep framework and its main characteristics is introduced. Next, a use case using both adaptation platforms is detailed (see Section 5.1.2.2). Finally, in Section 5.1.2.3 and Section 5.1.2.4 a discussion and several conclusions are presented.

5.1.2.1 Imhotep

As mentioned in Section 1.1.1, Imhotep [49] is a framework developed and maintained at the University of Deusto which aims to ease the development of adaptable and more accessible user interfaces. The framework allows to develop applications without considering a specific design for the user interface, as it is built from the definition of a series of preprocessor directives which evaluate a user's and device's profile.
In order to fully understand how Imhotep works, Figure 5.5 illustrates its architecture design. As can be seen in the figure, Imhotep is divided into 4 modules:

- The first one is the Application Downloader, which is a tool installed in the user's device to connect to the repository and download adapted applications.

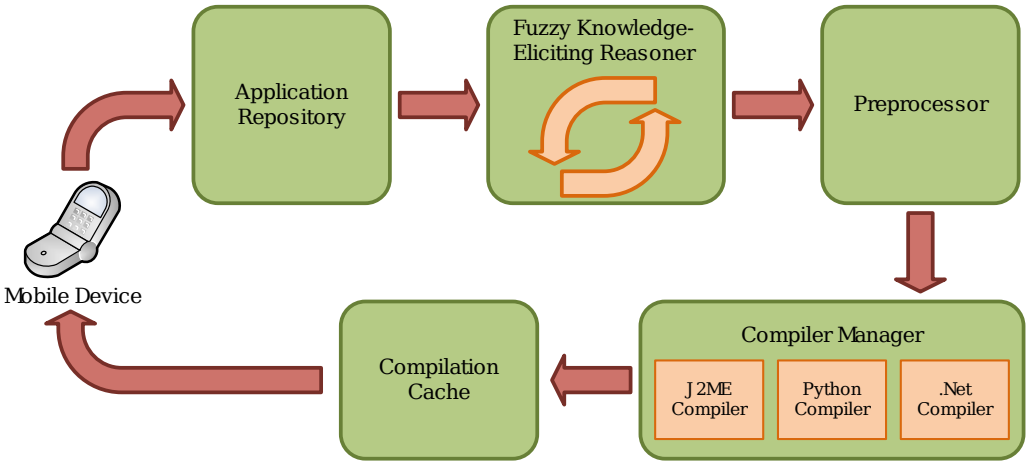

Figure 5.5: The Imhotep architecture [13].

- The Representational state transfer (REST) server contains the application repository. Once a user selects a desired application, the server will:
  - Use the Fuzzy Knowledge-Eliciting Reasoner to infer new values with the user and device configuration and the variables and rules established by the developer of the application.
  - Call the Preprocessor to select only the code that the final user requires.
  - Delegate the compilation of the application to the Compiler Manager. Currently the REST server only supports Android, but other compilers could be easily added.
  - Store the compiled application in the Compilation Cache, so future requests will not require to pass through the whole process if they have the same configuration values.
- The Wizard: developers use the wizard to establish the variables and the possible values of those variables. It uses the trends database to show the different values given a concrete user and a concrete device.
- The Capacity Tester. Users have to create a user profile with their capabilities. The Capacity Tester will gather those capabilities by testing them. The Current application is just a proof of concept of how the Capacity Tester should work.

The preprocessor directives define how the final source code must be generated, providing conditions for certain regions of code to be added or skipped and adding the corresponding UI variables that the preprocessor will adapt for each compilation. The preprocessor identifies the directives when they start by //# in languages that support inline comments

starting by //, such as Java, C# or C++, #// in languages that support inline comments starting by #, such as Python or Perl, and '// in VB.NET.
The preprocessor can avoid the compilation of fragments of code if certain conditions are met. These conditions can include calls to functions provided by the system. Basic string and *math* functions are available, including *lowercase*, *trim*, *contains*, *round* or *sqrt*, as well as functions to check if a certain variable is available. The conditions can be embedded, as shown in the code below. The syntax of the conditions is based on the syntax used by the Python programming language. The following code shows an example of using preprocessor directives in Java. During compiling time this code will be analysed by the framework taking into account the defined variables and their values. Therefore, the final binaries will contain just one reference to one of the methods shown in Listing 14.

```
1  //#if defined(${application.devices.height})
2    //#if ${devices.height} > 150 * 2
3      addTenRows();
4    //#elif ${devices.height} > 200
5      addTenCols();
6    //#else
7      addFiveRowsAndCols();
8    //#endif
9  //#else
10   addFiveRowsAndCols();
11 //#endif
```

Listing 14: Using the Imhotep framework through preprocessor directives [13].

Developers ask for user and device capabilities in these conditions. For example, one directive could state that if the user is blind the application should use a voice based interface. Five categories of user capabilities were defined (see Figure 5.6).
Now that the Imhotep framework has been introduced, a comparison between AdaptUI and Imhotep running the same experiments is presented. Despite the fact that both solutions have the same purpose (to help reducing the boundaries for those who cannot properly interact with a user interface), their differences are substantial. Imhotep directly depends on a server (allocated in the University of Deusto) while AdaptUI runs fully in the mobile phone, performing the adaptation on the fly.

5.1.2.2 Use Case: AssistedCity

As said before, Imhotep is a framework for developers which provides a preprocessor directive based procedure to build adaptive user interfaces. The Imhotep framework was

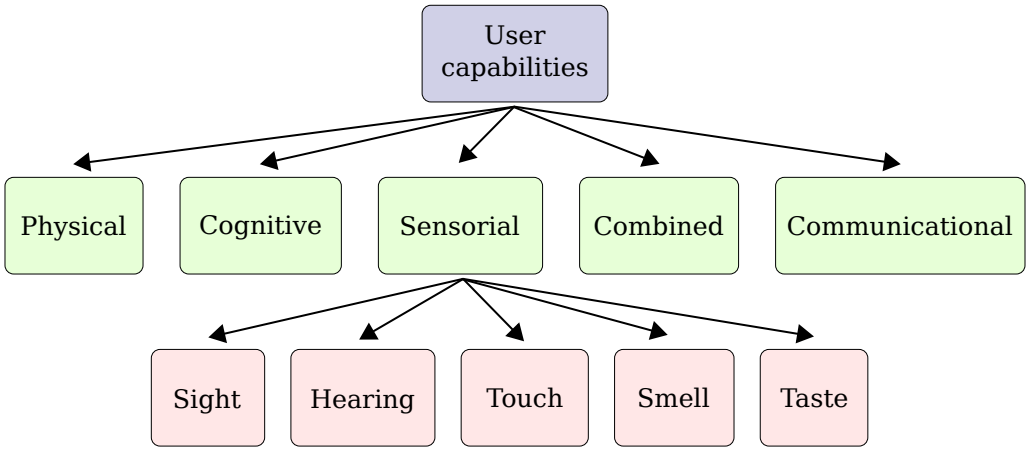

Figure 5.6: The set of Imhotep's user capabilities [13]. As is shown, user capabilities are classified into 5 different groups: physical, relative to user's motor skills; cognitive, which deals with memory and comprehension capabilities; sensorial, including capabilities related to sight, hearing, touch, smell and tast; combined, including combination of different disabilities; and communicational, which deals with speech.

originally tested with AssistedCity[1]. AssistedCity is a tour guide application which guides the user taking into account several points of interest categories (i.e., monuments, museums, restaurants, and so forth) that the user needs to choose from a menu. Then, using the camera and through an Augmented Reality interface the application guides the user to the desired point of interest. As a demonstrator of the Imhotep framework, this application was developed using several preprocessor directives which take into account sight sensory capabilities, hearing capabilities and device's screen size, to adapt the user interface of the application. Hence, we have chosen the same use case to evaluate the results of the adaptation using the corresponding platform.

This evaluation is split in two different parts. First, a visual comparison of the resulting adaptations is shown. Then, we focus on the performance of both solutions, specially on the required time to get the whole processes finished.

For the adaptation using Imhotep, an extra application is needed to send the user's and device's characteristics to the Imhotep server. Listing 15 shows the variables file used by Imhotep to perform the corresponding adaptation. This file is completed by the user when he/she uses the cited application. Once the variables file is completed, the user is able to download the desired provided application from the Imhotep server.

[1] `https://github.com/aitoralmeida/imhotep/tree/master/PiramideRestServer/deploy/piramide/applications/AssistedCity/1.0.0.0/project`

Entity	Class	Data property	value
User	*Display*	*userDisplayBrightnessIsStatic*	*false*
(UserCharacteristics)		*userDisplayIsApplicable*	*false*
	Audio	*userDisplayApplicableIsStatic*	*false*
		userAudioHasApplicable	*true*
		userAudioApplicableIsStatic	*false*
		userAudioHasVolume	5
	Interface	*userInterfaceInput*	*haptic*
		userInterfaceOutput	*default*
	Experience	*userHasExperience*	*high*
	View	*userViewIsStatic*	*false*
	Other	*userHasLanguage*	*English*
	vibration	*true*	
Device	*DeviceScreenResolution*	*deviceHasScreenResolution*	1280 x 720
(DeviceCharacteristics)	*OS*	*deviceHasOSVersion*	4.3

Table 5.6: User and device characteristics representation in AdaptUIOnt.

```
{
  "imhotep.devices.screen.height": 1280,
  "imhotep.devices.screen.width": 720,
  "imhotep.devices.modelname": "Samsung Galaxy SIII",
  "imhotep.devices.os": "Android",
  "imhotep.devices.os.version": "4.3",

  "imhotep.user.capabilities.problems": true,
  "imhotep.user.capabilities.problems.sight": true,
  "imhotep.user.capabilities.problems.sight.diopters": 10,
  "imhotep.user.capabilities.problems.hearing": false,
  "imhotep.user.capabilities.problems.smell": false,
  "imhotep.user.capabilities.problems.touch": false,
  "imhotep.user.capabilities.problems.problems": false
}
```

Listing 15: The variables file, in which device characteristics and user capabilities are described.

On the contrary, the corresponding AdaptUI models for the user and the device are represented through the AdaptUIOnt ontology, and there is no need for external servers. The similar user and device situation is represented as follows, in Table 5.6:

The results of the adaptations are illustrated in the following figures. First, Figure 5.7 shows the default user interface for the main menus of the application. It originally consists of several visual icons representing different venues categories (i.e., bars, restaurants, cafés, and so on). Once the user chooses one option, a list with the nearer corresponding

Figure 5.7: AssistedCity default menus and user interface.

Figure 5.8: AssistedCity adapted by Imhotep (left) and AdaptUI (right) taking into account the corresponding inputs shown in Listing 15 and Table 5.6.

type of venues is presented.

Next, a comparison between the two adaptations is illustrated through Figure 5.8. On the left the adaptation performed by Imhotep is shown. The parameters detailed in Listing 15 for the user and device characteristics are sent to the Imhotep server. Then, the corresponding application is generated and send back to the user. On the contrary, on the right side of Figure 5.8 there is the adaptation performed by AdaptUI. Considering the input detailed in Table 5.6 AdaptUI performs the corresponding adaptation (depending on the set of specified rules). The adaptation performed by AdaptUI matches the input capabilities shown in Table 5.6, which represents better the user capabilities than the specification given by Listing 15.

Regarding the time performance evaluation in Table 5.7, it shows the time required by Imhotep to adapt AssistedCity. As seen through this table, if the server maintains a cache of the already adapted applications for the specific user and device the response of the system does not exceed 1.3 seconds.

Device	Cached data			Non cached data		
	Mean	Median	Deviation	Mean	Median	Deviation
Galaxy SIII Mini	1,257	1,237	0,247	15,125	15,034	0,445
Galaxy SIII	1,175	1,107	0,239	15,094	14,950	0,501
Nexus 10	1,211	1,184	0,180	15,115	15,033	0,492

Table 5.7: Imhotep framework time analysis. The figures under Mean, Median and standard deviation (Std. deviation) are represented in seconds.

System	Platform	Cache/Trigger	Mean
Imhotep	Galaxy SIII Mini	Cached	1.257
		Not cached	15.125
	Galaxy SIII	Cached	1.211
		Not cached	15.115
	Nexus 10	Cached	1.175
		Not cached	15.094
AdaptUI	Galaxy SIII Mini	Context change	1.899
	Galaxy SIII	Context change	1.544
	Nexus 10	Context change	1.213

Table 5.8: Comparing Imhotep and AdaptUI time performance. The mean is represented in seconds.

On the contrary, AdaptUI does not require an external server, and it does not use any cache. Thus, the required time for the adaptation depends mostly on the hardware of the device running the reasoning process. Table 5.8 compares the means of using Imhotep or AdaptUI for an adaptation. In the case of AdaptUI, a light change in context triggers the process. Obviously, this represents one of the most significant benefits of AdaptUI, as the platform is able to dynamically react to these changes. On the contrary, a user of AssistedCity will need to indicate the new situation in the variables file, and the reasoning process will start again in the Imhotep server.

5.1.2.3 **Discussion**

Imhotep was conceived in 2010, when the hardware characteristics of the available Android devices was still limited. First 1 GHz processors started to power these devices, and their RAM memory vaguely exceeded 500 Megabyte (MB). Considering these limitations, Imhotep was designed following the same approach that several solutions cited in Chapter 2 used: externalizing the computational complex processes to an external server. Hence, in Imhotep the developers have to use several available preprocessor directives where the user interface alternative code has to be included. Then, their source code is uploaded to an external server, which would generate the corresponding binaries (according to user's requests) with the corresponding adaptation. These requests include the

characteristics of the mobile device and the user's profile.
Although Imhotep was a good approximation as a tool for developers to write applications with adaptive user interfaces, we have found several problems in it:

- Regarding final users, *Imhotep requires an extra tool for downloading the required application* with the corresponding and adapted user interface. This means that every time the user requires a different adaptation the profile configuration process is launched. Thus, if there is no cached data in the server, it will cause a significant delay in the adaptation process.

- Besides, *Imhotep depends on Internet connection* for for the adaptation, which might be problematic. As an external server is needed to perform the adaptive operations and source code compiling, there is the possibility of a server or network failure, which would obviously affect or even cancel the whole process.

- *Context is not taken into account for the adaptation*. Only the user and the characteristics of the current device were considered. Several examples of the benefits of including context for the adaptation process have been presented during this section.

On the contrary, AdaptUI was conceived during 2013, when multi-core kernels and CPUs were already deployed into mobile devices. These new characteristics increased their capabilities, making possible more complex processing. Therefore, AdaptUI first designs were definitely based on Android. This brings the whole adaptation process to the mobile. This means that, on the one hand, there is the advantage of not depending on the network or an external server to process anything. But it also means that the workload has to be managed by a hardware not as powerful or capable as a computer's one. As also detailed in Table 5.8, the adaptation performance relies on the each device's hardware capabilities. Besides, AdaptUI includes the current context situation in the equation for the user interface adaptation. This fact presents several challenges and possibilities, as is considered in AdaptUI as a trigger for different situations involving the user and the device.

5.1.2.4 **Conclusions**

During this section a comparison between AdaptUI and Imhotep has been performed. Imhotep is a framework for easing the adaptation of user interfaces. In the presented figures and tables both solutions evaluation has been illustrated. The resulting adapted interfaces and the corresponding performance of both platforms have been shown. Next, we summarize our experiences through this evaluation with several considerations in the following lines.
As Imhotep needs an external server for the complex processing of the user interface adaptations, maintaining a cache is vital for this system. On the contrary, and as shown

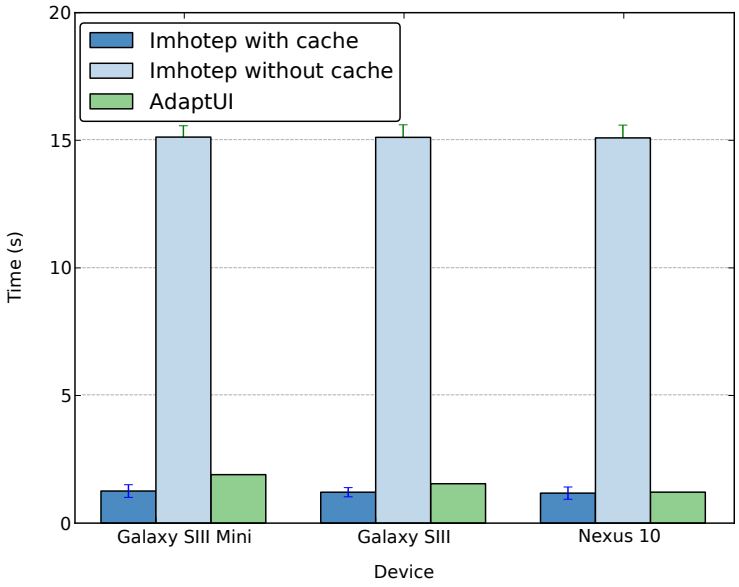

Figure 5.9: Receiving the corresponding adapted user interface for different devices using Imhotep without cache. See Table 5.7 and Table 5.8.

in Table 5.7, the response time from rebuilding the whole source code results in a 15 seconds delay. Comparing with the cached results, which imply less than 1.5 seconds to perform the same operation is clear that the difference is significant. Nevertheless, one of the benefits of this solution is that, as the adaptation process is delegated to an external server, is not important which device is being used by the user. The current device has just to be compatible with the applications available in the server.

One of the main problems of Imhotep is that context is not taken into account for the adaptations. The explanation for this decision was based on the time limitation of Imhotep for performing a complete adaptation. Figure 5.9 illustrates how operating without cached data takes around 15 seconds for receiving the corresponding user interface in the device. This means that the context might change during this period of time. On the other side, the cache was conceived to store information about the profile of the user and the device that might not change during time. By using this cache the adaptations were sent back to the device in less than 2 seconds.

Imhotep time responses are obtained as the result of the addition of three different processes. The first one is the required time in terms of networking, that takes to send the adaptation request from the device to the adaptation server. Secondly, the required time to process the request, compile and generate the adapted version (around 14 seconds). And finally, the time needed to send the adaptation back to the user's device with the corresponding user interface.

On the other hand, AdaptUI performance does not depend on the network traffic, as it

runs fully in the mobile device. Thus, the obtained performance results depend only on the hardware and software capabilities of the device. The triggers are launched by the user for Imhotep and by a context change for AdaptUI respectively. This is shown in Table 5.8.
Regarding the presented arguments, we conclude that:

a) Imhotep's performance when using a cache is similar (even better) when network conditions are optimal.

b) As Imhotep does not carry out complex operations in the mobile devices, leaving this workload to an external server, the hardware specifications of the current device are not as important as in AdaptUI.

c) On the contrary, this dependency carries several drawbacks. For example, a network failure would leave the adaptation process incomplete.

d) Besides, although working with a cache results in similar time responses than AdaptUI, first adaptations of each application will cost a high time to be performed. During this time (around 15 seconds under normal circumstances) the context of the user might change, and it is possible for the user to reject the adaptation.

e) As AdaptUI takes the context into account, the adapted solutions are more characterized for the current user situation.

f) Although depending on the device's hardware, it might result into slower adaptations, managing small semantic models (as AdaptUIOnt) is still manipulable by the platform. Besides, as the smartphones market grows in terms of hardware capabilities, the performance of such systems will be increased.

g) AdaptUI does not consider explicit physiological user capabilities. On the contrary, Imhotep works with a user model in which it is necessary to provide several physiological based capabilities. This results into a non practical solution, as we lack medical knowledge.

h) Another benefit from AdaptUI is the automation of the adaptation through several sets of adaptation rules. These rules, not considering explicit physiological user capabilities, infer knowledge about the user that is vital for the adaptation. The whole set of rules that manages the adaptation process in AdaptUI are detailed in Section 3.2.

5.1.3 Scenarios

In the previous section several quantitative experiments have been presented. The experiments have illustrated the performance differences considering the required time to manage the AdaptUIOnt ontology and Imhotep. Regarding the ontology itself, AdaptUI takes into account many context situations, user's capabilities and device's characteristics. This makes difficult to evaluate every one of these settings in real environments. In this section, several hypothetical scenarios are presented simulating the behaviour of the analysed adaptive systems in more complex situations.
The scenarios are included in the evaluation of AdaptUI compared to Imhotep. Consequently, both adaptation processes are taken into account for the resulting adaptation and the presented scenario conditions. Hence, the scenarios represent three stereotypical situations which might require a user interface adaptation:

- First, in the first scenario the situation is characterized by several context parameters that limit certain user capabilities. Consequently, AdaptUI would have to consider the user as an updated version of himself/herself, taking into account the new set of capabilities that he/she has due to the context situation.
- In the second scenario, a situation in which the user capabilities are delimited by a set of tasks that form an activity is presented. The user is impeded to act normally. Therefore, the AdaptUI platform would have to be aware of this limitation.
- The last scenario simulates a situation where the user suffers from a certain disability, so the platform should adapt the user interface accordingly.

Each scenario is presented as follows:

- First, an introduction to the problem to be evaluated is described.
- Secondly, the scenario situation is presented. It is also summarized with a table indicating the main characteristics of the three main entities.
- Thirdly, the adaptation process by Imhotep and AdaptUI is detailed.
- Finally, each scenario presents several conclusions after a brief discussion.

5.1.3.1 Scenario 1: Limitations Caused by Context Conditions

This scenario introduces several limitations that context might induce to certain user capabilities. As context is considered as a significant entity in AdaptUI, this scenario aims to solve a situation in which it impedes the user to interact properly. On the contrary, Imhotep does not include the context situation for adapting the user interface. Thus, it

	Scenario 1
User	
- Personal data	John, 36 years old, Finnish
- Activity	Sending a SMS
- Known disabilities	None
Context	
- Location	Relative: Rovaniemi, Finland
	Absolute: 66.497109, 25.724977
- Time	18:35
- Brightness	600 lx
- Temperature	-10 ^{o}C
Device	Samsung Galaxy SIII
	Battery: 85%

Table 5.9: Scenario 1 situation summary.

Class	**Scenario 1**	
	Data property	**value**
Display	*userDisplayBrightnessIsStatic**	true
	userDisplayIsApplicable	true
Audio	*userDisplayApplicableIsStatic*	false
	userAudioHasApplicable	true
	userAudioApplicableIsStatic	false
	userAudioHasVolume	4
Interface	*userInterfaceInput*	haptic
	userInterfaceOutput	default
Experience	*userHasExperience*	high
View	*userViewIsStatic*	false
Other	*userHasLanguage*	Finnish
	vibration	true

Table 5.10: User profile for Scenario 1.

would not be possible to have a coherent result regarding the proposed scenario. The following lines introduce the cited scenario:
John is going home after work. He lives and works in Rovaniemi, one of the northernmost and coldest regions in Finland. As can be seen in Table 5.9, John does not suffer from any disability. He removes his gloves and then he proceeds to send a Short Message Service (SMS) to his wife. The current temperature is -10 ^{o}C.
AdaptUI covers this situation through the following steps. First, the platform considers through the user model that John does not suffer from any disabilities. Analysed by the Capabilities Collector, AdaptUI does not find disabilities that would limit future adaptations. Table 5.10 shows the semantic representation of the model, carried out by the Semantic Modeller, that fits John's profile.

Scenario 1	
UserAux	**Value**
textithasDisplayApplicable	true
textithasDisplayBrightness	false
textithasRestriction	hands_restriction

Table 5.11: UserAux class generated by the pre-adaptation rules set and resulting user interface first adaptation for both scenarios.

Regarding John's model, it can be seen how it is configured as an open model for future adaptations. Besides, most of the preferences are configured as default. The only property that is limited from the model is the brightness, which John has established as a static property. This means that AdaptUI understands that this property should not be adapted no matter what the context situation is. As the contrast and the volume are not static, the modelled integer values are considered as preference values and not a limitation for the rules. This means that John prefers a volume level of 4, but he allows the system to manage it if certain conditions are met. These integer values are different depending on the used platform. In this case, as these experiments run over the Android platform, they refer to Android values [5].
Table 5.10 shows the semantic representation of this model which fits John's profile, using the AdaptUI ontology detailed in Chapter 3. The asterisk marks a property that is marked by the user in a way that limits the adaptation. In this case, the *userDisplayBrightnessIsStatic** is configured as *true* during the interaction between the user and the Capabilities Collector. Thus, although the context light might affect the sight capability of the user, the display brightness will not be adapted. Combining the user model from Table 5.10 with the current context situation depicted in Table 5.9, the pre-adaptation rules generate the *UserAux* profile for John, which is shown in Table 5.11. In this case, there is a temporary restriction for the user due to the current context conditions (freezing temperature) that is modelled through the *hasRestriction* datatype property (with the value *hands_restriction*). The final adaptation for this situation is driven by the adaptation rules set. The brightness is not adapted because the property *userDisplayBrightnessIsStatic* from the Brightness class is configured as *true* (see Table 5.10). Besides, an increase of 10 pixels of the View size is needed to cover the situation in which the user makes more errors on the touch screen because of the freezing temperature. An extra interaction feedback is added (vibration). The final adaptation is shown in Table 5.12:
Finally, the system should provide the user a chance to evaluate, provide feedback, on whether the interaction with the presented adaptation is usable enough or if another adaptation should be triggered. There are also a few rules which take into account the device's dynamic capabilities. The remaining memory, available processor and battery are monitored in order to avoid adaptations that could freeze the device or consume the remaining

Scenario 1	
Adaptation	**Value**
hasBrightness	-
hasColourSet	-
hasViewSize	10
hasResponse	vibration

Table 5.12: Final adaptation for the Scenario 1.

Battery (%)	**Ontology values**
$x \leq 10$	*not_sufficient*
x $10 < x \leq 50$	*sufficient*
x $50 < x \leq 100$	*optimal*

Table 5.13: Battery percentage and corresponding ontology values.

power. As is shown in Table 5.13, battery levels under 10% are considered too low to perform an adaptation.

5.1.3.1.1 Discussion

This first scenario presents a situation in which the user capabilities are limited by certain context characteristics. More specifically, the current temperature (-10 ^{o}C) might make difficult for John to use his hands and properly interact with his device. In this case, thanks to the context modelling availability in AdaptUIOnt and due to the configuration of the user profile, several sets of rules lead to the results of Table 5.11, in which we can see how the pre-adaptation rules evaluate several characteristics of the user conditions, and the results of Table 5.12, in which the final adaptation is shown.
On the contrary, Imhotep does not take context conditions into account. The only way to indicate a change on context would be if the user configures the Imhotep user profile with a concrete disability. This would lead to a series of configurations each time the user is within a different environment.

5.1.3.1.2 Conclusions

John is perfectly healthy and he usually does not need complex adaptations. However, in this case the current freezing temperature makes difficult for him to use his fingers with the usual precision. Considering these premises, any adaptive system would generally ignore

Entity	Capability or characteristic	Solution	
		Imhotep	AdaptUI
User	Sight		
	Hearing		
	Other		✓
	Activity		
Context	Light		
	Noise		
	Temperature		✓
Device	Resolution	✓	✓
	Battery		✓

Table 5.14: AdaptUI and Imhotep comparison of the final reached adaptation for the Scenario 1.

the current situation (including Imhotep). Nevertheless, with these weather conditions, John is partially disabled. The weather, more concretely the temperature, might affect several physiological capabilities of the user. Therefore, the interaction between John and the SMS application might experience several difficulties.
Imhotep does not take into account the context situation, but we can consider the new set of capabilities shown in Table 5.10 as the starting user capabilities. Besides, regarding Table 5.12 where the adaptation from AdaptUI is shown, Imhotep would not make an adequate adaptation. This is not just because the user has not a permanent disability. Imhotep considers through the user profile sight problems and hearing problems. Considering that the profile is correctly configured, an adaptation in such ways will not cover the user needs, which deals with mobility and precision using fingers on the screen. Table 5.14 shows several significant differences between those characteristics that Imhotep is able to take into account in this scenario and the ones that AdaptUI considers. It is shown how Imhotep unawareness of context features makes it inaccurate when adapting the user interface to the current situation. Besides, it is not able to model an activity, but the consequences of it (as AdaptUI, which takes into account that the user is performing an activity that, due to the context conditions, cannot carry out properly).

5.1.3.2 **Scenario 2: Limitations Caused by Activities**

In this case, the Scenario 2 presents a situation in which the user cannot act properly due to the activity or activities that he/she is doing. Again, Imhotep cannot characterize activities. Nevertheless, in this scenario user disabilities and device characteristics are going to be used in Imhotep to follow the corresponding user interface adaptation. The Scenario 2 is described now in the following lines:
Karen is a 60 years old woman with several disabilities due to her ageing. She is colour blind, which means that she has problems distinguishing several colours. In addition, she

	Scenario 2
User	
- Personal data	Karen, 60 years old, German
- Activity	Driving
- Known disabilities	Colour blindness
	Hearing loss
Context	
- Location	Relative: Berlin, Germany
- Time	11:10
- Brightness	1,100 lx
- Temperature	15 ^{o}C
Device	Samsung Galaxy Tab

Table 5.15: Scenario 2 situation summary.

Class	**Scenario 2**	
	Data property	**value**
Display	*userDisplayBrightnessIsStatic*	false
	userDisplayIsApplicable	true
Audio	*userDisplayApplicableIsStatic*	true
	userAudioHasApplicable	true
	userAudioApplicableIsStatic	false
	userAudioHasVolume	4
Interface	*userInterfaceInput*	default
	userInterfaceOutput	default
Experience	*userHasExperience*	low
View	*userViewIsStatic*	false
Other	*userHasLanguage*	German
	vibration	true

Table 5.16: User profile for Scenario 2.

suffers from a light hearing loss, which is not severe. She is driving to down town using a Global Positioning System (GPS) mobile application. For this scenario, the most significant information to infer is the impossibility to use the hands and the impossibility to distract her, as Karen is driving. Besides, the user model should consider the capabilities cited in Table 5.15, as she is colour blind. Therefore, with this information a first user model is semantically represented by AdaptUI in Table 5.16. The following lines analyse the adaptation process due to the situation summarized in Table 5.15.

Karen suffers from two different disabilities: colour blindness and hearing loss. As AdaptUIOnt is focused on the user's preferences rather than on the disabilities, the user model configures her capabilities as is shown in Table 5.16.

Table 5.16 presents several important user parameters regarding the final adaptation. The

Scenario 2	
UserAux	**Value**
hasDisplayApplicable	*true*
hasAudioApplicable	*true*
hasRestriction	*hands_restriction*
	attention_restriction

Table 5.17: UserAux class generated by the pre-adaptation rules set and resulting UI first adaptation for both scenarios.

Scenario 2	
Adaptation	**Value**
hasColourSet	Colour blindness
hasViewSize	20
hasInput	Voice and haptic
hasOutput	Visual and audio
hasVolume	7 (max)

Table 5.18: Final adaptation for the Scenario 2.

volume of the device is allowed to be adapted by the platform. In addition, Karen uses the default input/output interaction channels. In this case, there is a temporary restriction for the user due to the current activity (i.e., driving) that is modelled through the *hasRestriction* datatype property (with the value *hands_restriction* and *attention_restriction*). This is shown in Table 5.17.

The final adaptation for this situation is modelled by the adaptation rules. As the limitations in this case come from the special situation generated by the activities performed by the user, AdaptUI has to change the interaction channels considering that the user has several restrictions. Thus, the size of the corresponding views is increased (not because Karen's colour blindness, but because of the activity of driving), the interaction channel allows voice interaction, and the volume is also increased due to the possible noise while driving a car or traffic. These adaptations are shown in Table 5.18:

As said before, Imhotep does not consider activities in its user and device profile. However, we can assume a possible user configuration profile under the presented circumstances in Table 5.15. Thus, a possible profile is illustrated through Listing 16.

```
{
  "imhotep.devices.screen.height": 1024,
  "imhotep.devices.screen.width": 600,
  "imhotep.devices.modelname": "Samsung Galaxy Tab 2 7.0",
  "imhotep.devices.os": "Android",
  "imhotep.devices.os.version": "4.0",
  "imhotep.devices.capabilities.video": true,
  "imhotep.devices.capabilities.audio": true,
  "imhotep.devices.capabilities.gps": true,
  "imhotep.devices.capabilities.flash": true,

  "imhotep.user.capabilities.problems": true,
  "imhotep.user.capabilities.problems.sight": true,
  "imhotep.user.capabilities.problems.sight.colour.blindness":
                                          "red_green",
  "imhotep.user.capabilities.problems.hearing": true,
  "imhotep.user.capabilities.problems.hearing.loss": "hearing",
  "imhotep.user.capabilities.problems.smell": false,
  "imhotep.user.capabilities.problems.touch": false
}
```

Listing 16: The variables file, in which device characteristics and user capabilities are described for the Scenario 2.

Regarding Listing 16 the Imhotep framework will avoid combining red/green colours, but no magnification of the screen or the size of the text would be added. Voice control will also be provided, and volume levels will be increased by indicating a hearing disability.

5.1.3.2.1 Discussion

In this scenario a situation in which the user capabilities are limited by the current activity is presented. Karen is driving in this scenario, and this activity makes her unable to use the usual interaction channel with the device. This activity normally requires a full attention to the road and the use of both hands. Therefore, we find two different limitations that this scenario combine. Considering this situation AdaptUI models several restrictions through the pre-adaptation rules. These restrictions and their representation in the AdaptUIOnt ontology are shown in Table 5.17. The final adaptation for this situation is shown in Table 5.18. In this case the main adaptations are centred in the interaction channel, as the user is driving and suffers from several restrictions due to the current activity. Hence, audio control and bigger views are presented.

Solution	User capabilities			
	Sight	**Hearing**	**Other**	**Activity**
Imhotep	✓	✓	✓	
AdaptUI	✓	✓	colour blind	"driving"

Table 5.19: AdaptUI and Imhotep comparison of the final reached adaptation for the Scenario 2 regarding the user.

As happens in Scenario 1, Imhotep does not cover the circumstances described for this scenario through Table 5.15. However, it is possible to somehow represent this through the user profile variables. Listing 16 shows an example of a modified user profile representing several limitations that, although they might not be true, they represent a practical situation in which the user is limited by external agents (which would be the activity of driving). Thus, Imhotep considers that the user is blind and suffers from hearing loss. Consequently, the resulting user interface also increases the volume levels, uses TTS for reading the displayed information and changes the colour set.

5.1.3.2.2 Conclusions

Karen suffers from two disabilities which, combined with the current activity makes the interaction with her mobile device troublesome. In this scenario, Karen is driving her car and intends to use a GPS guiding application to go to a certain location. AdaptUI covers this situation adapting the interaction channel and through a few modifications in the displayed views. On the contrary, to be able to represent a similar situation with Imhotep, the user variables profile has to be modified.
The main differences between both solutions is shown when Karen interacts with the application and the results are not satisfactory. This means that, although the application's user interface has been adapted, it might not be close to what the user needs. In AdaptUI the Adaptation Polisher dynamically monitors this interaction and performs several changes under the post-adaptation rules set. On the contrary, Imhotep would require a whole new specification of the user profile, with the corresponding consequences.
Table 5.19 and Table 5.20 show the main difference of these two solutions in the modelling of the user, the context and the device. Through them we can see how both solutions model the user disabilities, but only AdaptUI is aware of the activity of the user and the restrictions it implies.

Solution	Context characteristics			Device characteristics	
	Light	Noise	Temperature	Resolution	Battery
Imhotep				✓	
AdaptUI			✓	✓	✓

Table 5.20: AdaptUI and Imhotep comparison of the final reached adaptation for the Scenario 2 regarding the context and the device.

	Scenario 3
User	
- Personal data	Patrick, 25 years old, British
- Activity	
- Known disabilities	Sight disability
Context	
- Location	Relative: London, United Kingdom
- Time	12:00
- Brightness	20,000 lx
- Temperature	14 ºC
Device	Samsung Galaxy Ace
	Battery: 55%

Table 5.21: Scenario 3 situation summary.

5.1.3.3 Scenario 3: Limitations Caused by Disabilities

The last scenario introduces a situation in which the user suffers from a disability that impedes a proper interaction with the device. The scenario and its main characteristics are described in the following lines.

Patrick is a 25 years old man who lives in London. He lost his eyesight when he was 10. He uses accessibility tools when he interacts with computers and home devices. His PC is equipped with an utility which reads from the screen so he can navigate and use it. The problem is that a similar feature in his mobile device is often not enough depending on the context situation. Mostly traffic and street noise usually make difficult for Patrick to hear the messages read by the device.

Table 5.21 shows the configuration of the first model for this scenario. As can be seen in this table, Patrick needs from some adaptation to interact with his mobile device. Common accessibility tools cover this issue by reading the text in the screen. However, they do not consider context. This entails several obstacles during the interaction, for example when there are problems to hear the text read by the mobile device.

To reach the final adaptation the following steps are performed by the AdaptUI platform. As Patrick configures his profile indicating that the preferred input interaction should be *"voice_control"* (as can be seen in Table 5.23, the system becomes aware of the fact that Patrick suffers from a sight disability). Besides, the profile is configured indicating that

Class	Scenario 3	
	Data property	**value**
Display	*userDisplayBrightnessIsStatic*	true
	userDisplayIsApplicable	false
Audio	*userDisplayApplicableIsStatic*	true
	userAudioHasApplicable	true
	userAudioApplicableIsStatic	false
	userAudioHasVolume	4
Interface	*userInterfaceInput*	voice_control
	userInterfaceOutput	audio
Experience	*userHasExperience*	default
View	*userViewIsStatic*	true
Other	*userHasLanguage*	English
	vibration	true

Table 5.22: User profile for Scenario 3.

Scenario 3	
UserAux	**Value**
hasDisplayApplicable	false
hasDisplayBrightness	false
hasAudioApplicable	true

Table 5.23: UserAux class generated by the pre-adaptation rules set and resulting UI first adaptation for both scenarios.

the Display is not applicable, and that its configuration should not be adapted (because in this case it is not necessary, as Patrick is blind). Therefore, the adaptation should result into a promotion of an alternative channel, so the user would be able to interact alternatively and perform the same tasks. The volume value configured by Patrick is defined as 4. This is because the user considers that this value is enough to clearly hear the device's voice. Nevertheless, Patrick knows that this should not be an static value, as when he is in the street sometimes that value does not allow him to hear properly. As a consequence, the *userAudioApplicableIsStatic* property is modelled as false.
Taking into account the input shown in Table 5.21 and Table 5.22, AdaptUI begins the process filling the auxiliary classes and properties as is shown in Table 5.23
In the final adaptation is shown how due to Patrick's sight disability the AdaptUI platform adapts the interface channel with the user. Thus, the voice becomes the best choice for interacting with the device.
Imhotep models user visual disabilities in its profile under the *problems.sight* variable. The problem comes when the user has to fill the profile. In AdaptUI the Capabilities Collector allows blind users to configure the interaction channels thanks to voice control. In any case, considering that the user profile is configured, it is shown in Listing 17.

Scenario 3	
Adaptation	**Value**
hasBrightness	false
hasInput	voice
hasResponse	vibration
hasVolume	7 (max)

Table 5.24: Final adaptation for the scenarios 3.

```
{
  "imhotep.devices.screen.height": 480,
  "imhotep.devices.screen.width": 320,
  "imhotep.devices.modelname": "Samsung Galaxy Ace",
  "imhotep.devices.os": "Android",
  "imhotep.devices.os.version": "2.3.6",
  "imhotep.devices.capabilities.video": true,
  "imhotep.devices.capabilities.audio": true,
  "imhotep.devices.capabilities.gps": true,
  "imhotep.devices.capabilities.flash": true,

  "imhotep.user.capabilities.problems": true,
  "imhotep.user.capabilities.problems.sight": true,
  "imhotep.user.capabilities.problems.sight.dioptres": "blindness",
  "imhotep.user.capabilities.problems.hearing": false,
  "imhotep.user.capabilities.problems.smell": false,
  "imhotep.user.capabilities.problems.touch": false
}
```

Listing 17: The variables file, in which device characteristics and user capabilities are described for the Scenario 3.

Regarding the profile it is very difficult for Imhotep to make the proper decisions. The input file lacks of practical information about the user disability. In fact, the Imhotep's adapted user interface is the same as in the Scenario 2.

5.1.3.3.1 Discussion

In this scenario we present a situation in which the adaptation should be lead by a concrete user disability. Here, context or activities are not as important as in the other scenarios. Thus, the efforts of the AdaptUI platform are focused on avoiding any visual interface.
In the case of AdaptUI this is carried out based on the user model filled during the inter-

Solution	User capabilities			
	Sight	Hearing	Other	Activity
Imhotep	✓			
AdaptUI	✓		✓	✓

Table 5.25: AdaptUI and Imhotep comparison of the final reached adaptation for the Scenario 3 regarding the user.

Solution	Context characteristics			Device characteristics	
	Light	Noise	Temperature	Resolution	Battery
Imhotep				✓	
AdaptUI		✓		✓	✓

Table 5.26: AdaptUI and Imhotep comparison of the final reached adaptation for the Scenario 3 regarding the context and the device.

action with the Capabilities Collector. This module allows blind users to interact with it using their voice. On the contrary, in the case of Imhotep this is not possible. Filling the user profile (represented by the variables file) is impossible for a blind user.
Nevertheless, considering that the user is able to fill it, Imhotep cannot react to the specific disability of the user. It is true that it will present an alternative user interface, with audio instructions and a different colour configuration set, but it will not take the specific problem of the user into account. On the contrary, the AdaptUI platform does not consider any specific disability. As it is centred in the preferences of the user, the resulting adapted user interface will be more practical.

5.1.3.3.2 Conclusions

As is shown in Table 5.22 Patrick has a sight disability. Imhotep takes sight and hearing disabilities into account for the adaptation process. Therefore, depending on the network range and on the server side programmed adaptations different UI configuration will be available. The main setback is that Patrick has to indicate the concrete physiological problem. For example, he would have to specify a figure detailing the dioptres he has. Then, the Imhotep server would act consequently. According to this, if the concrete figure has not been considered in the adaptation process, the resulting UI might not be adequate. Another issue is that, the network dependency would make the adaptation process too long. AdaptUI covers this problem as it runs fully in the mobile device.
The main differences when dealing with the user, context and device models in this scenario are shown in Table 5.25 and Table 5.26.

5.1.4 Developers Using AdaptUI

Before introducing the results obtained from working with final users (as consumers of adapted user interfaces), an experiment with developers has been performed. This experiment aims to evaluate the provided APIs, and also to compare the differences between performing a configuration of the user interface in Android with and without AdaptUI. Thus, developers with experience in both Android development and semantics are involved.
As AdaptUI provides an API conceptually divided in two (see Section 4.3), this experiment has been fragmented accordingly. The experiment, consisting in 6 steps and with a 20 minutes duration, has been carried out as follows:

1. First, the developer is introduced to the AdaptUI framework through a brief explanation of AdaptUI's goals, virtues and boundaries. Both APIs are described and explained to them, and the APIs descriptions are given in a printed version, as they appear in Table 4.9 and Table 4.10.

2. Secondly, a Java project built with the Eclipse IDE is presented. This project uses the AdaptUI framework as an imported library. The main class of this Java application instantiates AdaptUI and initializes the environment. Besides, several *TODO* tasks with the corresponding instructions are given for the developer to complete. This class is summarized through Listing 18. These tasks include:

   - The creation of a new class.
   - The creation of a new instance of the previously created class.
   - Adding a new datatype property with the corresponding value to the new instance.
   - Adding a new object property to the new instance, involving several classes[1].
   - The creation of a new rule, involving at least three classes (including the created class), and the previous mentioned datatype and object properties.

[1] As the developer might not know how the AdaptUIOnt ontology is structured, a brief explanation is given using Protégé.

```java
public class Main {
  private final static String ADAPTUI = "original/adaptui.owl";
  private static final String NAMESPACE =
        "http://www.morelab.deusto.es/ontologies/adaptui.owl#";

  public static void main(String[] args) {
    AdaptUI adaptUI = new AdaptUI();

    // Loading the ontology through the ontology manager
    adaptUI.loadOntologyFromFile(new FileInputStream(
            new File(ADAPTUI)));
    // TODO: create a new class

    // TODO: create a new individual of the created class

    // TODO: create a new datatype property for the individual

    // TODO: create a new object property for the individual

    // TODO: create a new SWRL rule

    // TODO: modify a Button instance background color

    // Storing the ontology modifications
    try {
      adaptUI.saveOntology("stored/test.owl");
    } catch (OntologySavingException e) {
      e.printStackTrace();
    }
  }
  // Check methods
  // ...
  // checkClassInsertion(NAMESPACE, className);
  // ...
}
```

Listing 18: The default main class with the corresponding tasks to be accomplished by the developer.

3. Once the corresponding knowledge modifications are made, the resulting ontology is stored. Developers are requested to check the changes in the ontology using Protégé. Figure 5.10 shows an example of a modified ontology by a participating developer. Hence, the developer is allowed to check if the changes made in the Java main class using the AdaptUI framework have been correctly represented in the ontology. If any errors are

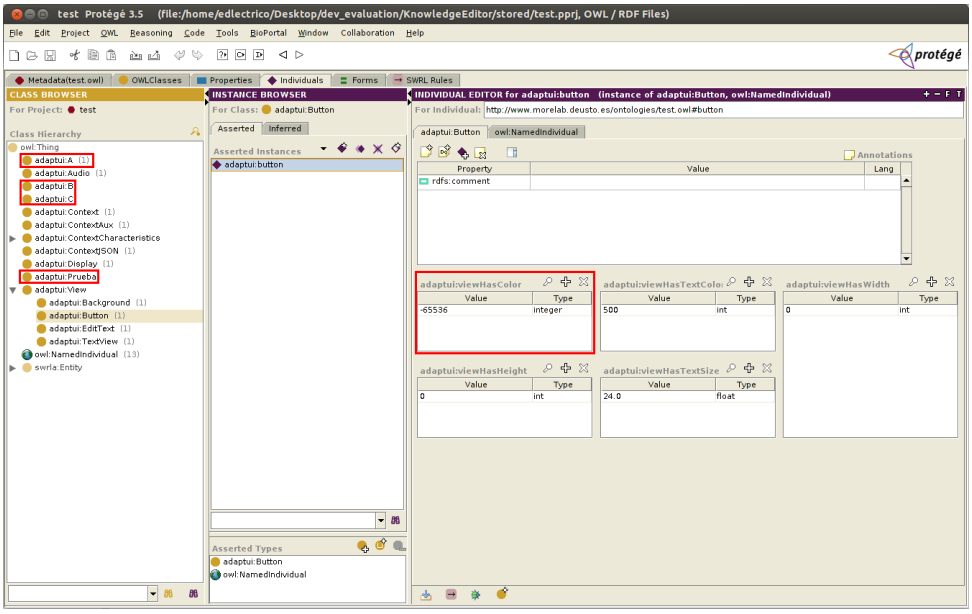

Figure 5.10: A test ontology modified by one of the participating developers. The modifications are highlighted in red. Several classes have been added. Also the *viewBackgroundColor* for the *Button* only instance has been modified.

found, the process is revised.

4. After the modification of the ontology, the AdaptUIDemo Android project is presented to the developer. This project's main activity shows a text view, an edit text and a button. The goal of this part of the experiment is to evaluate how the developer uses the adaptation API to dynamically modify these views. The changes performed in the ontology in the previous steps are taken into account. In order to make this application to use the new modified version of the ontology, an ADB command is needed. This command pushes the required file from the current computer disk to the desired Android folder. Listing 19 shows how the command ADB push command works.

```
1 adb push ontology.owl /sdcard/ontologies
```

Listing 19: The ADB push command. The first parameter specifies the location of the file to be sent to the device. The second parameter points at the absolute location in the device.

5. Next, once the developer checks that the adaptation to the views shown in the demo application matches the modifications performed with the knowledge API, the developer is asked to use the adaptation API through an Android project. Similar to the first case, the developer is asked to complete several tasks (see Listing 20).

```
1  public class ActivityExample extends Activity {
2    private GridLayout layout;
3    private TextView textView;
4    private Button button;
5    private EditText editText;
6
7    private static final String ONTOLOGY_FILE = "adaptui.owl";
8    private static final String ONT_PATH = "/sdcard/ontologies/";
9    private static final String ADAPTUI_NAMESPACE =
10         "http://www.morelab.deusto.es/ontologies/adaptui.owl#";
11
12   @Override
13   protected void onCreate(Bundle savedInstanceState) {
14     super.onCreate(savedInstanceState);
15     setContentView(R.layout.main_activity);
16
17     layout = (GridLayout) findViewById(R.id.layout);
18     textView = (TextView) findViewById(R.id.textView);
19     button = (Button) findViewById(R.id.button);
20     editText = (EditText) findViewById(R.id.editText);
21     // Initializing the framework
22     AdaptUI adaptUI = new AdaptUI(ADAPTUI_NAMESPACE, views);
23     // Simulating context change with a listener
24     button.setOnClickListener(new OnClickListener() {
25     @Override
26     public void onClick(View v) {
27       // TODO: Change layout color (the name of the
28       // class is "Background")
29
30       // TODO: Change button size (text size), and
31       // background and text color
32
33       // TODO: Change edit text size (text size), and
34       // background and text color
35
36       // TODO: Change text view size (text size), and
37       // background and text color
38       }
39     });
40
41     adaptUI.loadOntologyFromFile(ONT_PATH, ONTOLOGY_FILE);
42 }
```

Listing 20: The default Android version with the corresponding tasks to be completed by the developer. These tasks include adapting the background and text colour of a button, its size and the background colour of the whole activity

6. Finally, the developer uses the Capabilities Collector to adapt several views as a user. As the last task of the Capabilities Collector is to store the required values in the ontology through the Semantic Modeller, the developer is able to check these changes again in the AdaptUIDemo application.

5.1.4.1 **Results and Conclusions**

The resulting experiences with the provided APIs have been collected through a simple questionnaire. It is a very straightforward questionnaire, not based on any standard. Its purpose is to find out if the participating developers miss any possible feature and to gather their feedback about AdaptUI. The questionnaire is formed by the following questions:

1. #Q1: How useful do you find the AdaptUI framework for developing adaptive user interfaces?[1]
2. #Q2: Do you miss any additional functionality?
3. #Q3: If so, please specify the functionality you would add to the framework/API.
4. #Q4: Would you improve any API feature?
5. #Q5: If so, which one and how?
6. #Q6: Would you make AdaptUI part of your future applications (if you plan to use adaptive interfaces)?
7. #Q7: Comments.

The developers' responses are gathered in Table 5.27 and Table 5.28. The results show that the participating developers have found the AdaptUI API very useful. The comments, as expected, reveal that they would find it more useful if a OOP paradigm is followed. In the current version the definition and modification of the knowledge and of the adaptation is more declarative. These considerations are gathered in Section 6.4.

5.2 User Evaluation

As said in the introduction of this chapter, this section deals with the user evaluation. In order to make a proper comparison between the presented systems several users have been guided through a demonstration of AdaptUI and a comparison with Imhotep. Besides, they have been questioned according to the SUS guidelines.

[1] This question requires the user to select a value from a 1 to 5 scale, 1 meaning *not much* and 5 meaning *very much*.

#Developer	#Q1	#Q2	#Q3	#Q4	#Q5	#Q6	#Q7
#1	4	Yes	*	Yes	-	Yes	-
#2	5	No	-	Yes	*	Yes	-
#3	5	No	-	No	-	Yes	-
#4	4	No	-	Yes	*	Yes	-
#5	4	No	-	Yes	*	Yes	*

Table 5.27: Developers' responses to the questionnaire. As responses for questions #Q3, #Q5 and #Q7 might be too long, their details are given in Table 5.28. If a response for any of these questions is given by the developer an asterisk symbol(*) is shown in the corresponding cell.

#Developer	#Q3	#Q5	#Q7
#1	To be able to define more than one configuration for the same view.	-	-
#2	-	OOP paradigm.	-
#3	-	-	-
#4	-	Pass the object to the adaptation method as parameter.	OOP paradigm.
#5	-	Pass the object to the adaptation method as parameter.	This is my first experience with ontologies, and I've found it satisfactory.

Table 5.28: Developers' responses to questions #Q3, #Q5 and #Q7.

Figure 5.11: The DEC VT100 video terminal, introduced in August 1978 [39].

5.2.1 The SUS Questionnaire

The SUS [35] is a 10-item questionnaire developed by John Brooke in 1986 which gives an overview of satisfaction of users with software. Originally, it was conceived as a "quick and dirty" scale to collect the usability feedback from using systems similar to VT100 Terminal applications (see the DEC VT100 video terminal in Figure 5.11).
One of the SUS questionnaire main characteristics is its ability to collect usability feedback through a 10 item questionnaire with 5 response options:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

To measure the resulting responses from the users, the formula described below is applied:

Strongly Disagree 1	2	3	4	Strongly Agree 5
O	O	O	O	O

Figure 5.12: The SUS response format [39].

	Disagree				Agree
1. I think that I would like to use this system frequently.	☐	☐	☐	☐	☑
2. I found the system unnecessarily complex.	☐	☐	☐	☑	☐
3. I thought the system was easy to use.	☐	☑	☐	☐	☐
4. I think that I would need the support of a technical person to be able to use this system.	☑	☐	☐	☐	☐
5. I found the various functions in this system were well integrated.	☐	☑	☐	☐	☐
6. I thought there was too much inconsistency in this system.	☐	☐	☑	☐	☐
7. I would imagine that most people would learn to use this system very quickly.	☐	☑	☐	☐	☐
8. I found the system very cumbersome to use.	☐	☐	☐	☑	☐
9. I felt very confident using the system.	☐	☐	☐	☐	☑
10. I needed to learn a lot of things before I could get going with this system.	☐	☐	☑	☐	☐

Table 5.29: Example of a completed SUS questionnaire. Total score = 22; SUS Score = 22 * 22.5 = 55

- For odd items: subtract one from the user response.
- For even-numbered items: subtract the user responses from 5.
- This scales all values from 0 to 4 (with four being the most positive response).
- Add up the converted responses for each user and multiply that total by 2.5.
- This converts the range of possible values from 0 to 100 instead of from 0 to 40.

Thus, a figure between 0 and 100 is obtained, indicating the percentage of acceptance of the corresponding evaluated system. Figure 5.12 shows an example of the SUS questionnaire format. Besides, Table 5.29 shows an example of a completed SUS questionnaire evaluating a generic system.

Although there are other interesting questionnaires (i.e., Software Usability Measurement Inventory (SUMI) [34], Measuring the Usability of Multi-Media Software

(MUMMS) [23], Questionnaire for User Interaction Satisfaction (QUIS) [31], and so on), the SUS questionnaire has become an industry standard with references in over 600 publications [22].
For this evaluation, and along with the SUS questionnaire, several extra questions have been added. The purpose of these questions is to segment the whole users set into different subsets of users under similar conditions. Thus, three extra classifications have been performed following the following criteria:

- *By age*. The users of AdaptUI might encounter different difficulties when dealing with the adaptation platform. For example, users older than 65 may not be used to using smart devices or touching screens. As shown through the following charts, age and technology experience are related. To be aware of this issue an age based classification has been performed. The subgroups for this age classification are:
  - Users aged under 20 years old ($x < 20$)[1].
  - Users aged between 20 and 35 years old ($20 \leq x <$35).
  - Users aged between 35 and 50 years old ($35 \leq x < 50$).
  - Users aged between 50 and 65 years old ($50 \leq x < 65$).
  - Users aged over 65 years old ($x > 65$).
- *By experience with technology*. As it might happen with age, the technology experience of the users might result into different usability experiences when using AdaptUI. Touching screens, smart devices, or even technical computer related instructions may be complex depending on the user experience. Hence, a simple classification considering this problem is made with the following criteria:
  - *Low*, which indicates a user who is not very familiar with technology. This kind of users require non technical further explanations of the AdaptUI features, purpose and characteristics. Besides, several SUS questions might trouble these users. Thus, it is desirable to accompany them through the SUS process.
  - *Medium*, which means that the user usually interacts with technology and understands the most common interaction processes and technical vocabulary. Nevertheless, too technical instructions and features might confuse them.
  - *High*, which characterizes those users who have high level technical knowledge due to their jobs, hobbies, age, and so forth. These users do not require extra explanations or guidelines.

[1]The x represents the age of the user. In this case the x is also shown in the following charts during this section in the Legend.

- *By disability*. Users are asked if they sense that they might suffer from visual or hearing disabilities. Problems when dealing with devices under certain context conditions are included. On the contrary, no specific visual or hearing disability is concreted. The idea is to capture the feeling of the users when manipulating devices under certain conditions (due to context or due to their own capabilities).

Hence, before beginning with the SUS questionnaire, the following 4 questions are presented, allowing users to select the corresponding responses through several combo boxes:

a) Please select your age within the following ranges.

b) Which would you say is your experience level with technology?

c) Do you feel that, sometimes, you cannot use your mobile device as you would like due to a temporary or enduring visual problem? This problem might be caused by physiological or context conditions.

d) Do you feel that, sometimes, you cannot use your mobile device as you would like due to a temporary or enduring hearing problem? This problem might be caused by physiological or context conditions.

Besides, after these and the SUS questions two extra questions are presented seeking developers feedback. The responses are requested through several check boxes:

a) I am a software developer and I usually work with user interfaces.

b) Considering that I am a developer, I find the AdaptUI API very helpful to ease the adaptation of user interfaces.

5.2.2 Discussion

After using AdaptUI and checking its adaptation capabilities, users have been asked to complete the SUS questionnaire with the mentioned extra questions. This test has been completed by a total of 30 users. Classified into different groups regarding their age, technology experience and temporary or enduring disabilities the obtained results are explained in the following lines illustrated with several figures and charts.
First, without considering any of the cited classification criteria, the usability results regarding the AdaptUI adaptation system is illustrated through Figure 5.13. This figure shows that the 63,3% of the users punctuated AdaptUI with over 70 within the SUS scale. On the contrary, approximately 30% of the users find its usability between 50 and 70 points.

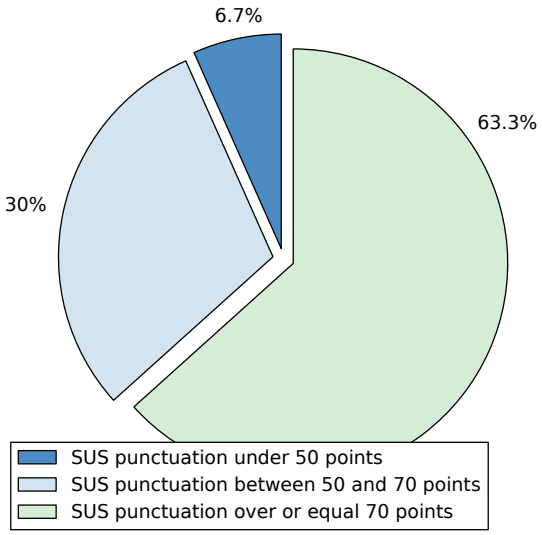

Figure 5.13: The SUS responses.

Next, the cited criteria regarding age, technology experience and possible disabilities result in the following bar charts. The Figure 5.14 illustrates the differences regarding the SUS results taking into account the users' ages. In this chart, it is shown how users between 20 and 35 years old are the ones who mostly support the AdaptUI usability results. 36,66% of these users consider that the usability of AdaptUI is over 70 in the SUS scale. Along with the age, usually the experience with technology is directly related. To have this into account, Figure 5.15 illustrates the resulting differences encountered after analysing the responses given to the SUS questionnaire. As shown, users with *high* technology experience result into bigger usability results, reaching a final result of 40% of users punctuating AdaptUI over 70 points in the SUS scale. On the opposite side there are those users who lack technology experience. 20% of these users gave less than 70 points.

Besides, users are asked about possible disabilities. These disabilities do not have to be physiological. They are related with the feeling of disability that users might sense, caused by the context under certain conditions or caused by other factors. Every user is asked about any possible disability when dealing with interactions with their devices. As AdaptUI does not consider physiological capabilities, users are not asked about this kind of issues directly. The obtained results show how 100% of the asked users with visual or hearing disabilities find AdaptUI totally usable, punctuating it in the SUS scale over 70.

Finally, we have distinguished between potential users and those who have development knowledge. In the first versions of the questionnaire this was not taken into account. However, several evaluations revealed that many users with developer profiles stated that they might not find AdaptUI useful as users. This was mainly based on the interactions they perform with their device, in which they do not feel the necessity of adaptations in

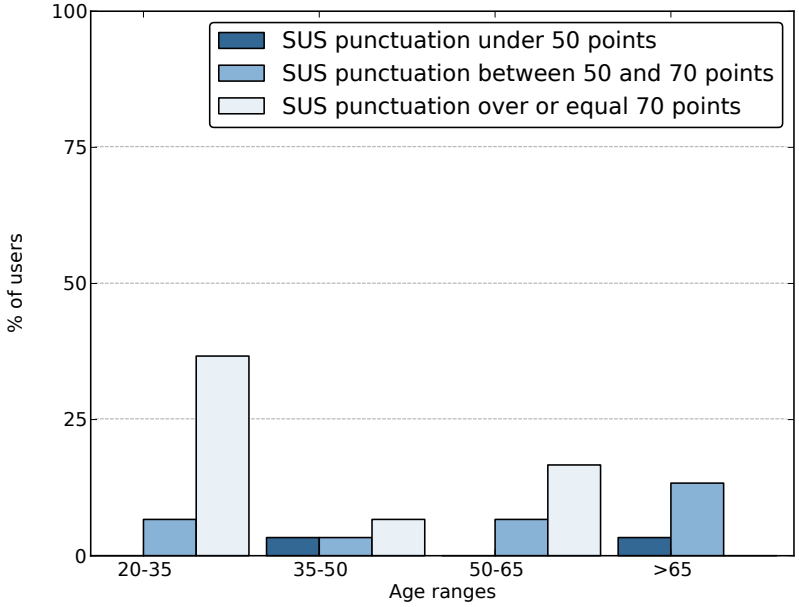

Figure 5.14: The SUS responses taking into account the age range of the users.

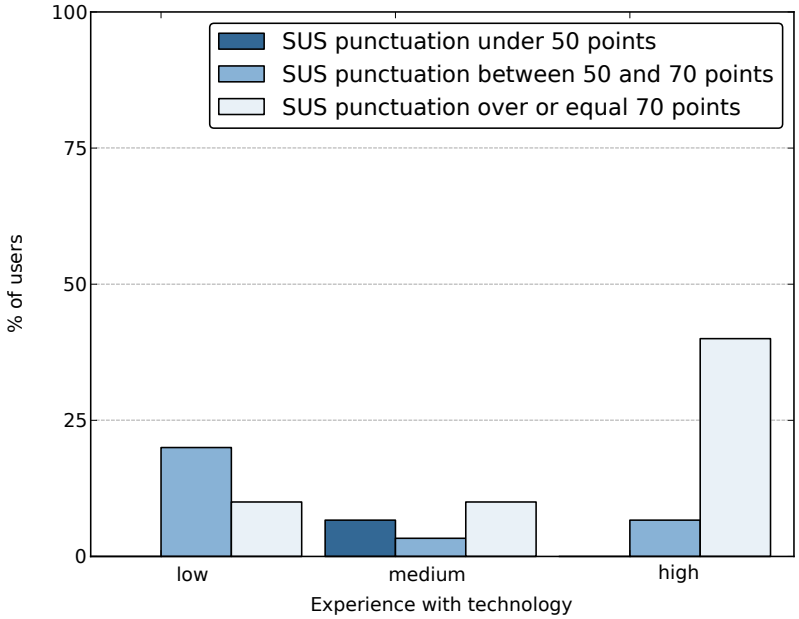

Figure 5.15: The SUS responses taking into account the experience with technology of the users.

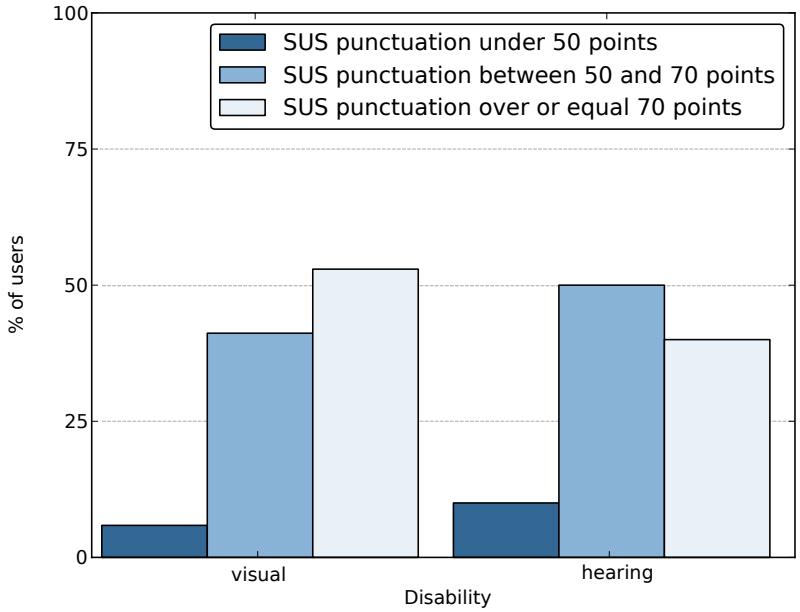

Figure 5.16: The SUS responses taking into account the visual and hearing disabilities indicated by the users.

any scenario. On the contrary, they were willing to use it as developers due to the user interface adaptation capabilities of the platform. 100% of the users that consider themselves developers stated that they would like to use the platform. However, Figure 5.17 shown how only 81.8% of these punctuated AdaptUI with a value of over 70 in the SUS scale. On the other side, 18.2% of the developers stated a value between 50 and 70.

5.2.3 Conclusions

Reviewing the results obtained during the user evaluation, several conclusions have been extracted. In the following lines we summarize these conclusions taking into account the segmentation of users by age, experience with technology, possible or temporary disabilities, and developer users:

- Users ageing' is fundamental when dealing with AdaptUI for the first time. While users under 35 years old find most of the features and terminology easy to understand, older users usually need more high-level explanations. More precisely, 36.6% of the users ageing between 20 and 35 support AdaptUI with more than 70 points in the SUS scale. On the contrary, users ageing over 65 years old did not give more than 70 points in the SUS scale. This is mostly because they encountered several difficulties to understand not only the purpose of the system but also the features it provides, and the used and required terminology. This is perceptible

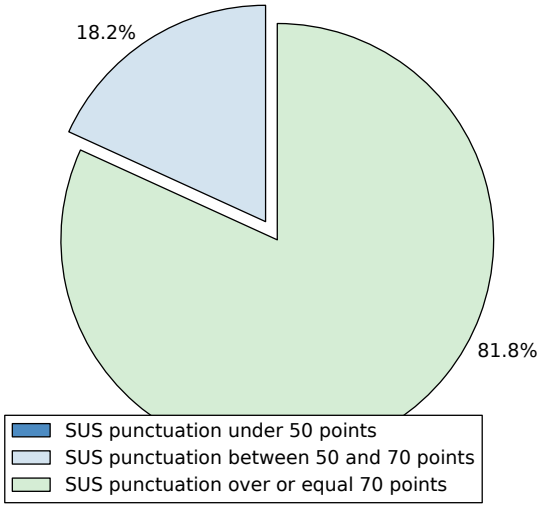

Figure 5.17: The SUS responses taking into account if users are developers.

through Figure 5.14.

- The experience with technology, highly associated with ageing, also indicates varying skill when using the AdaptUI system. Experienced users not only understand the features provided by AdaptUI, but they also predict the results of the experiments or even contribute with possible features to cover their daily experiences with user interfaces. As illustrated through Figure 5.15, 40% of the users with *high* experience with technology gave to AdaptUI more than 70 points in the usability scale. Contrarily, 10% of the *low* experienced users gave the same punctuation.

- Regarding the possible disabilities we might suffer from when manipulating our devices in different contexts, users are very comfortable with the purpose of AdaptUI. Nevertheless, in this point several differences are found. For example, a significant percentage of users that are also developers find the AdaptUI API more interesting than the scenarios. Focusing on the results, Figure 5.16 shows similar outcomes considering both visual and hearing disabilities. In the first case 52.9% of the users gave more than 70 points, while in the second case the percentage is 40%.

- Finally, regarding the results obtained from users who are also developers, we find that 81.8% of these users find AdaptUI usable, giving 70 or more points in the SUS scale. The rest of developer users stated that they find AdaptUI a necessary platform for the inclusive design of the user interface of their applications, expressing that, as users (not developers), they might not use AdaptUI. These outcomes are illustrated through Figure 5.17. Nonetheless, a further evaluation with these users is presented in Section 5.1.4, in which they are requested to use the provided APIs to make

several applications adaptable.

In general terms the acceptance of AdaptUI regarding usability and developer needs is promising. As Figure 5.13 shows, 63.3% of all the users find AdaptUI usable over 70 usability points. On the contrary only 6.7% gave less than 50 points. Nevertheless, and as detailed in Section 6.4, more efforts are planned to try to get better results for both users and developers.

5.2.4 The SUS Results

This section shows the results of the SUS questionnaire carried out among 30 potential users. Table 5.30 presents the replies to the additional questions detailed in Section 5.2.1. Table 5.31 shows the anonymous responses and final computed SUS value.
The SUS questions are the following:

1. #Q1: I think that I would like to use this system frequently.
2. #Q2: I found the system unnecessarily complex.
3. #Q3: I thought the system was easy to use.
4. #Q4: I think that I would need the support of a technical person to be able to use this system.
5. #Q5: I found the various functions in this system were well integrated.
6. #Q6: I thought there was too much inconsistency in this system.
7. #Q7: I would imagine that most people would learn to use this system very quickly.
8. #Q8: I found the system very cumbersome to use.
9. #Q9: I felt very confident using the system.
10. #Q10: I needed to learn a lot of things before I could get going with this system.

5.3 Evaluation Discussion

Along this chapter several experiments have been carried out to test the AdaptUI platform. First, we have compared the possible consequences of porting semantics and reasoning to a mobile Android based platform. Through the experiments shown in Section 5.1 it has been demonstrated how managing semantics in mobile phones is possible and affordable. The only condition we find that should be considered when dealing with mobile reasoning

#User	Age range	Experience	Sight temporary disability	Hearing temporary disability	Developer	I find it useful
#1	20-35	High	No	No	No	No
#2	20-35	Medium	Yes	Yes	No	No
#3	20-35	High	No	No	Yes	Yes
#4	20-35	High	No	No	Yes	Yes
#5	20-35	High	No	No	Yes	Yes
#6	51-65	Medium	No	No	No	No
#7	20-35	High	No	No	Yes	Yes
#8	20-35	High	No	No	Yes	Yes
#9	20-35	High	No	No	Yes	Yes
#10	20-35	High	No	No	Yes	Yes
#11	20-35	High	No	No	Yes	Yes
#12	20-35	High	No	No	Yes	Yes
#13	20-35	High	Yes	No	Yes	Yes
#14	51-65	Low	Yes	Yes	No	No
#15	51-65	Low	Yes	No	No	No
#16	20-35	High	Yes	No	No	No
#17	>65	Medium	No	No	No	No
#18	36-50	High	Yes	No	Yes	No
#19	51-65	Medium	Yes	Yes	No	No
#20	51-65	Low	Yes	Yes	No	No
#21	36-50	Low	Yes	Yes	No	No
#22	36-50	Medium	Yes	Yes	Yes	Yes
#23	>65	Low	Yes	Yes	No	No
#24	>65	Low	Yes	Yes	No	No
#25	>65	Medium	Yes	Yes	No	No
#26	>65	Low	Yes	Yes	No	No
#27	51-65	Low	Yes	No	No	No
#28	51-65	Low	Yes	No	No	No
#29	20-35	High	No	No	No	No
#30	36-50	High	Yes	No	Yes	No

Table 5.30: The additional questions.

#User	#Q1	#Q2	#Q3	#Q4	#Q5	#Q6	#Q7	#Q8	#Q9	#Q10	Total
#1	4	2	4	1	4	2	4	1	5	1	85
#2	5	2	4	2	3	3	5	2	4	1	77.5
#3	4	3	4	4	4	2	4	2	4	4	62.5
#4	2	1	4	1	4	2	3	1	4	1	77.5
#5	5	2	4	2	4	3	4	1	4	2	77.5
#6	5	1	5	4	5	1	5	1	3	3	82.5
#7	2	1	4	1	5	1	3	1	5	1	85
#8	3	2	5	3	4	2	5	2	4	4	70
#9	3	1	5	1	4	2	4	1	5	1	87.5
#10	2	2	4	1	4	2	4	1	5	1	80
#11	3	2	3	1	3	1	2	2	2	2	62.5
#12	4	1	4	2	4	1	3	1	4	1	82.5
#13	4	1	5	1	4	1	5	1	5	1	95
#14	4	1	5	3	5	1	5	1	2	3	80
#15	5	3	3	4	5	3	3	2	2	2	60
#16	4	1	5	2	4	2	4	1	5	1	87.5
#17	1	3	2	3	3	3	1	3	5	4	40
#18	5	1	5	3	3	2	3	2	4	1	77.5
#19	4	1	4	4	5	1	5	1	3	3	77.5
#20	5	2	4	5	5	1	5	1	2	3	72.5
#21	4	2	3	4	4	3	5	3	3	4	57.5
#22	2	3	3	3	2	3	2	2	3	2	47.5
#23	4	2	3	4	5	3	5	4	3	5	55
#24	4	5	4	4	5	3	5	2	3	3	60
#25	4	2	4	4	5	3	5	3	3	3	65
#26	4	4	5	2	3	3	5	4	3	5	55
#27	5	2	5	3	4	2	5	2	2	3	72.5
#28	5	2	3	3	5	3	4	2	2	2	67.5
#29	4	3	5	2	5	1	4	2	4	1	82.5
#30	5	3	4	2	4	2	4	1	4	1	80

Table 5.31: Results of the SUS questionnaire.

is that the ABox and SWRL axioms set should be limited or small enough to avoid the overloading of the whole system. Table 5.3, Table 5.4 and Table 5.5 show these differences remarking performance of reasoning for the corresponding execution platform.
After this first part of the experiments, a comparison between other alternatives has been presented. As is explained in Section 5.1.2.1, Imhotep is a framework which aims to ease the development of adaptive user interfaces. The main problems or limitations of the Imhotep framework are:

- It cannot be deployed in a mobile phone. The design of the platform requires the existence of an external server which will manage the adaptation.
- To collect the user profile and capabilities a client application in the user's mobile phone is needed. This application sends user and device profile to the adaptation server.
- Besides, the knowledge about the user capabilities is physiological, which we have defended along this dissertation to be non practical and difficult to manage.
- It does not consider the current context, limiting the adaptations to the user's and devices capabilities.
- The adaptations are static. Once the server compiles the corresponding sources and the user interfaces are adapted, if the user is not comfortable with them the whole process has to be repeated.
- Finally, there is the problem of the network dependency, which approximately adds 14 seconds to the whole process for compiling the corresponding sources. Although Imhotep's performance when using the cache is good enough (see Figure 5.9), the response time decreases significantly when this cache is not present.

AdaptUI faces these problems inherited from Imhotep and solve them as follows:

- Regarding the problem of using an external server, the AdaptUI platform runs fully in the mobile device. It is true that nowadays these devices' capabilities are far from the ones present when Imhotep was conceived. This allowed us to design a platform compatible with Android and network or external services independent. Through the ontology and several optimized adaptation engines the adaptation is carried out without needing external processing assistance.
- Although AdaptUI also requires a software module to collect user capabilities, the perspective differs. In AdaptUI, the Capabilities Collector module aims to collect user capabilities regarding their preferences, not the physiological capabilities themselves. Besides, this module is fully integrated in the AdaptUI platform.

- As said before, AdaptUI does not consider physiological capabilities of the user. Instead of that, AdaptUI looks for several preferences of the user, modelled through the AdaptUIOnt ontology. Thus, adaptations and corrections on the corresponding adaptations are easier and practical.
- Regarding the context issue, AdaptUI considers the environment as one of the three main entities in the user interfaces adaptation domain. Besides, the disabilities that users might suffer are understood as a set of conditions that somehow limit the user. These conditions do not have to be specifically physiological. To us, context might also generate several situations in which the user might feel temporary disabled.
- The adaptations are dynamic, not just considering that they are modifiable in running time. They also change according to the experience monitored by the Adaptation Polisher. This module aims to always monitor the user interaction with the adapted interfaces to provide alternatives if the usability decreases.
- AdaptUI does not depend on any network connection. It runs fully in the mobile device, which allows a complete independence from external connectivity issues.

Regarding the performance of both platforms, Table 5.8 shows how after loading the ontology, the adaptation process response in AdaptUI is acceptable, not exceeding 2 seconds. On the contrary, if the cache is not used, Imhotep has to deal with a performance over 14 seconds due to the network dependency.
After these comparisons, several qualitative experiments have been presented through different scenarios (see Section 5.1.3). These experiments are based on several complex situations, in which AdaptUI is scrutinized. The presented scenarios have been categorized into 3 different groups, each group representing a concrete situation in which an adaptation of the user interface would be needed. In this case, these categories are:

a) Limitations caused by the context current conditions.

b) Limitations caused by the set of activities the user might be doing.

c) Limitations caused by the disabilities the user might suffer from.

These scenarios present a series of characteristics that distinguish them. After analysing each specific situation, the adaptation process is detailed. Besides, before presenting several conclusions, this adaptation process performed by AdaptUI is compared with Imhotep. Thus, an overview of the behaviour and performance of both systems is depicted.
Finally, the users' feedback is collected. To do so, an experiment with AdaptUI has been developed. Users have to use an application which translates the interaction to the

AdaptUIOnt ontology. Then, several context changes are triggered so that the users see how the user interface adapts dynamically. After this experiment, a questionnaire is presented, so their experiences with the AdaptUI platform can be collected. Along with several extra questions to ease the analysis of the results, the SUS questionnaire is used. The main idea extracted from these results is that users find the platform useful. Although there are several differences considering one group of users or another (e.g., based on their age, technical experience or disability), the obtained results are promising. Besides, the requested developers also consider that the framework would benefit from the design of adaptive user interfaces, integrating them with their personal applications. Hence, we conclude that, although much work can be performed, the results of the evaluation performed and described in this chapter are promising.

5.4 Conclusions

During this dissertation we have introduced AdaptUI as a platform for dynamic user interface adaptation. As new technical contributions have been developed and detailed (e.g., a mobile reasoning engine), several experiments regarding this area have been presented. Comparing the presented system with others gives us the perspective to criticise AdaptUI and extract several conclusions. In this section these conclusions are depicted.

On the one hand, the performed technical experiments revealed that mobile reasoning engines are useful, practical and affordable regarding performance. It is true that, in this case the *Pellet4Android* suffers depending on the hardware of the device. Besides, the more axioms we add to the corresponding ontology (specially to the SWRL axioms set), the bigger the delay in the performance of the system added. However, we believe that the AdaptUIOnt ontology is light enough to be easily managed by any Android based device. Furthermore, the smartphones market is increasing the possibilities of these devices by improving their hardware capabilities. This means that in the near future complex processing will be totally affordable by these devices.

The comparison with Imhotep exposes several benefits of AdaptUI. The first one, is the lack of dependency on external services or servers. This fact does not only have an affect on the performance, it avoids possible network failure problems. Another fundamental aspect that should be emphasized due to this comparison is the way users are modelled. While in Imhotep specific user disabilities are needed, AdaptUI just requires a simple interaction with the Capabilities Collector module. The interaction of the user is translated to the AdaptUIOnt ontology, and the rest of the adaptation process is delegated to the AdaptUI platform. Besides, AdaptUI relies on the Adaptation Polisher to perform slight modifications on the fly of the adapted user interface, in case the interaction with the user is not satisfactory. Regarding the time performances, although using cached applications in Imhotep, showed a faster response, the truth is that new and real scenarios revealed a

much bigger accuracy in AdaptUI. This has been shown through the scenarios presented in Section 5.1.3.
On the other hand 30 users have tested the platform and have been asked about its usability. Users seem to react satisfactorily to AdaptUI. The obtained results through the questionnaire show how depending on the users ageing and technical knowledge they feel more or less comfortable with it. These results also expose how most developers would use it as a tool for their applications. On the contrary, users who declare to suffer sometimes from context disabilities are willing to use it integrated with their Android device.

I've seen things you people wouldn't believe. Attack ships on fire off the shoulder of Orion. I watched c-beams glitter in the dark near the Tannhäuser Gate. All those moments will be lost in time, like tears in rain.

Blade Runner

CHAPTER

6

Conclusions

This chapter reviews the most significant results of this thesis, remarking the specific contributions and identifying several further research areas. First, an introduction discussing the contributions detailed during this dissertation is presented. These contributions are described in Section 6.2. Besides, several publications that support the different elements presented in this dissertation are enumerated in Section 6.3.

6.1 Discussion

This thesis arises as remarked in the introduction from a series of problems identified since 2010, when Imhotep was developed as a response to the observed demand of adaptive user interfaces solutions for people with disabilities. Dealing with the available technology and analysed approaches a solution in which developers were given several tools for developing adaptive interfaces was designed. Based on preprocessor primitives, developers were able to include pieces of source code in which user interface configuration were completed with a user and a device profile. Those profiles were configured by the potential users through another application.

Imhotep was well accepted by the scientific community. Several publications were produced (as shown in Section 6.3) and an award was given based on the AssistedCity developed use case.

However, in spite of the success of Imhotep, technology improvements regarding mobile devices have brought new challenges and opportunities. Thus, from the weaknesses of Imhotep new opportunities arose. These challenges motivated and led this dissertation.

First, a further analysis of the literature and state of the art in user interface adaptation solutions were needed. A lack of mobile based user interface adaptive solutions was iden-

tified. Besides, the current ongoing software and hardware mobile devices' improvements enhanced the design of a mobile platform.

After taking the decision of using a mobile centred platform, the challenge of *what* and *how* to model interface adaptations arose. Based on our previous experience and on the reviewed literature, a semantic model based on three main entities was designed. This model combines user's interaction characteristics, the current context situation description, and different device's characteristics. As one of the identified problems in the analysed user adaptation platforms in the literature is their lack of independence from the considered domain, a semantic based design was performed. This allows an easy method to represent the knowledge of the domain, extend it, share it, and adapt it to any desired sub-domain by just combining different ontologies with the provided AdaptUIOnt.

The combination of the two previous mentioned decisions brought a significant problem: the lack of readily available mobile-based semantic reasoning engines. Therefore, as a technical contribution, AdaptUI provides a mobile reasoning engine based on Pellet and compatible with Android. It is called *Pellet4Android*, it is open source, and it provides support for OWL 2 and SWRL rules in Android.

Once the semantic models and the mobile semantic infrastructure for reasoning were operative, the required architecture was designed, as shown in Figure 6.1.

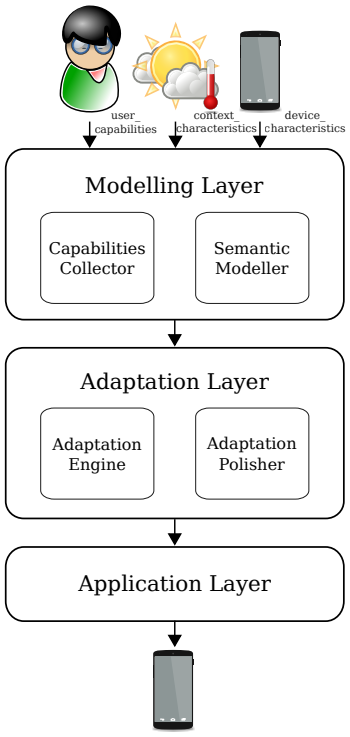

Figure 6.1: AdaptUI's three-layered global architecture.

Besides, in order to make AdaptUI accessible for developers, two main APIs are provided. These APIs aim to allow developers not only to adapt their application's user interface,

but also to adapt the whole AdaptUI platform to the domain they work with.
Although AdaptUI has several benefits and it contributes with the mentioned completed goals, the presented platform has also several constraints:

- Depending on the hardware of the used devices performance penalties might be identified if such devices have not the necessary computing capabilities. However, the results presented in Chapter 5 brought promising results with popular devices.
- Off-the-shelf, although we have demonstrated that several temporary context disabilities and limitations are considerably reduced, people with disabilities might still suffer several interaction problems. Thus, further efforts are needed to improve and refine the whole system. This issue is detailed in Section 6.4.
- Another limitation is that AdaptUI mostly considers disabilities based on visual and hearing constraints. Although others have been studied analysing the ICF (e.g., motor disabilities), the experiments are difficult to carry out. This is also detailed in Section 6.4, as we aim to cover more limitations and experiment with such contexts.

6.2 Contributions

This section details a summary of the different contributions described in this thesis:

a) In Chapter 2, an in-depth analysis of the state of the art is presented:
   - Evaluating several approaches to modelling and reasoning over user, context and device.
   - Analysing several adaptive and adaptable solutions in the field of user interfaces.
   - Studying these solutions' architectures and possibilities considering nowadays technology.
   - This covers the objective 1 detailed in Section 1.3.

b) In Chapter 3 an ontology for modelling users, context, devices and the corresponding user interface adaptations is presented.
   - Allowing the inclusion of adaptation rules to manage the adaptation process.
   - Avoiding the explicit user physiological capabilities modelling.
   - Providing a dynamic design to allow the platform to dynamically update the corresponding entity (i.e., user, context and device).

- The results of this chapter accomplishes the goal specified in objectives 2 and 3 in Section 1.3.

c) Chapter 4 presents an Android compatible mobile reasoning engine based on Pellet, and the corresponding architecture for enabling the dynamic user interface adaptation supporting semantics.

- Providing a full Pellet port available for Android devices: *Pellet4Android*.
- Allowing the use of semantics and SWRL compatible reasoning.
- Describing the different modules which power the whole system.
- Including the decisions' consequences and their conclusions.
- This covers all the objectives mentioned in Section 1.3, as it provides the necessary infrastructure to build the whole adaptation platform.

d) Finally, the previous contributions are combined to offer an implementation of a dynamic user interface adaptation platform. This platform provides a practical and complete vision of the user interaction capabilities. Besides, it gathers users and also developers through a series of development tools seeking the design of more usable and inclusive user interfaces. These achievements cover the hypothesis introduced in Section 1.3.

6.3 Publications and Awards

The contributions of this thesis have been presented to the scientific community in a series of international forums, such as: journals and conferences.

6.3.1 International JCR Journals

- Eduardo Castillejo, Aitor Almeida, Diego López-de-Ipiña, 2014. Ontology Based Model for Supporting Dynamic and Adaptive User Interfaces. International Journal of Human-Computer Interaction 30, 771–786. DOI:10.1080/10447318.2014.927287. Impact Factor (2013): 0.723.

- Eduardo Castillejo, Aitor Almeida, Diego López-de-Ipiña and Liming Chen. Modeling Users, Context and Devices for Ambient Assisted Living Environments. Sensors, MDPI. vol. 14, no. 3, pp. 5354-5391, DOI: 10.3390/s140305354, ISSN 1424-8220, JCR Impact Factor (2012): 1.953, January 2014.

- Eduardo Castillejo, Aitor Almeida, and Diego López-de-Ipiña. Modelling users, context and devices for adaptive user interface systems. International Journal

of Pervasive Computing and Communications 10, no. 1 (January 2014): 5-5. DOI:10.1108/IJPCC-09-2013-0028.

As mentioned before, this dissertation was motivated from the weaknesses and future work identified in previous research works.

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López-de-Ipiña and Marcos Sacristan. A method for automatic generation of fuzzy membership functions for mobile device's characteristics based on Google Trends. Computers in Human Behaviour (Journal). Volume 29, Issue 2. Pages 510–517. Impact Factor (2011): 2.293 DOI: 10.1016/j.chb.2012.06.005. March 2013.

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López-de-Ipiña, Marcos Sacristán. Imhotep: an approach to user and device conscious mobile applications. Personal and Ubiquitous Computing (Journal). Volume 15, Issue 4, pp 419–429. Springer. Impact Factor (2009): 1.554. ISSN: 1617-4909. DOI: 10.1007/s00779-010-0359-8. January 2011.

6.3.2 International Conferences

As this thesis started as a result of previous research work related to user interface adaptation, the following publications are included:

- Asier Aztiria, Eduardo Castillejo, Aitor Almeida, Diego López-de-Ipiña, 2014. Adapting user interfaces based on user preferences and habits, in: 10th International Conference on Intelligent Environments (IE14), 2014, vol., no., pp.9,15, June 30 2014-July 4 2014, Shangai, China, DOI: 10.1109/IE.2014.9

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López-de-Ipiña, and Marcos Sacristán. An approach to automatic generation of fuzzy membership functions using popularity metrics. In Information Systems, E-learning, and Knowledge Management Research, pp. 528-533. Springer Berlin Heidelberg, 2013.

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López-de-Ipiña, Marcos Sacristán. Adaptative applications for heterogeneous intelligent environments. ICOST 2011: 9th International Conference on Smart Homes and Health Telematics. Montréal, Canada, June 2011. LNCS6719, Toward Useful Services for Elderly and People with Disabilities, Springer, International Standard Book Number (ISBN): 978-3-642-21534-6, pp. 1-8.

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López-de-Ipiña, Marcos Sacristán. A user-centric approach to adaptable mobile interfaces Actas del II International Workshop of Ambient Assisted Living (IWAAL 2010), p.p. 153-160 Valencia, Spain, September 7-10, 2010 (ISBN: 978-84-92812-67-7).

6.3.3 Awards

AssitedCity, the use case developed using the Imhotep framework for adaptive user interfaces was awarded in March 7, 2012 with the Via Inteligente award[1]. This prize was possible thank to the collaboration with Dr. Aitor Almeida and Dr. Pablo Orduña, and the supervision of Dr. Diego López-de-Ipiña.

6.4 Future Work

Despite all the accomplishments achieved through the AdaptUI platform, there are still several areas and features in which more efforts are required to improve the presented results. Consequently, in this section several ideas about future research actions are presented and, in some cases, an explanation of the steps that should be taken.

a) Dynamic self-generation adaptive rules. Although AdaptUI provides three different sets of rules, these rules are static, and they always depend on the corresponding developer to be designed. This means that rules have to be added in a previous stage, when users are still not present. In the near future we aim to provide to AdaptUI an extra module capable of generating and adapting at runtime different rules considering several context change triggers. Thus, from a default set of rules, these rules could be personalized taking into account the user experience.

b) Further *Pellet4Android* evaluation and tests are needed. Although it provides promising results running in Android devices, deeper processing experiments are needed. More specialized datasets should be used to compare the results.

c) AdaptUI basically covers disabilities related to visual and hearing sensory limitations. A further study and analysis of these and more disabilities, based on the ICF, would include more users and better adaptations of the user interfaces.

d) Use users with actual enduring disabilities.

e) Translate the models to JSON-LD [20], which is a JSON based Linked Data format.

[1] http://www.viainteligente.com/premios2012.html

f) Adapt the API, reaching a more object oriented paradigm. For instance, declaring an individual and being able to modify it with class methods would be desirable than the current approach, in which there is just a method to declare it through the AdaptUI object. Listing 21 shows an example of the mentioned changes.

```
public void adapt(){
  AdaptUI adaptUI = new AdaptUI(namespace, views);

  // Current API
  adaptUI.addClass(namespace, className);
  adaptUI.addIndividualMembership(namespace,
                indName, className);
  adaptUI.addDataTypePropertyValue(namespace,
                indName, dataPropName, value);
  adaptUI.addObjectProperty(namespace,
                indName1, indName2, objPropName);

  // Future API
  Individual ind = adaptUI.createIndividual(namespace);
  ind.putClass(className);
  ind.putName(indName);
  ind.putDatatypeProperty(dataProperty);
  ind.putObjectProperty(objProperty);
}
```

Listing 21: Projected changes in the API.

g) Including the usability metrics in the Modelling Layer might help to identify the user needs in the current context. This will also allow to avoid the necessity of using the Capabilities Collector. In the current version it is mandatory to use it to collect several user preferences. By anticipating the intervention of the usability metrics the interaction monitoring process might produce significant preferences results.

h) Integrate the proposed framework with the paradigm followed by Marmolin et al. [114], through which it would be possible to distinguish between superficial and deep customizations depending on the context situation. This could be used to enrich the functionalities of the Adaptation Polisher module.

6.5 Final Remarks

This dissertation is the result of years of research including several areas within the Adaptive User Interface field. With it, I have tried to contribute with significant improvements

considering the current state of the art. Besides, I deem that these contributions will further help a more widespread adoption of adaptive user interfaces design for users with disabilities. I also hope that this thesis will inspire and contribute to the scientific community, as others have undoubtedly inspired me.

Bibliography

[1] The fundamental importance of keeping an ABox and TBox split. http://www.mkbergman.com/489/ontology-best-practices-for-data-driven-applications-part-2/. Accessed: 2014-06-02. xx, 134, 135, 137

[2] Android activities. http://developer.android.com/guide/components/activities.html. Accessed: 2014-25-09. 93

[3] Accessibility in android. http://developer.android.com/guide/topics/ui/accessibility/index.html, . Accessed: 2014-08-06. 22

[4] Android view. http://developer.android.com/reference/android/view/View.html, . Accessed: 2014-21-08. 97

[5] Android audiomanager. http://developer.android.com/reference/android/media/AudioManager.html, . Accessed: 2014-06-06. 153

[6] Androjena: Jena android porting. https://code.google.com/p/androjena/, . Accessed: 2014-2-09. 104

[7] Cc/pp information page. http://www.w3.org/Mobile/CCPP/, . Accessed: 2014-08-08. 55

[8] Cc/pp current status. http://www.w3.org/standards/techs/ccpp, . Accessed: 2014-08-08. 56

[9] The cobra ontology. http://cobra.umbc.edu/ontologies.html. Accessed: 2014-29-08. 65

[10] Comparing Device Description Repositories. http://www.samaxes.com/2012/10/comparing-device-description-repositories/. Accessed: 2014-08-12. xix, 59

[11] Device independence and content adaptation. http://www.w3.org/standards/webofdevices/independence. Accessed: 2014-08-08. 56

[12] The friend of a friend (foaf) project. http://www.foaf-project.org/. Accessed: 2014-29-08. 63

[13] The Imhotep framework. http://www.morelab.deusto.es/imhotep/index.html. Accessed: 2014-05-27. xvii, xxiii, 3, 142, 143, 144

[14] Accessibility in ios. https://www.apple.com/accessibility/ios/,. Accessed: 2014-08-06. 22

[15] Siri for ios. https://www.apple.com/ios/siri/,. Accessed: 2014-08-06. xv, 23

[16] Voiceover for ios. https://www.apple.com/accessibility/ios/voiceover/,. Accessed: 2014-08-06. xv, 23

[17] The package javax.xml.bind. http://docs.oracle.com/javaee/5/api/javax/xml/bind/package-summary.html. Accessed: 2014-25-09. 104

[18] Introduction to jaxb. http://docs.oracle.com/javase/tutorial/jaxb/intro/. Accessed: 2014-25-09. 104

[19] A free and open source java framework for building semantic web and linked data applications. https://jena.apache.org/. Accessed: 2014-25-09. 103

[20] Json for linking data. http://json-ld.org/. Accessed: 2014-08-06. 190

[21] Wikipedia: Lux. luminance provided under various conditions. http://en.wikipedia.org/wiki/Lux#Illuminance. Accessed: 2014-12-09. xix, 79

[22] Measuring usability with the system usability scale (sus). http://www.measuringusability.com/sus.php. Accessed: 2014-07-15. 172

[23] Mumms: Measuring the usability of multi-media software. http://www.ucc.ie/research/hfrg/questionnaires/mumms/index.html. Accessed: 2014-29-09. 172

[24] Object role modeling (orm). the official site for conceptual data modeling. http://www.orm.net/. Accessed: 2014-08-08. 55

[25] Osgi™ - the dynamic module system for java™. www.osgi.org. Accessed: 2014-06-02. 47

[26] The owl api. http://owlapi.sourceforge.net/, . Accessed: 2014-25-09. 103

[27] Reasoners, OWL API support, papers about the OWL API. https://github.com/owlcs/owlapi/wiki/Reasoners,-OWL-API-Support,-papers-about-the-OWL-API, . Accessed: 2014-25-09. 15

[28] Pellet: Owl 2 reasoner for java. http://clarkparsia.com/pellet/, . Accessed: 2014-06-04. 102

[29] Pellet4android. https://github.com/edlectrico/Pellet4Android, . Accessed: 2014-29-09. 102

[30] Protégé: A free, open-source ontology editor and framework for building intelligent systems. http://protege.stanford.edu/. Accessed: 2014-25-09. 103

[31] Quis: Questionnaire for user interaction satisfaction. http://lap.umd.edu/quis/. Accessed: 2014-29-09. 172

[32] Android sharedpreferences. http://developer.android.com/reference/android/content/SharedPreferences.html. Accessed: 2014-29-09. 99

[33] Statista: Number of smartphone users in the U.S. from 2010 to 2018 (in millions). http://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/. Accessed: 2014-20-08. xvi, 89

[34] Sumi: The de facto industry standard evaluation questionnaire for assessing quality of use of software by end users. http://sumi.ucc.ie/. Accessed: 2014-29-09. 171

[35] Sus: System usability scale. http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. Accessed: 2014-29-09. 170

[36] Tactus. http://tactustechnology.com/. Accessed: 2014-08-07. 26

[37] Android text-to-speech. http://developer.android.com/reference/android/speech/tts/TextToSpeech.html. Accessed: 2014-25-09. 97

[38] Uaprofindex. http://www.w3.org/wiki/UAProfIndex. Accessed: 2014-08-08. 55, 57

[39] DEC VT100 video terminal. http://en.wikipedia.org/wiki/VT100. Accessed: 2014-07-15. xvii, 170, 171

[40] Wurfl - the mobile device database. http://wurfl.sourceforge.net/. Accessed: 2014-08-07. 25, 58

[41] The apache xerces project - xerces.apache.org. http://xerces.apache.org/. Accessed: 2014-25-09. 105

[42] Software engineering - product quality, ISO/IEC 9126-1. Technical report, International Organization for Standardization, 2001. xv, 9, 10

[43] *BS 7000-6:2005 Design management systems. Part 6: Managing inclusive design.* Guide. British Standards Institute, 2005. 15

[44] Google Glasses. http://www.google.com/glass/start/, 2014. Accessed: 2014-05-20. xv, 7

[45] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In Hans-W. Gellersen, editor, *Handheld and Ubiquitous Computing*, number 1707 in Lecture Notes in Computer Science, pages 304–307. Springer Berlin Heidelberg, January 1999. ISBN 978-3-540-66550-2, 978-3-540-48157-7. URL http://link.springer.com/chapter/10.1007/3-540-48157-5_29. 53

[46] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, page 217–253. Springer, 2011. URL http://link.springer.com/chapter/10.1007/978-0-387-85820-3_7. 43

[47] J. F. Allen. A plan-based approach to speech act recognition. 1979. URL http://dl.acm.org/citation.cfm?id=909217. 27

[48] A. Almeida and D. López-de-Ipiña. Assessing ambiguity of context data in intelligent environments: Towards a more reliable context managing system. *Sensors*, 12 (4):4934–4951, 2012. URL http://www.mdpi.com/1424-8220/12/4/4934. 44, 50, 53, 54

[49] A. Almeida, P. Orduña, E. Castillejo, D. López-de-Ipiña, and M. Sacristán. Imhotep: an approach to user and device conscious mobile applications. *Personal and Ubiquitous Computing*, 15(4):419–429, 2011. 3, 25, 58, 141

[50] W. Babisch. The noise/stress concept, risk assessment and research needs. *Noise and health*, 4(16):1, 2002. URL http://www.noiseandhealth.org/article.asp?issn=1463-1741;year=2002;volume=4;issue=16;spage=1;epage=11;aulast=Babisch. 33

[51] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.H. Lüke, and R. Schwaiger. Incarmusic: Context-aware music recommendations in a car. In *E-Commerce and Web Technologies*, page 89–100. Springer, 2011. URL http://link.springer.com/chapter/10.1007/978-3-642-23014-1_8. 44, 50, 52, 54

[52] J. Bauer, R. Kutsche, and R. Ehrmanntraut. Identification and modeling of contexts for different information scenarios in air traffic. *Technische Universität Berlin, Diplomarbeit*, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.7895&rep=rep1&type=pdf. 55

[53] C. Bobed, F. Bobillo, R. Yus, G. Esteban, and E. Mena. Android went semantic: Time for evaluation. URL http://ra.cps.unizar.es:8080/PUBLICATIONS/attachedFiles/document/ore2014_submission_1.pdf. 105

[54] G. Brajnik and C. Tasso. A shell for developing non-monotonic user modeling systems. *International Journal of Human-Computer Studies*, 40(1):31–62, 1994. 27, 39

[55] S. Burgstahler. Designing software that is accessible to individuals with disabilities. *Seattle, Washington: DO-IT, University of Washington-Seattle*, 2002. URL http://www3.cac.washington.edu/doit/Brochures/PDF/design_software.pdf. 2

[56] M. H. Butler. Implementing content negotiation using CC/PP and WAP UAProf. *HP LABORATORIES TECHNICAL REPORT HPL*, (190), 2001. URL http://www.hpl.hp.com/techreports/2001/HPL-2001-190.pdf. 57

[57] M. H. Butler. CC/PP and UAProf: issues, improvements and future directions. In *Proceedings of W3C Delivery Context Workshop (DIWS 2002)*, 2002. URL https://www.hpl.hp.com/techreports/2002/HPL-2002-35.pdf. 57

[58] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, N. Souchon, L. Bouillon, M. Florins, J. Vanderdonckt, et al. Plasticity of user interfaces: A revisited reference framework. In *In Task Models and Diagrams for User Interface Design*, 2002. 24, 44

[59] A. Caragliu, C. Del Bo, and P. Nijkamp. *Smart cities in Europe*. Vrije Universiteit, Faculty of Economics and Business Administration, 2009. 1

[60] S. K. Card, T. P. Moran, and A. Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, 1980. 7

[61] J. H. Carlisle. Evaluating the impact of office automation on top management communication. In *Proceedings of the June 7-10, 1976, national computer conference and exposition*, pages 611–616. ACM, 1976. 7

[62] J. J. Carroll and J. De Roo. OWL web ontology language test cases. *W3C recommendation*, 10, 2004. 103

[63] R. Casas, R. Blasco Marín, A. Robinet, A. Delgado, A. Yarza, J. Mcginn, R. Picking, and V. Grout. User modelling in ambient intelligence for elderly and disabled people. *Computers Helping People with Special Needs*, page 114–122, 2008. URL `http://www.springerlink.com/index/5085668312150127.pdf`. xv, 25, 27, 36, 37, 41, 42, 63, 64, 66

[64] E. Castillejo, A. Almeida, and D. López-De-Ipiña. Alleviating cold-user start problem with users' social network data in recommendation systems. August 2012. 52

[65] E. Castillejo, A. Almeida, and D. López-de-Ipiña. Social network analysis applied to recommendation systems: alleviating the cold-user problem. In *Ubiquitous Computing and Ambient Intelligence*, page 306–313. Springer, Vitoria-Gasteiz, Spain, 2012. URL `http://link.springer.com/chapter/10.1007/978-3-642-35377-2_42`. 52

[66] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000. xxv, 8, 44, 45, 52, 54

[67] H. Chen, F. Perich, T. Finin, and A. Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, page 258–267. IEEE, 2004. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1331732`. 60

[68] H. Chen, T. Finin, and A. Joshi. Using OWL in a pervasive computing broker. Technical report, 2005. 44, 48, 54

[69] K. Cheverst, K. Mitchell, and N. Davies. Design of an object model for a context sensitive tourist GUIDE. *Computers & Graphics*, 23(6):883–891, 1999. URL http://www.sciencedirect.com/science/article/pii/S0097849399001193. 55

[70] P. R. Cohen and C. R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive science*, 3(3):177–212, 1979. URL http://www.sciencedirect.com/science/article/pii/S0364021379800063. 27

[71] A. Cooper and P. Saffo. *The inmates are running the asylum*, volume 1. Sams, 2004. URL http://www.lavoisier.fr/livre/notice.asp?ouvrage=1480606. xxv, 36, 37

[72] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and others. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, page 293–296. ACM, 2010. URL http://dl.acm.org/citation.cfm?id=1864770. 40

[73] A. K. Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001. URL http://dl.acm.org/citation.cfm?id=593572. xxv, 11, 13, 23, 42, 43, 44

[74] P. Dourish. What we talk about when we talk about context. *Personal and ubiquitous computing*, 8(1):19–30, 2004. URL http://link.springer.com/article/10.1007/s00779-003-0253-8. 32

[75] P. Dourish. *Where the action is: the foundations of embodied interaction*. The MIT Press, 2004. URL http://books.google.es/books?hl=es&lr=&id=DCIy2zxrCqcC&oi=fnd&pg=PR7&dq=where+the+action+is+dourish&ots=oD33f4a4Vi&sig=ijawRNJJ-MUvz0gtxivpqJ8jel0. 32

[76] A. W. Drake. *Fundamentals of applied probability theory*. Mcgraw-Hill College, 1967. 28

[77] European Commission. Ageing report: Europe needs to prepare for growing older, 2012. URL http://ec.europa.eu/economy/finance/articles/structural/reforms/2012-05-15/ageing/report/en.html. 8

[78] C. Evers, R. Kniewel, K. Geihs, and L. Schmidt. Achieving user participation for adaptive applications. In *Ubiquitous Computing and Ambient Intelligence*, Lecture Notes in Computer Science, pages 200–207. Springer Berlin Heidelberg, January 2012. ISBN 978-3-642-35376-5, 978-3-642-35377-2. URL `http://link.springer.com/chapter/10.1007/978-3-642-35377-2_28`. 25, 27, 37, 41, 42, 52

[79] T. Finin and D. Drager. GUMS: a general user modeling system. In *Proceedings of the workshop on Strategic computing natural language*, page 224–230, 1986. URL `http://dl.acm.org/citation.cfm?id=1077173`. 39

[80] J. Fink and A. Kobsa. A review and analysis of commercial user modeling servers for personalization on the world wide web. *User Modeling and User-Adapted Interaction*, 10(2-3):209–249, 2000. URL `http://link.springer.com/article/10.1023/A:1026597308943`. 9

[81] J. Fink, A. Kobsa, and A. Nill. Adaptable and adaptive information access for all users, including the disabled and the elderly. In *International Conference UM97. Wien New York: Springer*, volume 171, page 173, 1997. URL `http://www.isr.uci.edu/~kobsa/papers/1997-UM97-kobsa.pdf`. 4

[82] G. Fischer. User modeling in human–computer interaction. *User modeling and user-adapted interaction*, 11(1):65–86, 2001. ISSN 0924-1868. xv, xix, xxv, 2, 13, 14, 21, 22, 27, 29, 30, 41, 77, 93

[83] S. Gauch, J. Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. *Web Intelligence and Agent Systems*, 1(3):219–234, 2003. URL `http://iospress.metapress.com/index/d68rmj5v6c897x3c.pdf`. 27, 30, 40, 42

[84] W. W. Gibbs. Considerate computing. *Scientific American*, 292(1): 54–61, 2005. URL `http://www.nature.com/scientificamerican/journal/v292/n1/full/scientificamerican0105-54.html`. 52

[85] M. Golemati, A. Katifori, C. Vassilakis, G. Lepouras, and C. Halatsis. Creating an ontology for the user profile: Method and applications. In *Proceedings of the First RCIS Conference*, page 407–412, 2007. 27, 36, 40, 42

[86] P. Gregor, A. F. Newell, and M. Zajicek. Designing for dynamic diversity: interfaces for older people. In *Proceedings of the fifth international ACM conference on Assistive technologies*, page 151–156, 2002. URL `http://dl.acm.org/citation.cfm?id=638277`. xv, 2, 27, 30, 31, 41, 42, 63, 64, 66, 71

[87] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993. xxv, 15

[88] T. Gu, H. K. Pung, and D. Q. Zhang. Toward an OSGi-based infrastructure for context-aware applications. *Pervasive Computing, IEEE*, 3(4):66–74, 2004. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1369163`. 44, 47, 48, 51, 53, 54

[89] T. Gu, X. H Wang, H. K Pung, and D. Q Zhang. An ontology-based context model in intelligent environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, volume 2004, page 270–275, 2004. 48, 51

[90] S. Hallsteinsen, E. Stav, and J. Floch. Self-adaptation for everyday systems. In *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*, page 69–74, 2004. 24

[91] M. Hatala and R. Wakkary. Ontology-based user modeling in an augmented audio reality system for museums. *User Modeling and User-Adapted Interaction*, 15 (3-4):339–380, 2005. URL `http://link.springer.com/article/10.1007/s11257-005-2304-5`. 27, 32, 40, 42

[92] D. Heckmann. *Ubiquitous user modeling*, volume 297. IOS Press, 2005. xv, 11, 38

[93] D. Heckmann, T. Schwartz, B.s Brandherm, M. Schmitz, and M. von Wilamowitz-Moellendorff. Gumo–the general user model ontology. In *User Modeling 2005*, pages 428–432. Springer, 2005. URL `http://link.springer.com/chapter/10.1007/11527886_58`. xv, 27, 33, 34, 40, 41, 42, 63, 65

[94] A. Held, S. Buchholz, and A. Schill. Modeling of context information for pervasive computing applications. In *Procceding of the World Multiconference on Systemics, Cybernetics and Informatics*, 2002. URL `http://www.rn.inf.tu-dresden.de/uploads/sci2002-paper512jh.pdf`. 44, 47, 54, 55

[95] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. *Pervasive Computing*, page 79–117, 2002. URL `http://www.springerlink.com/index/JBXD2FD5GA045P8W.pdf`. 44, 46, 51, 53, 54

[96] K. Henricksen, J. Indulska, and A. Rakotonirainy. Generating context management infrastructure from high-level context models. In *In 4th International Conference on Mobile Data Management (MDM) - Industrial Track*, page 1–6, 2003. 55

[97] R. Hervás, J. Bravo, and J. Fontecha. A context model based on ontological languages: a proposal for information visualization. *J. UCS*, 16(12):1539–1555, 2010. URL http://www.jucs.org/jucs_16_12/a_context_model_based/jucs_16_12_1539_1555_hervas.pdf. 60

[98] D. Hoch. Samsung remains king of the android market with 65% share of all android devices. http://www.localytics.com/blog/2014/samsung-remains-king-of-the-android-market/, 2014. Accessed: 2014-05-20. xvii, 136

[99] J. Hong, E. Suh, and S.J. Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509–8522, 2009. URL http://www.sciencedirect.com/science/article/pii/S0957417408007574. 75

[100] ISO, editor. *ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) – Part 9: Requirements for non-keyboard input devices*. 2000. 9

[101] ISO/IEC. *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC, 2001. xx, 109, 110, 111, 112

[102] A. Jameson. Modelling both the context and the user. *Personal and Ubiquitous Computing*, 5(1):29–33, 2001. URL http://dl.acm.org/citation.cfm?id=593578. xv, 8, 11, 23, 44, 45, 54

[103] A. Jameson. Adaptive interfaces and agents. *Human-Computer Interaction: Design Issues, Solutions, and Applications*, 105, 2009. xxv, 12

[104] J. Kay. The um toolkit for cooperative user modelling. *User Modeling and User-Adapted Interaction*, 4(3):149–196, 1994. 27, 39

[105] T. Kleinbauer, M. Bauer, and A. Jameson. SPECTER-a user-centered view on ubiquitous computing. In *ABIS 2003: 11th GI-Workshop" Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, pages 299–306, 2003. URL http://www2.dfki.de/specter/Docs/KleinbauerBA03.pdf. xv, 11

[106] A. Kobsa. Generic user modeling systems. *User modeling and user-adapted interaction*, 11(1):49–63, 2001. URL http://www.springerlink.com/index/g5g2030040352416.pdf. 8, 9, 23, 27, 39

[107] A. Kobsa and W. Pohl. The user modeling shell system bgp-ms. *User Modeling and User-Adapted Interaction*, 4(2):59–106, 1994. 39

[108] M. Krötzsch, F. Simancik, and I. Horrocks. A description logic primer. *arXiv preprint arXiv:1201.4089*, 2012. URL http://arxiv.org/abs/1201.4089. 134

[109] M. Lehtonen, R. Petit, O. Heinonen, and G. Lindén. A dynamic user interface for document assembly. In *Proceedings of the 2002 ACM symposium on Document engineering*, page 134–141, 2002. 24

[110] T. Lemlouma and N. Layaïda. Context-aware adaptation for mobile devices. In *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*, page 106–111, 2004. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1263048. 56

[111] W. Liao, W. Zhang, Z. Zhu, and Q. Ji. A decision theoretic model for stress recognition and user assistance. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL IN[U+2121]LIGENCE*, volume 20, page 529, 2005. URL http://www.aaai.org/Papers/AAAI/2005/AAAI05-083.pdf. 37, 52

[112] lux. International System of Units (SI). http://physics.nist.gov/cuu/Units/units.html. Accessed: 2015-01-25. 74, 98

[113] H. Maass. Location-aware mobile applications based on directory services. *Mobile Networks and Applications*, 3(2):157–173, 1998. URL http://www.springerlink.com/index/W5036673261V046P.pdf. 54

[114] H. Marmolin, Y. Sundblad, K. Schmidt, et al. Medium versus mechanism: Supporting collaboration through customisation. In *ECSCW'95: proceedings of the Fourth European Conference on Computer-Supported Cooperative Work, 10-14 September 1995, Stockholm, Sweden*, volume 8, page 133, 1995. 24, 191

[115] L. M. McAvoy, L. Chen, and M. Donnelly. An ontology-based context management system for smart environments. In *UBICOMM 2012, The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, page 18–23, 2012. URL http://www.thinkmind.org/index.php?view=article&articleid=ubicomm_2012_1_40_10164. 44, 50, 54

[116] T. Nelson. *The home computer revolution*. Distributors, 1977. xxv, 9

[117] E. G. Nilsson, J. Floch, S. Hallsteinsen, and E. Stav. Model-based user interface adaptation. *Computers & Graphics*, 30(5):692–701, 2006. 23, 24, 44

[118] J. Orwant. *Doppelgänger–a user modeling system*. PhD thesis, Massachusetts Institute of Technology, 1991. URL http://18.7.29.232/handle/1721.1/12952. 27, 28, 39, 40

[119] J. Orwant. Heterogeneous learning in the doppelgänger user modeling system. *User Modeling and User-Adapted Interaction*, 4(2):107–130, 1994. URL http://link.springer.com/article/10.1007/BF01099429. xxiii, 29

[120] A. Paiva and J. Self. Tagus—a user and learner modeling workbench. *User Modeling and User-Adapted Interaction*, 4(3):197–226, 1994. 27, 39

[121] F. Pereira. A triple user characterization model for video adaptation and quality of experience evaluatio. In *Multimedia Signal Processing, 2005 IEEE 7th Workshop on*, page 1–4, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4013946. 27, 33, 42, 52, 63, 71

[122] C. R. Perrault, J. F. Allen, and P. R. Cohen. Speech acts as a basis for understanding dialogue coherence. In *Proceedings of the 1978 workshop on Theoretical issues in natural language processing*, page 125–132, 1978. URL http://dl.acm.org/citation.cfm?id=980282. 27

[123] U. Persad, P. M. Langdon, and P. J. Clarkson. Exploring user capabilities and health: A population perspective. In *Proceedings of the Annual Ergonomics Society Conference on Contemporary Ergonomics' 06*, page 373–377, 2006. URL http://publications.eng.cam.ac.uk/325700/. 34

[124] U. Persad, P. Langdon, D. Brown, and P. J. Clarkson. Cognitive scales and mental models for inclusive design. In *Proceedings of the 4th international conference on Universal access in human computer interaction: coping with diversity*, page 776–785, 2007. 33, 34, 63, 71, 75

[125] U. Persad, P. Langdon, and J. Clarkson. Characterising user capabilities to support inclusive design evaluation. *Universal Access in the Information Society*, 6(2):119–135, 2007. xix, 27, 33, 35, 36, 40, 41, 42, 63, 71, 75

[126] D. Petrelli, A. De Angeli, and G. Convertino. A user-centered approach to user modelling. *COURSES AND LECTURES-INTERNATIONAL CENTRE FOR MECHANICAL SCIENCES*, pages 255–264, 1999. 4

[127] W. Pohl. Logic-based representation and reasoning for user modeling shell systems. *User Modeling and User-Adapted Interaction*, 9(3):217–282, 1999. URL http://link.springer.com/article/10.1023/A:1008325713804. xxv, 27, 28, 38, 39, 40

[128] L. Razmerita, A. Angehrn, and A. Maedche. Ontology-based user modeling for knowledge management systems. In *User Modeling 2003*, page 213–217. Springer, 2003. URL http://link.springer.com/chapter/10.1007/3-540-44963-9_29. 27, 31, 40, 41, 42, 53, 63, 71

[129] P. Repo. Facilitating user interface adaptation to mobile devices. In *Proceedings of the third Nordic conference on Human-computer interaction*, page 433–436, 2004. 24

[130] E. Rich. Building and exploiting user models. In *Proceedings of the 6th international joint conference on Artificial intelligence-Volume 2*, page 720–722, 1979. URL http://dl.acm.org/citation.cfm?id=1623079. 27

[131] E. Rich. User modeling via stereotypes. *Cognitive science*, 3(4):329–354, 1979. URL http://www.sciencedirect.com/science/article/pii/S0364021379800129. 27

[132] Z. Roupa, M. Nikas, E. Gerasimou, V.i Zafeiri, L. Giasyrani, E. Kazitori, and P. Sotiropoulou. The use of technology by the elderly. *Health Science Journal*, 4(2):118–126, 2010. URL http://www.hsj.gr/volume4/issue2/428.pdf. 11

[133] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, page 85–90, 1994. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4624429. 8, 45

[134] B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, 1994. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=313011. xxv, 13, 44

[135] B. N. Schilit, M. M. Theimer, and B. B. Welch. *Customizing mobile applications*. Citeseer, 1993. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.2550&rep=rep1&type=pdf. 54

[136] A. Schmidt, M. Beigl, and H. W. Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999. URL http://www.sciencedirect.com/science/article/pii/S009784939900120X. 44, 52, 53

[137] K. L. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, W. Burns, and I. Solheim. Ontological user profile modeling for context-aware application personalization. In *Ubiquitous Computing and Ambient Intelligence*, page

261–268. Springer, 2012. URL http://link.springer.com/chapter/10.1007/978-3-642-35377-2_36. xv, 27, 38, 40, 41, 42, 63, 71

[138] S. Sterling. Get old, tune out: Is technology leaving the elderly in the dust? 11

[139] M. F. Story, J. L. Mueller, and R. L. Mace. The universal design file: Designing for people of all ages and abilities. 1998. xxv, 12

[140] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Workshop Proceedings*, 2004. URL http://elib.dlr.de/7444/. 41, 53, 54, 55, 88

[141] W. Stuerzlinger, O. Chapuis, D. Phillips, and N. Roussel. User interface façades: towards fully adaptable user interfaces. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, page 309–318, 2006. 24

[142] G. M. Voelker and B. N. Bershad. Mobisaic: An information system for a mobile wireless computing environment. *Mobile Computing*, page 375–395, 1996. URL http://www.springerlink.com/index/MHWV733969L34757.pdf. 54

[143] W. Wahlster and A. Kobsa. *User models in dialog systems*. Springer, 1989. xxv, 32, 38, 39

[144] S. Weerawarana, F. Curbera, M. J. Duftler, D. A. Epstein, and J. Kesselman. Bean markup language: A composition language for JavaBeans components. In *Proceedings of the 6th conference on USENIX Conference on Object-Oriented Technologies and Systems-Volume 6*, page 13–13, 2001. URL http://dl.acm.org/citation.cfm?id=1268254. 24

[145] A. Wood, J. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *Network, IEEE*, 22(4):26–33, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4579768. 44, 49, 54

[146] T. Yamabe, A. Takagi, and T. Nakajima. Citron: A context information acquisition framework for personal devices. In *Embedded and Real-Time Computing Systems and Applications, 2005. Proceedings. 11th IEEE International Conference on*, page 489–495, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1541130. 44, 49, 54

[147] R. Yus, C. Bobed, G. Esteban, F. Bobillo, and E. Mena. Android goes semantic: DL reasoners on smartphones. In *ORE*, pages 46–52. Citeseer, 2013. 104, 105

Declaration

I herewith declare that I have produced this work without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This work has not previously been presented in identical or similar form to any examination board.

The dissertation work was conducted from 2010 to 2015 under the supervision of Dr. Diego López-de-Ipiña and Dr. Aitor Almeida at the University of Deusto.

Bilbao,

This dissertation was finished writing in Bilbao on Monday 16th February, 2015