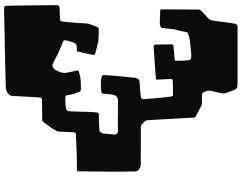
eman ta zabal zazu



Universidad del País Vasco
Euskal Herriko Unibertsitatea

Department of Computer Architecture and Technology

A FRAMEWORK FOR ABSTRACTION AND VIRTUALIZATION OF SENSORS IN MOBILE CONTEXT-AWARE COMPUTING

Candidate Borja Gamecho
(borja.gamecho@gmail.com)

Supervisors

Julio Abascal

Luis Gardeazabal

Deposit

May 2015

- ISBN: 978-84-9082-423-8

To my parents.

"The worthwhile problems are the ones you can really solve or help solve, the ones you can really contribute something to. ... No problem is too small or too trivial if we can really do something about it."

Richard Feynman (1918 - 1988)

Acknowledgements

Aprovecho estas líneas para agradecer y recordar a todas las personas y compañeros que me han ayudado directa o indirectamente en el desarrollo de esta tesis:

En primer lugar, a mis directores Luis y Julio por darme la oportunidad de empezar en el grupo de investigación. Y más importante todavía, por su apoyo y supervisión durante mi etapa de doctorando. Sin vuestra paciencia y constancia, esta tesis no habría llegado a buen puerto.

Al resto de los miembros de Egokituz, con los que he compartido reuniones y seminarios, por todo lo que he podido aprender de vosotros y vuestro trabajo. En especial me gustaría destacar al "Egoki Team": Amaia, Idoia y Raúl, por los momentos compartidos durante el desarrollo de Egoki y el proyecto INREDIS, junto a vosotros aprendí el significado de la palabra equipo. No me olvido tampoco de Ana, Carlos, Christian, Edu, Myriam, Nestor, Unai B., Unai M., Xabi G., Xabi V., Zigor ... por los buenos ratos vividos en el laboratorio 303.

I also would like to express my gratitude to the people of the BIT team in Lisbon, and specially to Hugo Silva and Prof. Ana Fred for giving me the chance to do a research stay in their group. André, José, Priscilla, Carlos, David, etc. this thesis would be very different without those months in Lisbon. You give me the opportunity to challenge myself. Muito obrigado!

A los participantes de los experimentos por dedicarme parte de vuestro tiempo tanto en Lisboa, como en Donosti. En especial a Marcos, MariFe, Maritxin y Ramón por ofrecerse voluntarios para probar el sistema. También a la asociación de jubilados Jatorra por las facilidades y predisposición que habéis puesto durante los experimentos sin esperar nada a cambio.

A los doctorandos y doctores de otros grupos de la facultad que he tenido la suerte de conocer desde que empecé el Máster SIA. Habéis conseguido que el buen humor y las risas se abran paso entre los agobios y *deadlines* que no hemos dejado de superar. Estoy seguro de que aunque no os mencione uno por uno, acabaremos celebrando esta tesis como se merece.

A mis padres, a mi tía, y a mi cuadrilla al completo por haber estado disponibles cuando os he necesitado y por entenderme cuando he tenido que estar más ausente.

Borja Gamecho ha sido beneficiario del Programa Predoctoral de Formación de Personal Investigador No Doctor del Departamento de Educación, Política Lingüística y Cultura del Gobierno Vasco desde el año 2011 al 2014, habiendo desarrollado esta tesis durante ese periodo.

Borja Gamecho held a PhD scholarship from the Research Staff Training Programme of the Basque Government from 2011 to 2014. This PhD Thesis was carried out during the scholarship time.

This work has been supported by the Department of Education, Universities and Research of the Basque Government under Grant IT395-10, by the Ministry of Economy and Competitiveness of the Spanish Government and by the European Regional Development Fund (project TIN2014-52665-C2-1).

ABSTRACT

The latest mobile devices available nowadays are leading to the development of a new generation of mobile applications that are able to react to context. Context-awareness requires data from the environment, usually collected by means of sensors embedded in mobile devices or connected to them through wireless networks.

Developers of mobile applications are faced with several challenges when it comes to the creation of context-aware applications. Sensor and device heterogeneity stand out among these challenges. In order to assist designers, we propose a layered conceptual framework for sensor abstraction and virtualization, called Igerri. Its main objective is to facilitate the development of context-aware applications independently of the specific sensors available in the user environment. To avoid the need to directly manage physical sensors, a layered structure of virtual and abstract sensors is conceived.

Two software components, based on the proposed framework, have been designed in order to test Igerri's robustness. The first one processes the information from the successive sensor layers and generates high-level context information. The second is responsible for managing network aspects and real time settings. This implementation has been tested using a representative context-aware application in different scenarios. The results obtained show that the implementation, and therefore the conceptual framework, is suitable for dealing with context information and hiding sensor programming.

RESUMEN

Los dispositivos móviles disponibles actuales facilitan el desarrollo de una nueva generación de aplicaciones móviles que son capaces de reaccionar al contexto. La computación sensible al contexto requiere datos del entorno que normalmente se obtienen por medio de sensores embebidos en dispositivos móviles o conectados a ellos a través de redes inalámbricas.

Los desarrolladores de aplicaciones móviles se enfrentan a varios retos para crear aplicaciones sensibles al contexto. Entre estos retos destaca la necesidad de tratar la heterogeneidad de los sensores y de los dispositivos móviles. Con el fin de ayudar a los desarrolladores, esta tesis propone un marco conceptual para la abstracción multinivel y la virtualización de sensores, llamado Igerri. Su principal objetivo es facilitar el desarrollo de aplicaciones sensibles al contexto independientemente de los sensores específicos que se encuentren en el entorno. Para evitar la necesidad de manipular directamente los sensores físicos, se ha concebido una estructura multinivel de sensores virtuales y abstractos.

Se han diseñado dos componentes software basados en el marco propuesto para comprobar la robustez de Igerri. El primero procesa la información de la estructura multinivel de sensores y genera información de contexto de alto nivel. El segundo es responsable de administrar, en tiempo real, las opciones de red y la configuración de los sensores. Esta implementación ha sido probada usando una aplicación representativa, sensible al contexto y en diferentes escenarios. Los resultados obtenidos muestran que la implementación, y por tanto el marco conceptual que le da soporte, es adecuada para tratar la información de contexto y ocultar los problemas de programación de los sensores.

LABURPENA

Gaur egungo gailu mugikor puntakoenek inguruneari erantzuteko gai diren aplikazio mugikorren garapenean oinarritzen dira. Testuingurua nabaritzeko ingurunearen informazioa behar da, zeina gailu mugikorretan txertatutako sentsoreen edo haririk gabeko sareen bitartez biltzen den.

Aplikazio mugikorren garatzaileek erronka askori aurre egin behar izaten diete testuingurua kontuan hartzen duten aplikazioak garatzerakoan. Erronka nagusien artean, sentsoreen eta gailuen heterogeneotasuna izaten dira. Garatzaileei laguntzeko asmoz, Igerri izeneko sentsoreen abstrakzio eta birtualizaziorako marko kontzeptual bat proposatzen dugu. Bere helburu nagusia, testuinguruaren aplikazio hautemangarrien garapena erraztea da, erabiltzailearen ingurunean dauden sentsore espezifikoak edozein direla ere. Sentsore fisikoak zuzenean manipulatu behar izatea saihesteko, sentsore birtual eta abstraktuen egitura bat asmatu da.

Igerri-ren sendotasuna egiaztatzeko, proposatutako markoan oinarritutako bi software osagai diseinatu dira. Lehenak, sentsore geruzen informazio geruzak prozesatu eta maila altuko testuinguru informazioa ematen du. Bigarrenak, sare aukerak kudeatu eta sentsoreen konfigurazioa denbora errealean burutzen ditu. Inplementazio hau testuingurua hautemateko gai eta adierazgarria den aplikazio batekin egoera desberdinetan frogatu da. Lortutako emaitzek erakusten dute inplementazioa, eta ondorioz marko kontzeptuala ere, aproposa dela testuinguruaren informazioa erabiltzeko eta sentsoreen programazioa ezkutatzeko.

Contents

Figures

Tables

Chapter 1

Introduction

In this chapter, the motivations and research interests for the present work are summarized. For this purpose, a general introduction to the topic and a description of a previous work named EGOKI user interface generator is presented. Finally, a number of research questions this thesis seeks to answer are detailed.

1.1 Ubiquitous Computing

The proliferation and popularization of mobile and small electronic devices over the last decade is changing the shape of ITC technology. The main representative of this change is the smartphone, a mobile device with enhanced sensing capabilities and novel interaction features. Encouraged by the success of smartphones, the industry is also leading the introduction of wearable computers, e.g. smartwatches or wristbands. This illustrates that in the near future interaction between human users and computer programs will take place with more than one computer at the same time. This scenario was forecast by Mark Weiser in the early 90's [77] and it was coined as the 'The Third Wave of Computing' or 'Ubiquitous Computing Era' [78].

Ubiquitous computing, also called Pervasive computing [66][1], is a multidisciplinary field of computer science that studies the relationship between human users and smart environments abundant with embedded computers. Computers, seamlessly integrated with the environment, help users to complete tasks in the real world. To this end: reliable networks between the computers, mobility of the user and proactive computer programs are required.

[1] Although some authors support that there are differences between Ubiquitous Computing, Pervasive Computing and Ambient Intelligence (Aarts 2009) [4], other authors insist that these names refer to the same concept (Dourish 2004) [27]. The discussion about terminology is out of scope of this dissertation and throughout this work we consider the terms: Ubiquitous Computing, Pervasive Computing, Context-Aware Computing and Ambient Intelligence as synonyms.

Not only that, but also a number of sensors must be spread out over the environment to increase the sensing capabilities of the computer systems. To fulfill these requirements, a series of challenges have been proposed and studied: heterogeneity, scalability, mobility, context-awareness and context-management among others [65, 66].

1.2 Perception and Context-Awareness Computing

A subject of great interest in pervasive computing is the ***perception of context*** in the real physical environment by an ubiquitous system, thereby leading to Context-information inference [3, 18, 21, 65].

Perception in ubiquitous computing can be defined as the capability of a system to acquire context-information in a smart environment [65]. To achieve this, computers distributed in the environment have to be aware of what is happening. To this end, sensors are a valuable asset with which to obtain data from the real world [18]. Using these sensor measurements, applications can obtain appropriate context information. With this information, the interaction of the user with the application can be improved. In this way, developers create Context-Aware applications.

Context-Awareness is a central topic of this work. As stated by Dey 2001 [25] in his definition, "an application is Context-Aware if the interaction between the user and the application is affected by relevant information related to the entities of that context". Usually these entities are people, objects and the environment where the interaction takes place.

Due to the success of smartphones, Context-Aware applications became evermore popular in recent years. The basis of their success is the clever use of the embedded sensors such as GPS or the accelerometers to feed context-aware mobile applications.

For instance, the combination of GPS with other of features of the smartphones benefits navigation assistants. According to the work of Hervás et al. [44], it is possible to provide suitable context information to people with mild cognitive impairments while navigating. In the same way, as stated in Fontecha et al. [30], data collected by accelerometer can be combined with clinical information records to obtain assessment for the elderly frailty detection.

One of the consequences of the interest and success in these kind of applications is the growth in the number and type of sensors included in mobile devices, a good sample of new devices and sensors can be seen in Swan's 2012 work 'Sensor Mania!' [74].

There is enough related work concerning the topic of obtaining context information from sensors in mobile devices, from the classic work of Smith et al. (1999) [70] to the recent Wiese et al. (2013) [79].

However, when developers want to create context aware applications, they need to deal with several requirements:

- Data of the sensors must be accessed while dealing with wireless networks and different sensor specifications.
- Processing techniques and algorithms to get the context information usually are not easy to program.
- The context change in real time and the applications must detect these changes.

Several frameworks contributing to deal with these issues has been proposed in the past, for instance the Context Toolkit [23] was a very influential framework for building Context-Aware applications. More recently, AWARE framework [28] contributed to the instrumentation of the smartphone to build context aware applications. Frameworks are useful tools for developers to deal with context information.

1.3 Virtualization and Abstraction of Sensors

Two interesting issues regarding the use of sensors in smart devices full ecosystems can be underlined.

- The combination of diverse sensor outputs produces Context information (see Table 1.1). This process of processing the data from physical sensors to obtain higher level information is considered a **Virtual Sensor**. The context information obtained from virtual sensors made possible new kinds of sensor-enhanced applications and interaction paradigms.
- Equivalent context information can be obtained from different sets of sensors (see Table 1.1). We consider all the virtual sensors producing the same context information as instances of an **Abstract Sensor**. Abstraction of sensors grants independence of sensors to Context-Aware applications.

We argue that a framework to deal with Virtual and Abstract sensors can be beneficial for Context-Aware application developers.

- Virtual sensors encapsulate the algorithms to obtain context information from physical sensors. This encapsulation made Virtual Sensors reusable for different applications. They can process one or more signals to obtain higher level information, for instance, an Activity monitor (AM1) can be obtained from an Electrocardiography (ECG) and an Accelerometer (ACC). Developers only need to subscribe for this Virtual sensor and manage the context information directly.
- Abstract sensors made possible to interchange one Virtual Sensor with another similar one. When two Virtual sensors are represented with the same Abstract sensor, it is possible to interchange them. For instance,

Table 1.1: Examples of combination of physical sensors transformed into context information found in the literature

Author(s)	Physical Sensors	Context Information
Schmidt el al. [70]	Temperature, Pressure, CO Gas Meter, Photodiode, Accelerometers, PIR and Microphone	Mobile phone, User Activity
Haag et al. [43]	EMG, Electro Dermal Activity sensor (EDA), Skin Temperature, Blood Volume Pulse, ECG and Respiration	User Emotional State
Parkka et al. [57]	Air Pressure, Microphone, Accelerometer, Humidity, Luminosity, ... (up to 22 signals)	User Activity
Chon and Cha [17]	GPS, Accelerometers, Compass, BT, WiFi and GSM	Smartphone, User Activity
Wiese et al. [79]	Accelerometer, Light/Proximity, Capacitive and Multispectral	Smartphone, User Activity
Jang et al. [45]	Skin Temperature, ECG, EDA and Photoplethysmography (PPG)	User emotional state
Reddy et al. [61]	Accelerometer, GPS	Transportation Modes

another Activity monitor (AM2) can use a the GPS and a Blood Volume Pulse (BVP) sensor to get context information similar to AM1. Developers do not need to know which Activity monitor are using (AM1 or AM2), they only need to subscribe to the Abstract Sensor for Activity monitor and delegate the instantiation to the framework (See Figure 1.1).

Two operations must be considered in the framework to provide advanced support for abstract and virtual sensors :

- **Virtualization**: It refers to the operation of obtaining higher level information processing one or more physical sensors.

- **Abstraction**: It refers to the operation of selecting a sensor from all the physical and virtual sensors that produces equivalent output data.

In order to assist the development of Context-Aware applications regarding the Abstract and Virtual sensors, this thesis proposes the *Igerri* conceptual framework. Chapter 3 extends the ideas introduced in this section describing the conceptual framework in depth.

In the following section, we contextualize the origin of Igerri in Egoki, a previous research project of Egokituz laboratory.

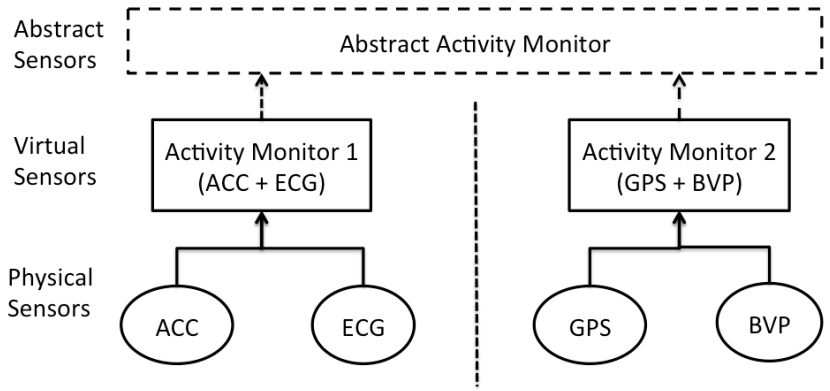


Figure 1.1: Example of Virtual and Abstract sensors.

1.4 Egoki: Ubiquitous Computing in the Egokituz Laboratory

Egoki is a user interface generator designed for Ubiquitous Systems. Its main goal is to generate remote and accessible user interfaces for people with special needs. These user interfaces are adapted to the abilities of each user. In this way, users can operate services, such as information kiosks or vending machines, if they are supported by ubiquitous computing [6].

EGOKI allows ubiquitous services to be accessed by means of user-tailored interfaces, running on accessible mobile devices. The user-adapted interfaces are based on Web technologies (e.g. HTML, JavaScript and CSS) and are accessible from the Web browser of the personal device [7]. Through the Web browser, all the personal devices with the proper assistive technologies (such as screen-readers or screen magnifiers) can interact with the ubiquitous services. Therefore, not only a specific user interface was required for each ubiquitous service, but also accessibility must be ensured for each different user and device. To achieve this, Egoki uses a model-based paradigm [53, 54]. To store these models Egoki provides a Knowledge Base named EGONTO.

Unlike other approaches that use the original version of the user interface to generate and adapt one for the user [5], Egoki built the user interface from the following models:

- **User model**: it defines the abilities and preferences of the users in the Ubiquitous Environment.
- **Device model**: it characterizes the device that renders the user interface.
- **Adaptation model**: it describes how and which information stored in the models is used for the adaptation
- **User interface model**: it details the interaction elements with the services and the available multimedia resources. It uses a user interface Mod-

elling Language (UIML) format, a XML based user interface description language [36].

These models provide valuable information about the interaction context with the ubiquitous services: the appropriate multimedia resources, navigation schemes and adapted interaction techniques. All this information was used to deliver a suitable user interface to the users.

1.4.1 Requirements for Egoki Systems

In order to successfully generate the user interfaces, the Egoki System needs: A middleware for remote interfacing the Ubiquitous System, and the provision of UIML files and multimedia resources for each Ubiquitous System.

For the first requirement, the URC/UCH middleware [82] was adopted [31]. This middleware layer was implemented as an external server that adds functionality to the user interfaces generated by Egoki.

For the second requirement, we provided tools to assist Ubiquitous Services designers in the creation of the UIML code [55].

1.4.2 Evaluation of Generated User Interfaces for Egoki

Egoki was tested in different scenarios:

- Scenario 1 [54]: A remote controller for a television using a personal mobile computer in the context of the INREDIS project.
- Scenario 2 [6]: A metro ticket vending machine accessed by blind people with their personal mobile computer (laptop).
- Scenario 3 [37]: Two Ubiquitous Services provided to people with cognitive impairments: A meal selection service in a canteen and an interactive bus timetable. They were evaluated with two user samples.
    1. Cognitive impaired people interacting with a Tablet.
    2. Blind people using their mobile computer or smartphone.

1.4.3 From Egoki to Igerri

The results of the evaluation of Egoki were successful but also raise a number of questions to be addressed in future works:

> *(Q1) What are the limitations of Egoki when it comes to generating new interaction modalities? Would it be possible to add gesture support in the current Egoki System?*

We proved that Egoki can generate different output modalities such as: image or audio at the same time. But for input modalities only the touchscreen and point and click paradigms were tested, for gesture recognition and techniques alike, a detection mechanism for more input modalities is required.

> *(Q2) How good is an adapted user interface in a mobile context? For instance, a noisy environment affects the usability of a user interface based in audio modality, how can Egoki deal with this?*

The whole context of the interaction is not fully considered, just some parameters for the user (interaction ability), and some device features (e.g. screen size or the availability of touch-screen). Thus, Egoki manages static context information that does not change while the interaction takes place. In mobile contexts, the environment usually is changing and this affects the quality of the interaction with the user interface. Moreover, sometimes the user can experience temporal restrictions due to environmental noise (loss of hearing ability) or be confused with a foreign language (degradation of his/her cognitive skills). Egoki should be able to generate adaptive user interfaces to improve the interaction in such dynamic contexts.

> *(Q3) Ubiquitous computing proposes changes in the interaction loop, with proactive applications able to modify their response without explicitly asking the user. This would be translated to applications that are not controlled with a device. Would Egoki be able to manage this kind of interaction model?*

This question is related to the previous one. As Egoki only generates the user interface, it does not manage the interaction. This interaction is controlled by a Web browser and the Middleware layer. For instance, it is impossible for Egoki to notice if the user is walking or standing still, or if the device is in his or her pocket or on a table. In both cases, the interaction should be different. To support this dynamic interaction model, mechanisms to react to the context must be *injected* in the Web based user interfaces.

> *(Q4) In the Ubiquitous Computing field, since the smartphone emergence, there has been a shift from remote applications to mobile native applications (commonly called apps). Can Egoki be adapted to generate user interfaces for these mobile apps?*

Unfortunately, the design of the Egoki systems is tied to the existence of remotely accessible ubiquitous services and it is not easy to use it for this purpose.

One of the foundations of Egoki is the generation of user interfaces for ubiquitous services. It was a legacy requirement of the INREDIS project, a research work started in 2007 and finished in 2010. In that period, context-aware computing was not so '*smartphone centric*', due to this fact, the INREDIS project was focused on positional commerce: sporadic and opportunistic services tied to the location of the user. Simultaneously, smartphone use and application stores started to grow and currently, these are widely used on a daily basis. People show great interest in standalone applications that run locally. These

applications take advantage of the context information inferred from the sensors in smartphones. Having said this, Egoki should try to generate user interfaces for this kind of applications.

From the analysis of the above mentioned questions, we concluded that a context model is necessary to enhance the perception in Egoki. A first approach to the Egoki extension for context-awareness, called context server, was described in our work for the 2013 Mobile Accessibility Workshop [32](See Figure 1.2).

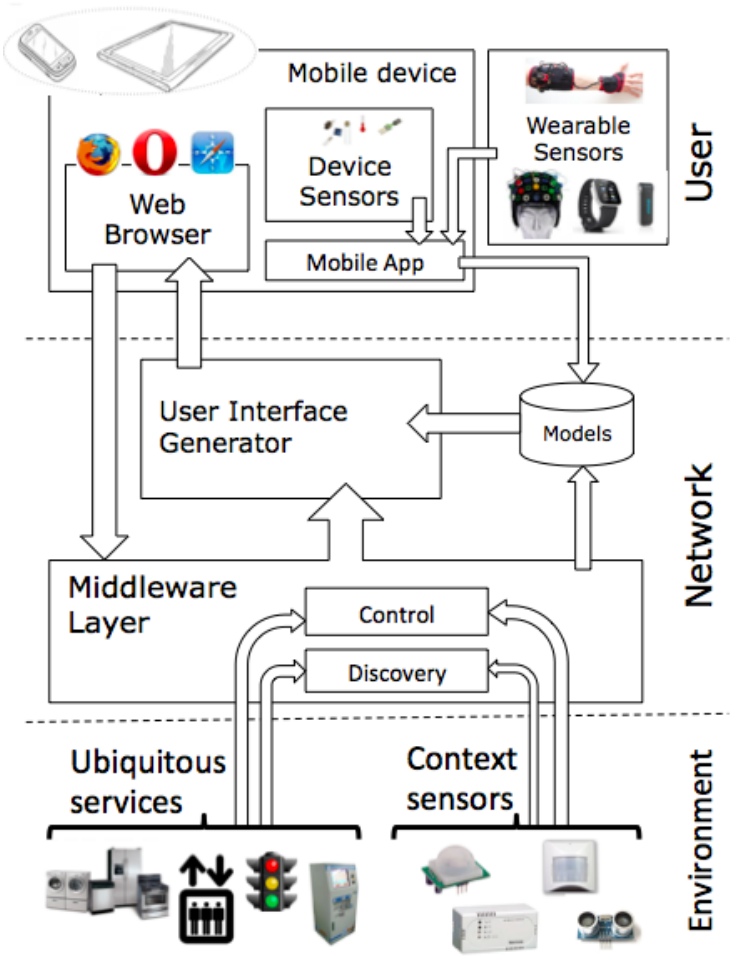


Figure 1.2: First approach to the context extension for Egoki

To answer Q4 a major redesign of Egoki would be needed or the creation of a complete new user interface generator. Nonetheless, we firmly believe that it is worth the effort to create an external standalone component. It should have two components:

1. To serve as an extension of the user interfaces generated by the Egoki system (from Q1-3).

2. To be able to manage native context-aware applications for the most popular mobile platforms (from Q4).

In this way, Egoki would be able to support both Ubiquitous Services and standalone apps.

From the previous analysis we deduce the following requirements for the context component:

- (R1) It has to manage data from sensors embedded in the environments smart devices, providing the abstraction and virtualization of sensors.
- (R2) It has to produce standalone applications with sensor driven interaction techniques using web technologies, to be extensible to Egoki.
- (R3) It has to ease the development effort of developers, and be able to work as a rapid-prototyping framework.

This was the original foundation and motivation for the Igerri framework, the creation of both, an extension to Egoki, and, a sensor abstraction and virtualization framework for mobile context-aware applications. In addition to that, with Egoki being a starting point for this thesis work, another reason for Egoki and Igerri to be decoupled is that we wanted to work in a self-contained original piece of research. In the next section the research questions to be addressed with Igerri will be presented.

1.5 Research Questions and Hypothesis

A good way to add perception is the adequate use of the embedded sensors in the ubiquitous system. For that reason we propose a framework for sensor abstraction and virtualization in mobile context-aware computing.

Generally speaking, this means that we want to offer mechanisms to assist the generation of applications able to use different sensors simultaneously. This is not new in the context-aware literature. There are several frameworks for the composition of context-aware applications, but they have different motivations and goals. In our case, we want to contribute with one framework that is designed to deal with different kinds of sensors at the same time in an easy way, granting the usability of the created applications. Moreover, we want to offer a tool to assist developers in the creation of applications with sensors hiding common issues of the sensor-driven applications.

Regarding the role of developers, usually they have to address the following challenges (Cn) to program a context-aware application [33] :

- (C1) *Seamless Integration*, to achieve seamless integration of the different components in ubiquitous environments we have to face the heterogeneity challenge:
  - (C1.1) *Sensor heterogeneity*. Due to the variety of devices and manufacturers, each sensor requires different low-level management in

order to obtain the information provided.

  - (C1.2) *Platform heterogeneity*. Different mobile development platforms use incompatible Software Development Kits (SDK) making impossible to share the same code for the same application running in different target smartphones, even if they have the same set of sensors.

  - (C1.3) *Network heterogeneity*. There are different networks available in mobile devices. Each of them with different characteristics. Most of the smartphones and tablets have Wifi and Bluetooth networking properties. In addition to this, there are no methods in the automatic discovery mechanisms to identify the properties of connected devices and to access the sensor information.

- (C2) *Context recognition*. Developers devote considerable efforts to recognizing activities happening in the mobile phone context. For each activity the available data has to be analysed and models to match these activities have to be designed and trained.

- (C3) *Performance*. Mobile devices usually make a trade-off between power consumption rate and processor activity. The proliferation of independent sensor readings and heavy processing algorithms run by different applications critically affect the performance of the battery.

It would be really helpful if the sensor abstraction and virtualization framework could provide mechanisms to deal with these issues. The design and implementation of the framework will take into account theses issues.

Aside from the development aspect, we want to focus on testing whether the usability of the created applications is appropriate or if they contain flaws. We consider users as the best test-bed for Mobile Context-Aware applications. If users feel comfortable with the applications the sensor integration should be well suited and the performance flawless. For this reason, the following research questions are proposed:

> **Research Question 1.** *Can Igerri produce functional and usable applications with its implementation of the mobile Context-Aware framework ?*
>
> **Research Question 2.** *Are the users able to control sensor enhanced applications following the Igerri approach ?*
>
> **Research Question 3.** *Do the users perceive sensor enhanced Igerri applications as appealing, engaging and/or of added value ?*

On the whole, these questions lead to the following hypothesis: *The abstraction and virtualization of sensors as presented in Igerri are valid techniques to develop usable Context-Aware applications.*

Igerri is designed to provide clear insight into the above mentioned research questions and hypothesis.

1.6 Research Process

To start, the revision of the work carried out with Egoki [6, 36, 37, 54] pointed out the need of a context-aware module and inspired this work.

Literature in context-aware computing was reviewed and adequate research questions were set after deciding to focus on the abstraction and virtualization of sensors.

Next the interest in the research addressed with this thesis was contrasted with three position papers for different research communities: Mobile Accessibility [32], Pervasive Computing [33] and Human Computer Interaction [34]. The first approach to the proposed system is depicted in Figure 1.3).

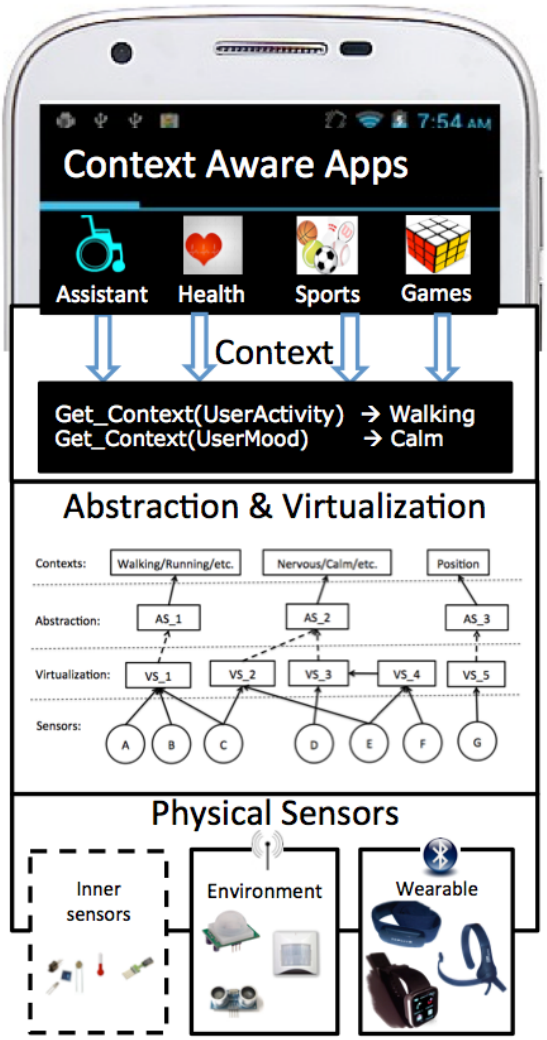


Figure 1.3: Summary of the elements in the Abstraction and Virtualization Framework

After collecting all the feedback, a conceptual framework named Igerri was established. Later Igerri was instantiated with an implementation divided in two components: MobileBIT, which was devoted to sensor abstraction and virtualization and PervasiveBIT, which is dedicated to conceal networking and sensor discovery issues.

When the implementation was ready, representative applications were designed and implemented. The usability of the applications were tested with a total of 29 users. The conclusions for this work were obtained using the results from the usability evaluation.

1.7 Conclusion

This chapter sets the basis of the research that has been conducted in the topic of sensor abstraction and virtualization. In the next Chapter, the state of the art is described and related work is summarized. In Chapter 3, the conceptual framework with all the required concepts to resolve these problems is explained. In Chapter 4, a reference implementation of the abstraction and virtualization of sensors is depicted. In Chapter 5, the evaluation of the previous implementation is described. Finally Chapter 6, conclude this thesis work and describes the validation of the hypothesis and the contributions to the mobile context-aware area.

Chapter 2

Background and Related Work

This chapter provides background in the area of the topics covered by this thesis. Previous and significant works are introduced as part of the state of the art and related work is presented.

2.1 Introduction

This thesis is located in the field of Ubiquitous Computing, and more precisely within the topic of Context-Awareness. Its objective is to contribute with a framework for sensor abstraction and virtualization for mobile context-aware computing. Thus we focus on how to discover and extract context information using the data collected by sensors.

A general overview of the Ubiquitous Computing and Context Awareness was briefly introduced in the first chapter. Nevertheless it is worth mentioning some works in this area in order to better understand the research work proposed in this thesis.

2.2 Ubiquitous Computing

Ubiquitous computing is a multidisciplinary field of research, born out of the evolution of distributed computing and mobile computing [66]. Other areas such as human-computer interaction, expert systems and software agents are also influential in ubiquitous computing.

The tasks forming the challenges to be overcome for ubiquitous computing to become a reality were an important research driver for the ubiquitous computing field. From year 2000 to 2003, coincident with the tenth anniversary of

the field, several seminal papers [9, 22, 65, 66] established the research agenda for ubiquitous computing. Challenges such as: Effective use of smart spaces, Scalability, Heterogeneity, Integration, Invisibility, Perception, Smartness, etc. were described.

These works agreed on identifying the discovery and management of context information as being one of the main challenges. This process involves the term *perception* and it is commonly related to Context-Awareness.

2.3 Perception

From a psychological perspective, human perception is the process of noticing and understanding the stimulus energies (e.g. light or sound) from the environment. Perception is closely related with sensation, which is the process by which these stimulus energies are received [81]. The perception of the physical world is carried out by means of sensory receptors. They are able to detect and transmit to the brain information originating from a stimulus. In other words, humans can see, hear, taste and smell thanks to the sensory receptors, and perception takes place when those stimuli are organized and interpreted by the brain.

An interesting approach to explain how perception works is the perceptual process described by Goldstein [42]. This process starts with a stimulus in the environment of the person and ends with a behavioural response in the subject. This response involves perceiving, recognizing and taking actions. Goldstein identifies perception and recognition as part of a mental process assisted by previously existing knowledge.

This perceptual process can be imitated by computers systems. The sensors are connected to devices with computational power in order to analyse the data collected and to transform them into context-information. Research areas such as Artificial Intelligence and Robotics have their own uses of the perception concept. However, for this thesis we focus on the Ubiquitous Computing background. Perception has been described for this area in several seminal works as it is a key concept for the ubiquitous computing.

Schmidt identified perception as a part of his influential work about implicit human computer interaction [68]. When a person interacts with a computer an explicit interaction is expected. The computer expects to receive commands to be operated in a certain way. Implicit interaction is the additional information that a computer can understand which is not primarily aimed for interaction. There are two main concepts for Implicit interaction, namely perception and interpretation. Schmidt's vision foresees devices with the ability to see, hear and feel. "Based on their perception, these devices will be able to act and react according to the situational context in which they are used".

The work of Saha and Mukherjee [65] identified perception as being as much of a challenge for the field of pervasive computing as other challenges

such as, Scalability, Heterogeneity or Invisibility. For these authors, perception is equivalent to Context-Awareness. They identify issues derived from implementing perception: location monitoring, uncertainty modelling, real-time information processing and merging data from multiple, and possibly disagreeing, sensors. The perception challenge is complemented with another concept called Smartness or Context Management, which is defined as the means of using the perception effectively.

Cook et al. [18] described perception as part of the sensing process for the Ambient Intelligence (AmI). A variety of sensors can be used to achieve the Perception. Software algorithms perceive the environment using these sensors. After the sensing stage, the reasoning process is carried out.

Coutaz et al. [21] describe perception as part of the abstraction for a general-purpose infrastructure for Context-Aware computing. They locate the perception layer between the sensing layer and the situation and context identification layer.

Finally, Aarts and Wichert [3], describe a 3-step process to obtain context-information for perception of the situation in AmI. For them, perception is again related to the context-awareness. The first step, called sensing, relies on wireless sensor networks to gather information. The second step entails, among others, the processing of information, data combination, classification or sensor fusion. Finally, the third step is the interpretation of the contextual information to obtain information on a higher semantic level.

We have found different definitions for the concept of perception in the above mentioned works. Some of them use it as a synonym of Context-Awareness, but in most cases they refer to the process of acquiring data from sensors and to transform it into high level contextual information. Usually, perception is the previous stage to the interpretation, reasoning or similar processes which take place after the perception of the context-information.

Igerri aims to provide a comprehensive vision of perception and to support the process of transforming sensor information into usable context-information. With the proposed framework, the ubiquitous computing system devices have the capability of using sensors to transform the stimuli of physical environment including the users and objects in that location into high level context-information. This is possible, firstly through virtualizing sensors able to perceive context-information, and secondly by achieving sensor abstraction.

2.4 Context-Aware Systems

The concept of perception is part of the Context-Awareness property of Ubiquitous Computing. Usually systems that focus on Context Awareness are called Context Aware Systems.

Probably, the first work relating context with ubiquitous computing was the project Active Badge [76]. A project to locate people in office environments

using wearable badges as beacons. Nevertheless it was only after the work of Shcilit et al. [67] when Context Aware Computing was defined, inspired the Ubiquitous Computing. The authors introduced a handheld wireless device called PARCTAB able to react to changes in the environment. Four categories for context applications were defined based on whether the task was obtained information or activating a command, and on whether the command was effected manually or automatically. The first context aware systems were strongly linked to location information.

Many authors tried to contribute with useful Context definitions for Context-Aware Systems [14, 64, 67]. Anyway, this thesis assumes a popular definition of Context stated by Dey in 2001 [24]:

> " Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and a application, including the user and application themselves". — *Anind K. Dey (2001).*

The same author defined a Context-Aware System as follows,

> " A system is context-aware if it uses context to provide relevant information and or services to the user, where the relevancy depends on the user's task". —*Anind K. Dey (2001).*

Context-Aware systems are usually composed of a set of networked devices which share context-information acquired from the embedded sensors. Then, Context-Aware applications require access to that context-information in order to adapt the interaction. All the entities involved in the interaction can be identified by the developer when it is designing the application. These entities are then characterized by the context-information available in the context-aware systems. For this purpose, sensors are a valuable source of context-information.

2.5 Mobile Phones in Context-Aware Computing

Mobile Phones have been used for Context Aware Computing since its origins. They can be seen as the instantiation of the original devices envisioned by Weiser in the foundational works of Ubiquitous Computing [77]. These devices are very useful because they are carried by potential users of Context Aware environments and they have embedded diverse sensors that enable the acquisition of context. One of the advantages of having multiple sensors in the same device with different functionalities is the possibility of combining several sources of data to obtain higher-level context information. A plethora of works have been proposed following this approach. Over time, the complexity and number of sensors included in mobile devices has increased.

One of the first works presenting this approach was the TEA project (Tecnology for Enabled Awareness) by Schmidt et al. [70]. The TEA project pro-

posed a self-contained multi-sensor device connected to a Nokia 6110 mobile phone using a serial interface. This work describes the fusion of eight sensors (temperature, pressure, CO gas meters, a photodiode, two accelerometers, a passive IR, and a microphone) to recognize different means of transport. Additionally, it detects whether the mobile phone is in the hand, in the suitcase or on the table, among other contexts. With this context information, the mobile phone swaps between different preprogrammed profiles affecting the notification of calls: the volume of audio alarms, vibration, or the use of silent mode.

Korpipää et. al [49] published a very interesting work regarding the use of ontologies. They created an ontology for managing context information in mobile devices and introduced a framework for context-aware application development for Nokia series 60 smartphones (Symbian platform) [48]. This system uses a central node to store the context information following the blackboard model proposed by Winograad [80]. Context information related to Environment, Device placement and user activity was obtained by means of accelerometers, audio, light, temperature and touch sensors.

Since the arrival in 2007 of the iPhone followed by Android Smartphones in 2008, the area of Mobile Context Aware computing started to use these devices intensively. Modern Smartphones contributed to the general use of the GPS to provide outdoor location and inertial sensors for movement acquisition. For instance, LifeMap [17] is an application able to recognize the context within which a smartphone is being used by means of regular sensors (GPS, accelerometers, compass, etc.). It can detect whether the user is walking, running, etc.

Other approaches propose to add new advanced sensors to obtain context-information. As an example *Phoneprioception* [79] studies the ability of smartphones to identify where they are (bag/pocket/hand/etc.). To do this, it combines four sensors: accelerometer, light/proximity, capacitive and multispectral. The last two sensors are prototypes incorporated specifically for this purpose in the smartphone.

2.5.1 Wearable Devices and Physiological Signals

In recent years, the sensing capabilities of modern smartphones have been extended with wireless wearable devices. Due to advances in body area networks, miniaturization and the affordability of advanced sensors for physiological computing, these devices now provide opportunities to discover new context information regarding physiological signals such as:

- Heart beats and heart rate can be acquired by means of diverse techniques such as: Electrocardiography (ECG) or Photoplethysmography (PPG).
- Muscles activity can be perceived by means of Electromyography (EMG) and Acceleremoters (or Inertial Measurements Units) attached to body limbs.

- Arousal level is obtained by means of Electrodermal Activity (EDA), Galvanic Skin Response (GSR) or Skin Temperature sensors (ST).
- Other physiological signals such as Respiration rate (RR), can also be detected by means of diverse physiological sensors.

An interesting example of these devices is BITalino, a wireless sensor platform highly adaptable for wearable computing [72]. BITalino includes EMG, ECG, EDA, Accelerometry and Luminosity sensors in a single board. This PhD used BITalino as sensor source device in the evaluation chapter.

Background information about the user can be obtained by combining information from wearable sensors [56]. They enable the detection of physiological signals (or biosignals), which leads to the extraction of information on actions performed by the person wearing them, as shown in the survey published by Avci et al. [8]. There are plenty of applications for these external sensors, for instance Costa and Duarte [19] aim to use surface EMG sensors to improve accessibility for blind people with mobile devices.

A different approach is to use physiological wireless sensors to provide *Mobile E-Health* services and applications. For instance, the work of Villareal et al. [75] propose an architecture for the medical control of chronic diseases and presents a case study for patients with diabetes. Similarly, mHealthDroid [11] follows a smartphone centered approach and provides a framework to create smartphone applications for the medical domain.

Additionally, user's emotions can also be detected. For instance, the work by Haag et al. [43] identified the emotional states of a user in terms of valence and arousal, by combining a set of biosignals: EMG, EDA, ST, Blood Volume Pulse sensors (BVP), ECG and RR. Similarly, the work of Jang et al. [45] classified three negative emotions (fear, surprise and stress) from four biosignals: ST, ECG, EDA and PPG.

2.5.2 Sensors in Context-Aware Computing

Sensors are one of the enabling elements to generate context information, and consequently to support Context-Aware information. We focus on sensors that obtain information from physical stimuli. We call them Physical Sensors. A definition of Physical Sensors can be found at Kalantar et al. [46],

> " A sensor is a device which responds to stimuli (or an input quality) by generating processable outputs. These outputs are functionally related to the input stimuli which are generally referred to as measurands". —*Sensors: An Introductory Course (2013) [46].*

Beyond this definition we can also obtain a first approach of how a sensor works,

> " A sensor is commonly made of two major components: a sensitive element and a transducer. The sensitive element has the capability to interact with a target measurand and cause a change in the

> operation of the transducer. Affected by this change, the transducer produces a signal, which is translated into readable information by a data acquisition system... The processing information is then sent to a processing system, where it is processed into meaningful information". —*Sensors: An Introductory Course (2013) [46].*

2.5.3 Sensor Categorization Regarding Context Entities

Sensors can be classified following their relationship with the entities in the interaction context. In this way, it is possible to categorize which sensors are useful to gather context-information about the: 1) environment, 2) objects and 3) people, involved in the context-aware application. With this information, the context-aware applications can adapt the interaction with the user.

- **Environment**. A number of environmental variables can be gathered using sensors. E.g. the level of light in a room can be measured to adapt the brightness of a smartphone apps. The noise level can also be measured using a microphone and therefore applications can use this context-information to change the modality of an audio user interface to a more suitable one.
- **Object**. Several context-aware applications require information about certain objects in order to work properly. These objects usually include embedded sensors to gather this information. The most common object considered in the context-aware domain is the electronic device used to access the context-aware application, namely the Smartphone/Tablet. E.g. By means of inclinometer sensors embedded in a smartphone, it is possible to know what it is the position of the display and switch the visualization of apps between landscape or portrait layouts.
- **Person**. Context-aware applications can require information about people involved in the interaction with the application. The main person considered for this purpose is the user. It is possible to gather information about the user activity, emotional state, and the position. For these purposes, both the smartphone and wearable devices are used to obtain data. Context-aware applications can use activity information to improve the interaction with the user, e.g. avoid displaying certain notification messages if the user is running.

More examples are depicted in table 2.1,

In Igerri we consider all sensors characterize entities. In some cases, the same sensor type can be used to characterize different entities in different applications. This is the case of the inertial sensors embedded in smartphones. They can be used to obtain context information about both the position of the smartphone (in the hand, in the pocket) and the user activity (walking, running, resting). Therefore the framework must be flexible enough to avoid always matching a specific sensor with the same entity. Some sensors using electrodes depend on the location of the electrode to obtain context-information

Table 2.1: Different physical stimulus measured with sensors

Characterized Entity	Measurand	Sensor Name
Environment	Temperature	Thermometer
	Humidity	Humidity sensor
	Pressure	Barometer
	Lux	Luminosity sensor
Object	Movement	Accelerometer
		Gyroscope
	Magnetic fields	Magnetometer
Person	Heart Rate	Electrocardyograph
	Muscle Activation	Electromyograph
	Arousal	Electrodermal Activity Sensor
	Movement	Accelerometer
		Gyroscope

about a specific limb. The developers of context-aware applications have to specify how to set up the devices and sensors before starting the context-aware application.

2.5.4 Sensor Categorization Regarding the Communication Interface

When it comes to communication of the sensors with the system, we can find different classes:

- **Simple sensors**. Sensors that are sold separately and need to be connected to a microcontroller. Usually these sensors are wired and need to be physically connected to an input pin or a bus of the microcontroller. So far, we can identify two kind of connections:
  - **Pin based**: They are wired to analog or digital input pins. For the former they have to be connected to an Analog to Digital Conversor (ADC) to obtain binary values. For the latter, they directly produce binary values.
  - **Bus based**: They use a specific communication bus. Some sensors attached to a basic microcontroller still need to be connected to a more powerful microcontroller to manipulate the data. This kind of sensor uses communication protocols and buses, such as the serial port (RS-232), I2C, SPI, etc.
- **Sensor devices** They are also called sensor platforms. They have all the

components that enable the the sensors to be used. We can distinguish between two types of sensor devices:

- **Network based**: Some sensors require a networking interface. For instance, Bluetooth or WiFi.

- **API based**: Other sensors are embedded in the device where the context is used. Usually, these sensors are accessible through the systems API's.

In Figure 2.1 we can see different communication interfaces. From left to right: Pin-based accelerometer (require 3 analog inputs), Bus-based distance sensor (I2C), Network-based sensor device called SensorDrone (using Bluetooth Classic Protocol), and finally an API-based sensor device (a regular Android smartphone).

Figure 2.1: Example of sensors with different communication interfaces

We can find works in the literature that use all kind of sensors in context-aware computing. Originally, almost all works were based on simple sensors connected to diverse platforms, from prototype boards to smartphones. In recent years, most works features the sensors embedded in smartphones and wearable devices. Nevertheless, in Igerri we want to consider any sensor which is able to get context-information. This way, Igerri has mechanisms to gather data from sensor platforms and also from simple sensors connected to an embedded system. We decided to collect all the data in API based devices. To achieve this, simple sensors are part of an embedded platform connected to API based devices thereby becoming network-based sensors. Network sensors send the information using a wireless network to the API based device.

2.6 Related Work

2.6.1 Context Widgets

Dey's Conceptual Framework [25] and its implementation, the Context Toolkit, is one of the seminal works in the topic of Context Aware Frameworks because of the identified problems and provided solutions. This work identified and established the features necessary for architectural support to Context-Aware applications: context specification, separation of concerns, context interpretation, transparent distributed communications, constant availability of context acquisition, context storage and resource discovery. One of the most interesting contributions of the conceptual framework is the architectural building blocks. A reusable set of components which are implemented in the Context Toolkit were introduced:

- **Widgets** collect information about the environment through the use of sensors.
- **Interpreter** transforms low level context information into high level.
- **Aggregators** provide applications with related context about an entity.
- **Discoverer** locates context components that are of interest to applications.
- **Services** are responsible for changing the environment using actuators.

The Functional Blocks presented in MobileBIT which are an implementation of the Igerri framework, are influenced by the Building Blocks and Widget approach. While Dey's work provides a full structural and architectural support for Context Aware applications, Igerri specialised in the Sensor Abstraction and Virtualization.

2.6.2 Computing in Context

Schmidt's *Ubiquitous Computing, Computing in Context* [69], provides a perception architecture for Context-Aware systems. This layered architecture has three levels:

- *Sensor Layer* composed of both physical and logical sensors. While physical sensors are similar to the ones used for this PhD, logical sensors are components that provide information about the real world but do not take it from the environment (e.g. a server offering the current exchange rate).
- *Cue Layer* provides an abstraction of the sensor layer. A cue is described as a function with values of single sensors up to a certain time as the input, and a symbolic output. Each sensor has its own cues and can have more than one cue. The cues implement *perception methods*, a kind of

processing methods based, among others, on statistical functions, time domain analysis and rule based systems.

- *Context Layer* is composed of conditions with which to evaluate the cue layer. It contains relevant context descriptions for a particular application. This layer can support learning capabilities to change contexts over time based on adaptive algorithms.

Igerri layered framework is comparable to the perception architecture. Firstly, the sensor layer is equivalent to Igerri's physical sensor layer. Logical sensors are not considered in Igerri, but they can be implemented using the Source Blocks of MobileBIT. As far as the Cue and Context Layer are concerned, these are comparable with Igerri's virtual sensor layer and context service layer. But they are more constrained due to the fact that Igerri's abstract sensor layer enables the interchange of similar virtual sensors which contains the same information.

2.6.3 BeTelGeuse

According to Kukkonen et al. [50]: "BeTelGeuse is an extensible data collection platform for mobile devices which automatically infers higher-level context from sensor data.". BeTelGeuse is designed to be multiplatform, extensible for new kinds of sensing devices, data accessible, high-level context extraction and takes into account the user experience. It uses a blackboard model to access the context information [80]. BeTelGeuse considers four types of sensors, external Bluetooth sensors, integrated phone sensors, software sensors and internet sensors. These sensors can be extended using Context Parsers, abstraction of sensors that read and parse data, and write it on the blackboard. This platform was implemented for Nokia S60 devices.

Regarding the abstraction of sensors, all the processing is managed by the Context Parsers, while Igerri divides it in different layers. The most interesting feature is the extensibility of the framework. The implementation of Igerri allows the addition of new Functional Blocks with different purposes in a similar way.

2.6.4 AWARE Framework for Mobile Instrumentation

Ferreira proposed a mobile instrumentation toolkit called AWARE [28] in order to study the human behaviour, routines and context. AWARE is aimed to assist researchers, developers and users. It is focused in four activities: sensing context, storing context, sharing context and using context. Aware has an eight layer theoretical architecture from which two layers are relevant to this thesis: the sensing layer and the context layer.

The sensing contains three types of sensors:

- hardware-based: They can be embedded in the mobile or externally connected.
- software-based: They include network data sensors, algorithm-based sensors, and a derivative of sensor-fusion
- human-based: They collect data directly asking the user

The Context layer, abstracts the data and produces context. This layer contains reusable add-ons called Context Sensors that use techniques such as data mining or machine learning to obtain higher-level contexts. Context sensors work unattended and are not operated by users. AWARE framework is composed of a client application developed for Android and a Server to store and reuse context data with other external sensors and applications, installable in a web server.

AWARE supports more types of sensors apart from Igerri physical sensors and the layered framework takes several concerns into account. Nevertheless, Igerri has implicit support for substitution sensors with the abstraction of sensors.

2.6.5 A Pluggable Middleware Architecture

The work of Paspallis et al. [58] presents a pluggable and modular middleware architecture for developing context-aware mobile applications. It uses components with the roles of context providers (abstractions for sensors) and context-consumers (applications). The context plugins are designed to be reusable and the context management is dynamically composed by the middleware layer. A very interesting property is that the middleware is able to activate or deactivate context plug-ins as needed. When an application is started it registers for certain context types and then the middleware checks if there are available plugins to obtain this context and activate them on demand.

Despite not being focused on the sensor abstraction, this work shares similar properties to the MobileBIT implementation. Using a DPL generated with PervasiveBIT it is possible to obtain similar behaviour in the applications. Nonetheless, our conceptual framework is designed to separate concerns more accurately, due to the Virtual Sensor and Abstract sensor layer.

2.6.6 mHealthDroid

mHealthDroid [11] is the implementation of a mobile health framework intended for agile development of applications. Despite being restricted to the domain of mobile health, it was considered for the related work revision due to the capabilities of obtaining and visualizing context information. The architecture of the framework is composed of several modules called managers. Communication manager and data processing managers are the most relevant to this PhD:

- The communication manager abstracts the applications from the underlying health technologies. This level is composed of adapters, standalone modules intended to manage the connection with biomedical devices and deals with data acquisition.
- The data processing Manager provides signal processing, data mining and machine learning techniques. These tasks are split into four independent modules: preprocesing, segmentation, feature extraction and classification.

The MobileBIT's source blocks share a common goal with the communication managers. Similarly the MobileBIT's normal blocks are used for similar tasks to those of the data processing manager.

2.6.7 Ghiani et. al's Context Server

In Ghiani et al. work [39] a context-dependent multimodal augmentation engine is presented for web applications. This system is able to react to the context and adapt in real time to the modality of the visited web. The context acquisition is done by means of a Context Server and Context delegates:

- Context Delegates can be deployed in smartphones or in other devices in the environments and communicates the sensor data to the context server.
- Context sever is subscribed to the delegates and processes the data to obtain the context information. Once a change in the context information is detected the modality of the user interface can change.

This work is the perfect example of what can be achieved with Egoki when it is combined with Igerri. The main difference is the lack of mechanisms with which to transform sensor data in the smartphone. Following our sensor abstraction approach it would not be necessary to send sensor data to the server in order to discover changes in the context.

2.7 Conclusion

In this chapter the state of the art in sensing technologies, context-awareness and mobile context-aware computing provide the background for this research. The work developed in this thesis was compared with similar research and the differences and advantages of Igerri and its implementation have been pointed out.

Chapter 3

Igerri Conceptual framework

In this chapter, a conceptual framework for sensor abstraction and virtualization is proposed. For that purpose, this chapter outlines the concepts necessary to understand the approach and the definition of the system model under consideration. In addition, the formal model of the system is described to define the relationship of the sensors with the context-information in an abstract way.

3.1 Introduction

Virtual machines have been used in diverse fields, such as Operating Systems and Computer Architecture, to structure and model complex computers. The main idea is to create a hierarchy of machines, each one able to run its specific set of commands. These machines are virtual because they cannot run their language by themselves. Thus, each machine rewrites or translates its own commands into the language of the next lower machine, maybe adding some required contextual information. The task comes down the hierarchy until reaching the lowest layer machine, the hardware, the only machine able to run its own language. Among the many advantages of this structure, two stand out:

- *Abstraction*: more powerful and abstract languages can be designed. The only condition is to be able to translate them into a lower lever language run by the next virtual machine.
- *Independence* of each machine from the lower layer virtual machines. A virtual machine can be replaced only by rewriting the interfacing programs that translate from a machine to the next one.

Since our objective is to design a framework that provides context information to context-aware applications in a simple way that ensures independence from the hardware and extensible abstraction, we proposed a multilayer

model inspired in the Virtual Machines model for computer architecture. The definition of the conceptual model (independent of the implementation) facilitates communication and mutual understanding among the development team members, helps the implementation of coherent and compatible systems and improves the modularity and reusability of the interaction software developed. In the next sections we will define the elements that compose the model, the abstraction layers, and the procedures to translate one layer to the next one. With this model we want to achieve independence of the application from the specific hardware sensors required for context awareness, and abstraction to provide the applications with more expressive context information.

3.2 Definitions

3.2.1 Sensors

Physical Sensor

A physical sensor is a device that returns a numeric value within a range, based on the state of a physical parameter.

```
Physical_Sensor (Settings, Output, Function)
```

where:

- Settings are all values and procedures required for the sensor to return a value. They may include activation parameters, programming, protocols, etc.
- Output is the current value of the sensor. To facilitate its interpretation, it can include extended information such as range, resolution, etc.
- Function is the semantic interpretation given to the numerical output. To do it, a lexicon containing the properties of all the functions available is required. This semantic interpretation is required in order to be able to progress in higher abstraction. Higher layers ignore technical details and characteristics of the sensors and concentrate in their purpose. They need enriched semantics in order to be able to do suitable matches.

Properties:

- Physical sensors are pieces of hardware and software able to produce outputs related to the status of specific physical parameters.

Virtual Sensor

A virtual sensor is an element that transforms data from one or more physical sensors into another set of numerical data.

```
Virtual_Sensor (Settings, Input, Output, Function)
```

where:

- Input is the set of physical or virtual sensors that feed data to the virtual sensor.
- Settings, Output and Function are defined as for the physical sensor.

Properties:

- Virtual sensors allow the combination of information from different sensors.
- They also allow processing the output data (e.g. applying translations or data-mining algorithms).

Abstract Sensor

An Abstract Sensor is a representation of all sensors that have the same function. Abstract sensors ignore settings, physical parameters and several other characteristics that distinguish the sensors that produce a compatible output whenever measuring the same physical parameter.

```
Abstract_Sensor (Output, Function)
```

where:

- Output and function are defined as for physical and virtual sensors.

Properties:

- Abstract sensors are independent of the hardware.
- They allow the designer to use a sensor without worrying about their specific characteristics.

3.2.2 Hierarchy of Layers

The proposed sensor layers are structured in the following hierarchy (see Figure 3.1):

Virtual Layer of Context Service (VLCS)

The Context Service is a layer that takes data from abstract sensors and produces predefined context information within a specified range described in a repository named “Parameters”. It allows developers to use previously specified specific context properties without directly processing data of sensors. It

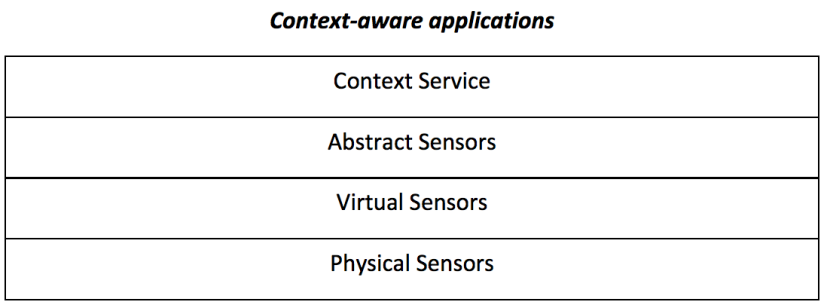


Figure 3.1: Sensor layer hierarchy

also allows applications to easily share context information. In this way applications can be simultaneously context consumer and provider. It requires information such as the names of the diverse context sources (Function) and how the abstracts sensors are matched to virtual sensors. This information is contained in the Parameters repository.

Virtual Layer of Abstract Sensors (VLAS)

The designers of the sensor structure that supports the context service can only access a virtual layer that contains abstract sensors, to avoid programming the specific sensors available in a device. The context request made from the Context Service layer always refers to abstract sensors, which are described in the Parameters repository.

Virtual Layer of Virtual Sensors (VLVS)

Abstract sensors are matched with virtual sensors that collect and process information from physical sensors and other virtual sensors using the descriptions contained in the Parameters repository.

Physical Layer of Physical Sensors (PLPS)

This contains the actual sensors installed on a particular device and its environment. They are able to collect contextual information. A physical sensor may be composed of a transducer that converts the physical parameter to an electrical signal, a conditioner which adjusts the values of the electrical signal to be processed, and an analog/digital converter that samples the electrical signal and assigns a numeric value within a range to each sample.

3.3 Transformations

An important characteristic of the virtual multilayer structure is that interactions are only allowed between adjacent layers and through well defined interfaces. Therefore, since interactions between non-adjacent layers are not allowed,void transformations are required in order to communicate elements located in non-adjacent layers. In our framework, the interface between virtual layers is defined in terms of transformations. An element of a layer is the result of applying a transformation to another element in an adjacent layer.

```
Element (parameters) ← Transformation_from_Layer_to_layer
   (Transformation_Parameters) [Element(parameters)]
```

where Transformation_Parameters are the data structures required for the transformation. The syntax $A \leftarrow B[C]$ indicates that the element A is obtained applying the transformation B to the list of elements C. Two types of transformations are defined: translations and requests. Translations are transformations that allow progress to be made upwards in the abstraction scheme. When applying a translation to an element a more abstract element is obtained.

```
Element(parameters) ← Translation_from_Layer_to_Layer
  (Translation_Parameters) [Element (parameters)]
```

Requests are transformations that allow progress to be made downwards in the abstraction scheme. When applying a request to an element a more concrete element is obtained.

```
Element (parameters) ← Request_from_Layer_to_Layer
   (Request_Parameters) [Element(parameters)]
```

3.3.1 Translations

Virtualization Translation

This transformation is part of the interface between the Layer of Physical Sensors and the Virtual Layer of Virtual sensors, matching physical sensors to virtual sensors. It is assisted by a repository of information (Translation_Parameters) containing the names and characteristics of the available physical and virtual sensors.

- Input: one or more physical or virtual sensors.
- Output: a virtual sensor (settings, data, function).
- Processing: Process and/or combine the sensor data input and produces numerical data within a predefined range.

$$\texttt{Virtual\_Sensor (Settings, Input, Output, Function)} \leftarrow$$
$$\texttt{Virtualization\_Translation (Translation\_Parameters) [Sensor\_List]}$$

where:

$$\texttt{Virtualization\_Translation (Translation\_Parameters)}$$
$$\equiv \texttt{Translation\_from\_VLPS\_to\_VLVS (Translation\_Parameters)}$$

$$\texttt{Sensor\_List = \{ Physical\_Sensor (Settings, Output, Function)}$$
$$\texttt{| Virtual\_Sensor (Settings, Input, Output, Function)}\}^{+}$$

that is,

- Virtualization is a Translation from the Virtual Layer of Physical Sensors to the Virtual Layer of Virtual Sensors.
- Sensor_List is a not-empty list of Physical Sensors and/or Virtual Sensors:

Abstraction Translation

This transformation is part of the interface between the Virtual Layer of Virtual Sensors and the Virtual Layer of Abstract Sensors, matching virtual sensors to abstract sensors. It is also assisted by a repository of information (Translation_Parameters) containing the names and characteristics of the available virtual and abstract sensors.

- Input: a virtual sensor
- Output: an abstract sensor
- Processing: It hides the characteristics of the input sensor.

$$\texttt{Abstract\_Sensor (Output, Function)} \leftarrow$$
$$\texttt{Abstraction\_Translation (Translation\_Parameters)}$$
$$\texttt{[Virtual\_Sensor (Settings, Input, Output, Function)]}$$

where

$$\texttt{Abstraction\_Translation (Translation\_Parameters)} \equiv$$
$$\texttt{Translation\_from\_VLVS\_to\_VLAS(Translation\_Parameters)}$$

that is,

- Abstraction is a Translation from the Virtual Layer of Virtual Sensors to the Virtual Layer of Abstract Sensors.

Context Translation

This transformation is part of the interface between the Virtual Layer of Context service and the Virtual Layer of Virtual Sensors, matching abstract sensors to context information. It uses the Translation_Parameters repository to match them.

- Input: an abstract sensor
- Output: semantically enriched context information
- Processing: It selects the abstract sensor that better matches the semantically enriched context information required.

$$\texttt{Context\_Information (Output, Function)} \leftarrow \texttt{Context\_Translation (Translation\_Parameters)[Abstract\_Sensor (Output, Function)]}$$

where

$$\texttt{Context\_Translation (Translation\_Parameters)} \equiv \texttt{Translation\_from\_VLCS\_to\_VLVS(Translation\_Parameters)}$$

that is,

- Abstraction is a Translation from the Virtual Layer of Abstract Sensors to the Virtual Layer of Context Service.

3.3.2 Requests

Actualize Request

This transformation is part of the interface between the Layer of Physical Sensors and the Layer of Virtual sensors, matching virtual sensors to physical sensors. It also requires a repository of information (Translation_Parameters) containing the names and characteristics of the available physical and virtual sensors.

- Input: a virtual sensor.
- Output: one or more physical or virtual sensors.
- Processing: A request for the current data collected by one or more physical sensors is made. To this end, the routines established in Settings are used.

$$\texttt{Input [Sensors\_list]} \leftarrow \texttt{Actualize\_Request (Request\_Parameters, Virtual\_Layer) [Virtual\_Sensor (Settings, Input, Output, Function]}$$

where

```
Actualize_Request (Request_Parameters) ≡
Request_from_VLVS_to_VLPS (Request_Parameters),
```

that is,

- Actualize is a Request from the Virtual Layer of Virtual Sensors to the Virtual Layer of Physical Sensors
- Sensor_List is a not-empty list of Physical Sensors

Instantiate Request

This transformation is part of the interface between the Layer of Virtual Sensors and the Layer of Abstract Sensors, matching abstract sensors to virtual sensors. It also requires the information contained in a repository of information (Translation_Parameters) containing the names and characteristics of the available virtual and abstract sensors.

- Input: an abstract sensor.
- Output: one or more virtual or abstract sensors.
- Processing: A request for the current data collected by one or more virtual sensors is made. To this end, the routines established in Settings are used.

```
Input [Sensors_list] ← Instantiate_Request
(Request_Parameters) [Abstract_Sensor (Output, Function)]
```

where

```
Instantiate_Request (Request_Parameters) ≡
Request_from_VLAS_to_VLVS (Request_Parameters)
```

that is,

- Instantiate is a Request from the Virtual Layer of Abstract Sensors to the Virtual Layer of Virtual Sensors
- Sensor_List is a not-empty list of Virtual Sensors

Contextualize Request

- Input: A context concept.
- Output: one abstract sensor.
- Processing: A request for the current data associated to a context concept and collected by an associated abstract sensor is made.

```
Abstract_Sensor (Output, Function) ← Contextualize_Request
 (Request_Parameters) [Context_Information (Function)]
```

where Context_Information (Function) is a request to receive (or subscribe to a service) providing information about specific context characteristics defined by "function", referred in the "Parameters" repository. This information can contain multiple data, such as: name, value, time-stamp, etc.

3.4 Independence between Virtual Layers

Virtual layers (VL) are interconnected via well defined interfaces (in this case, "transformations"). Any layer can be replaced by a different one, provided that the appropriate transformations are introduced. For example, if the system changes to a different device with different physical sensors (VLPS), only the transformations that provide the interface with the upper layer have to be changed. The rest of the layers and structures remain the same.

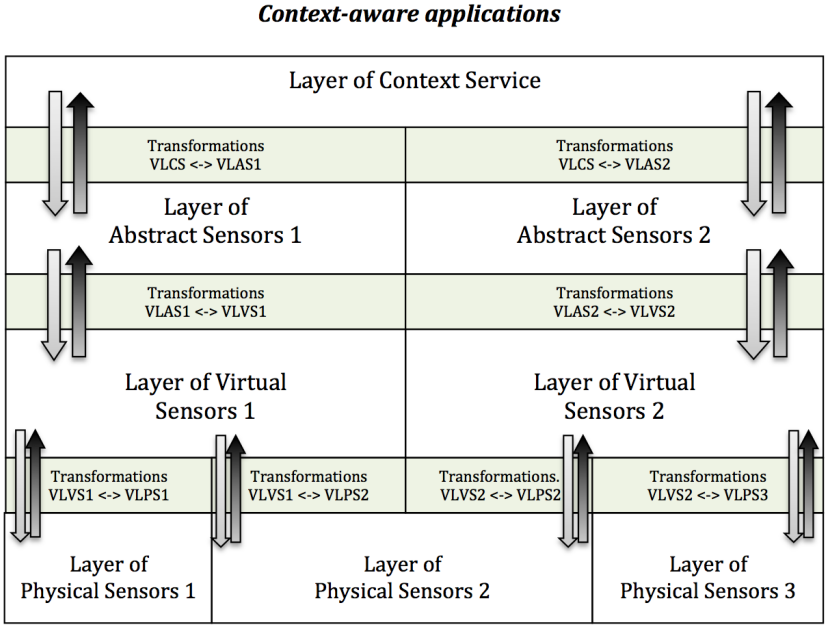


Figure 3.2: Transformations between layers

This abstraction scheme allows:

1. To define virtual sensors from physical sensors and/or other virtual sensors. This allows combinations of physical parameters to be made. Also to process data coming from a sensor or present the incoming data differently.

2. To define abstract sensors that represent all the sensors that are equivalent to each other for a specific purpose. This allows the application programmer to design independently from the physical devices and their drivers.

3. To create context through a service layer that provides information on context characteristics previously defined and delimited.

Therefore, to design actual implementations it is only necessary to create data structures that adequately represent the sensors and applications that perform the translations defined.

3.5 Example

Figure 3.3 depicts an example of abstraction and virtualization using Igerri. The Physical Sensor Layer in this system has three physical sensors: a gyroscope (that outputs a numerical value representing the angle turned respect axis X, Y and Z), an accelerometer (that outputs data representing the acceleration in the three axes X, Y, Z) and a GPS that returns the geographical coordinates and time.

In the Virtual Sensor Layer three virtual sensors have been created:

- Movement_1 obtains data from two physical sensors: gyroscope and accelerometer, and produces one of four values: Stopped, Walking, Running, Driving. Its semantic function is "Movement".

- Movement_2 obtains data from the physical sensor GPS, and produces one of four values: Stopped, Walking, Running, Driving. Its semantic function is also "Movement".

The definition of these sensors is done by means of the Virtualization Translation that describes the Settings, Input, Output and Function of the virtual sensor in terms of the physical sensors. But virtual sensors can also be created from other virtual sensors or from combinations of physical and virtual sensors. For instance:

- Device_Activity obtains data from the virtual sensor Movement_1 and a physical sensor Light. After processing the input data, it produces one of two values: active, inactive. Its semantic function is "Device_Activity".

Similarly, an abstract sensor Movement has been created in the Abstract Sensor Layer. Its input comes from the virtual sensor Movement_1. The definition of this sensor is done by means of the Abstraction Translation that describes the Output and the Function of the abstract sensor in terms of the virtual sensor. This sensor produces one of four values: Stopped, Walking, Running, Driving. The Abstraction Translation uses the data stored in the repository Translation Parameters to select one of the two sensors (Movement_1 and Movement_2) that produce equivalent data.

Virtual Layer of Context Services	**Context_Information** (Output, Function) • *Function*: "User-Activity"	
VLCS <-> VLSL	Context_Information (O, F) ← **Context_Translation** (T_P) [Virtual_Sensor_Speed (S, I, O, F)]	Input [Abstract Sensor_Movement] **Contextualise_Request** (R_P) ← Output [(Context-information (F)]
Virtual Layer of Abstract Sensors	**Abstract Sensor_Movement** (Output, Function) • *Output*: {Stopped, Walking, Running, Driving} • *Function*: "Movement"	
VLAS <-> VLVS	Abstract_Sensor_Movement (O,F) ← **Abstraction_Translation** (T_P) [Virtual_Sensor_Movement_1 (S, I, O, F)]	Input [Virtual_Sensor_Movement_1 (O, F)] **Instantiate_Request** (R_P) ← Output [Abstract_Sensor_Movement (O, F)]
Virtual Layer of Virtual Sensors	**Virtual_Sensor_Movement_1** (Settings, Input, Output, Function) • *Settings*: {sensor programming routines} • *Input*: Physical_Sensor_Gyroscope • *Output*: {Stopped, Walking, Running, Driving} • *Function*: "Movement" **Virtual_Sensor_ Movement_2** (Settings, Input, Output, Function) • *Settings*: {sensor programming routines} • *Input*: Physical_Sensor_GPS and Physical_Sensor_Acelerometer • *Output*: {Stopped, Walking, Running, Driving} *Function*: "Movement" **Virtual_Sensor_Device_Activity** (Settings, Input, Output, Function) • *Settings*: {sensor programming routines} • *Input*: Physical_Sensor_Light and Virtual_Sensor_Movement_1 • *Output*: {active, inactive} • *Function*: "Device_Activity"	
VLVS <-> VLPS	Virtual_Sensor_Movement_1 (S, I, O, F) ← **Virtualization_Translation** (T_P) [Physical_Sensor_Gyroscope (S,O, F), Physical Sensor_Accelerometer (S, O, F)]	Input [Physical_Sensor_Gyroscope (S,O, F), Physical Sensor_Acelerometer (S, O, F)] ← **Actualize_Request** (Request_Parameters) Output [Virtual_Sensor_Movement_1 (S,I,O,F)]
Physical Layer of Physical Sensors	**Physical_Sensor_Gyroscope** (Setting, Output, Function) • *Settings*: {sensor programming and managing routines} • *Output*: Angle turned numerical value in degrees • *Function*: "Angle" **Physical Sensor_Accelerometer** (Settings, Output, Function) • *Settings*: {sensor programming and managing routines} • *Output*: X, Y, Z acceleration numerical values in m/s² • *Function*: "Acceleration" **Physical Sensor_GPS** (Settings, Output, Function) • *Settings*: {sensor programming and managing routines} • *Output*: X, Y, Z geographic coordinates *Function*: "Position" **Physical Sensor_Light** (Settings, Output, Function) • *Settings*: {sensor programming and managing routines} • *Output*: a numeric value representing lighting *Function*: "Light"	

Figure 3.3: Example of virtualization and abstraction using Igerri

The context translation defines how the information provided by the Movement abstract sensor can be used by the Context Service level to build the con-

text information named User-Activity that can be requested by context-aware applications.

When a context-aware application requests specific context information, in this case Status, to the Context Service, a Contextualize Request collects this information from the abstract sensor Movement. To this end it requires information about matching functions (in this case "User-Activity" to "Movement") that is stored in a repository named Request Parameters.

In order to be able to produce updated data, the abstract sensor Movement obtains information from the virtual sensor Movemet_1 by means of an Instantiate Request. The Instantiate Request can decide to collect these data from the equivalent sensor Movement_2, with the same semantic function. The information required (about matching functions) to take this decision is stored in the repository Request Parameters.

Similarly, the virtual sensor Movement_1 obtains data from the physical sensors Gyroscope and Accelerometer by means of an Actualize Request, also using the Request Parameters repository to identify the matching functions.

3.6 Conclusion

Application programmers usually have to deal with a large number of issues to program context-aware applications. The main goal of the Igerri conceptual framework is to provide clear mechanisms for effortless context management. The Igerri abstract model allows the translation of raw data coming from physical sensors into context information for context-aware applications running in ubiquitous systems. A set of growing abstraction layers is in charge of progressively abstracting the sensors and obtaining the data from them. These data are transformed into high layer context-information using translations, such as virtualization and abstraction. Developers only have to ensure they include a suitable call to abstract sensors in their programs. Diverse implementations based on this reference framework are possible. The formalization presented in this chapter has guided the development of MobileBIT and PervasiveBIT, described in the following chapter. MobileBIT and PervasiveBIT demonstrates the suitability and the flexibility of the proposed reference framework.

Chapter 4

Implementation of the Conceptual Framework

In this chapter the design and strategies used to instantiate the Igerri Conceptual Framework are introduced. This implementation is composed of a middleware library named MobileBIT, and a server named PervasiveBIT. It is in charge of the composition of context information within devices of a ubiquitous environment. Context-Aware applications use the mechanisms implemented in MobileBIT to request abstract context information (instead of raw data). Thus the sensor abstraction and virtualization proposed in this Thesis is achieved.

4.1 Introduction

In the previous chapter a Conceptual Framework was described. Since it is technology independent, it does not include any detail of the implementation. Therefore, in this chapter, we propose an implementation following the conceptual framework to create the Context-Aware applications.

Architecture for Igerri

Four software modules were designed as an instance of system supporting Igerri (See Figure 4.1).

1. **Sensors Module**. This module instantiates the abstract, virtual and physical sensors layers described in the Conceptual Framework. It is able to collect raw data from sensors and provide homogeneous processing regardless of the original sensor. It is implemented in the MobileBIT component.

2. **Context Module**. This is in charge of updating context-information to the Context-Aware applications subscribed to this module. This is the basic mechanism used by application developers to have access to the context-information. It is in charge of the context services of the conceptual framework. It is implemented in the MobileBIT component.

3. **Sensor Hub**. This module searches for all the sensors available in the devices connected to the pervasive network. It collects information from the sensors and devices available to assist in the configuration of the Context-Aware applications. It is implemented in the PervasiveBIT component.

4. **Knowledge Base**. This module matches the information of every device found with the sensors that are embedded in it and the context that can be inferred from it. The Knowledge base includes in the system a virtual sensors if the corresponding physical sensors are available. It is implemented in the PervasiveBIT component.

The last two software modules, 3 and 4, implement the repository of information needed for the transformations between layers in Igerri (Referred in the conceptual framework as parameters).

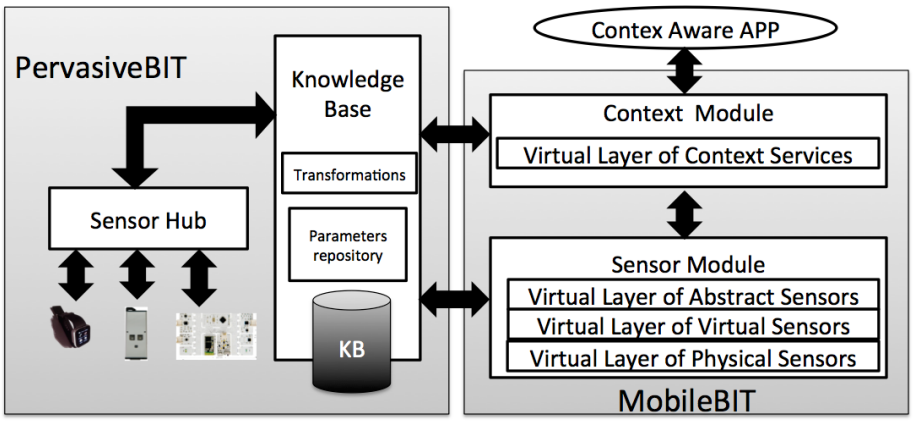


Figure 4.1: Modules that implement the conceptual framework

Implementation Design

We considered the design time and launch/run time of applications in order to instantiate the above mentioned modules. During design-time developers can use specific devices and sensors to test their applications. However, in a real ubiquitous system, there are some configuration parameters that remain unknown until run-time. For instance, network addresses or available devices and their embedded sensors. This information is very important in order to request virtual and physical sensors for applications. Similarly, the available context information remains unknown until the applications start to discover

nearby devices around them in runtime. The implementation is split into two components:

- **MobileBIT**. This contains mechanisms to use the sensors layers in a device as depicted in Section 4.2. It is also in charge of providing context information to applications. It is a middleware layer running in the mobile devices with enough computational power (smartphones, tablets and smart TVs). This component is useful in design time when the set of sensors and other configuration parameters for a Context Aware application are well known.

- **PervasiveBIT** which refers to the pervasive infrastructure, is in charge of discovering sensors present in ubiquitous environments. It is explained in Section 4.3. It runs on a dedicated server and, as a library, in the mobile devices. Using the library PervasiveBIT client applications report the set of sensors, bluetooth devices and the user for each device in the network. This way PervasiveBIT is informed about type of devices and sensors available on the network. This component complements MobileBIT and makes it possible to fulfil the requirements of the conceptual framework during launchtime .

Both components together allow the use of sensor abstraction and context model described in the conceptual framework.

A general overview of the framework can be found in Figure 4.2. MobileBIT can transform raw data into higher semantic level context information. PervasiveBIT allows to share new context information to be shared across devices in the network. Both components together implement the conceptual framework described in the previous chapter.

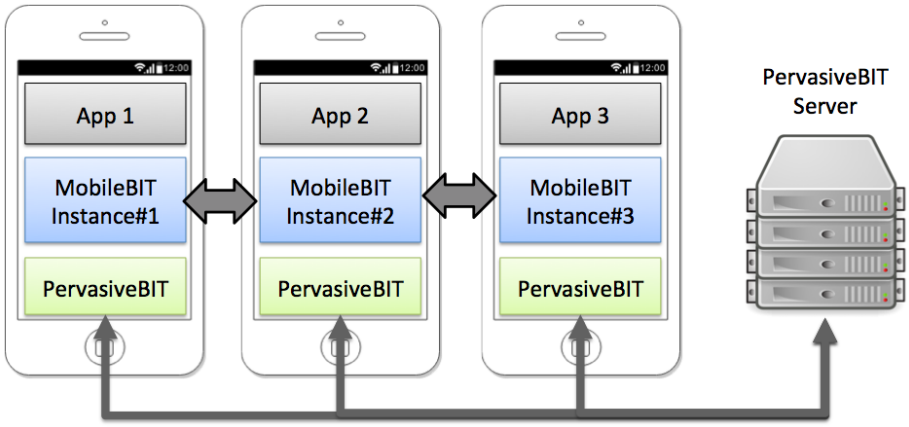


Figure 4.2: Components distribution regarding the devices connected to a network

4.2 MobileBIT

4.2.1 Introduction

MobileBIT is in charge of the abstraction layer of the sensors and the composition of context information within a mobile device (a smartphone or tablet, for instance) in a ubiquitous environment. MobileBIT is implemented as an Android library designed following a modular approach. It is extensible and allows the use of external libraries to add compatibility with new processing methods and sensor devices. In the following paragraphs, the main features and relevant details of MobileBIT are described.

MobileBIT framework [15], is based on a previous work of the Pattern Recognition and Image Analysis group (PIA) of the Instituto Superior Técnico - Univeristy of Lisbon (IST-UL). It was adopted as the sensor abstraction component during a research stay in the PIA group, because it fitted the requirements of the Igerri framework. Afterwards, we adopted, developed and extended MobileBIT to make it suitable for Context-Aware applications.

4.2.2 Sensor-Driven Mobile Applications

MobileBIT hides issues with sensors when programming mobile applications, providing a common interface and mechanisms to access sensor data without directly manipulating them. It is intended to facilitate rapid-prototyping of mobile applications for telemedicine and mobile health domains dealing with the following tasks:

- Real-time data acquisition
- Data processing (e.g. filtering)
- Data recording or storing
- Communication with external servers (e.g. cloud storing/processing)
- Data visualization (e.g. progress charts)

For its extension to the Context-Aware domain, the following task were also included:

- Sensor data abstraction to provide context information
- Real-time generation of context information
- Communication with other devices with similar characteristics (e.g. other smartphones or tablets)

4.2.3 Architecture of MobileBIT

The above mentioned tasks are divided and wrapped in software components called Functional Blocks which can be reused for different applications. At runtime, Functional Blocks are organised with a middleware layer known as Workflow Manager (WFM) to allow the delivery of sensor data from one component to the next. This way, developers can design and build the functionality of the mobile applications without being concerned about programming issues related with sensors. These elements of the MobileBIT architecture were developed for the Android operative system.

In order to specify which sensors are required and the tasks to be performed with their data, a JSON based description language called Data Processing Language (DPL) was defined to describe the behaviour of the functional blocks. Finally, the user interface of the application is developed using Web based technologies such as HTML, JavaScript and CSS.

MobileBIT follows a hybrid application approach [41]. These applications combine a Web layer with a Native layer. With regard to the sensors, MobileBIT hides the data manipulation in the native layer and allows specific operations from the Web layer (e.g. start/stop protocols for acquiring data from sensors). This way, developers rely on previously defined/programmed Functional Blocks to create a web application with the intended functionalities.

There are several reasons to follow the hybrid approach:

- Physical sensor programming issues are explicitly hidden using this approach. Developers focus on the final applications and MobileBIT modules deal with sensors.
- A web layer facilitates the developing of user interfaces to Web designers and developers. Developers can choose which Blocks to use and don't need to mind with Android programming issues. There is a large community that MobileBIT can address using this approach.
- Applications benefits from the Javascript and CSS frameworks obtaining a professional look and feel in their user interfaces.

On the other hand, the performance of hybrid applications comparing to native applications is always a drawback. If it is required, our conceptual framework allows to create Android native applications without the hybrid approach. Sensor data can be accessed from native applications, this option is more suitable for developers with previous experience using the android platform.

A full overview of the architecture is depicted in Figure 4.3.

Summarizing, MobileBIT includes four mechanisms that makes it useful for the development of sensor-driven mobile applications:

- Functional Blocks
- Data Processing Language

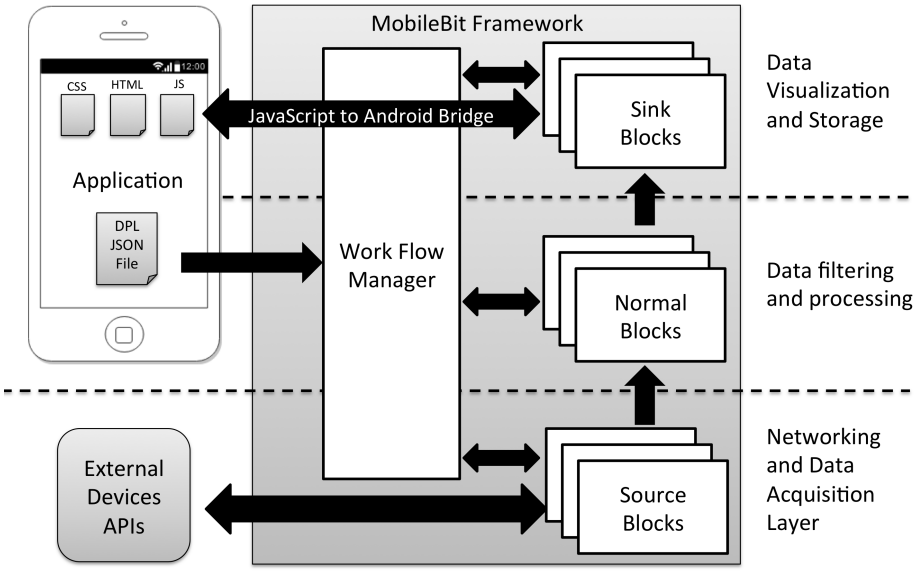


Figure 4.3: Architecture of the MobileBIT Framework for Sensor driven mobile applications

- Workflow Manager
- Web based user interfaces

The framework was extended and modified providing mechanisms for the Context-Aware domain. For these applications, the Web layer can obtain context information using an event driven approach. When new context information is generated it is registered and delivered to the JavaScript layer. Finally, developers decide how to use this information appropriately in Context-Aware applications.

Functional Blocks

Functional blocks are the basic components of MobileBIT. Every block encapsulates common tasks for applications dealing with sensors. The blocks communicate following the observer pattern and share information at runtime with the mobile applications.

Functional Blocks use the concept of "*channel*" to stream the data from one block to the next one. There are two different types of channels regarding the direction that follows the stream of data, input channel and output channel.

A Functional Block may require a specific amount of data in an input channel (sample size) to successfully fulfil a task. The values obtained are delivered to the next block using the output channel. For instance, processing data from an accelerometer with a specific fixed sampling rate of $100Hz$ can require 20 samples in order to obtain a valid processed value. This means that every 200

milliseconds there will be enough data in the Functional Block to process the task. In one second, the block can generate 5 new values (or a data set of 5 datum) in the output. In order to make the Functional Blocks compatible with each other, they can be configured using parameters (for instance, the sample rate). In this way, Functional Blocks can be reused several times.

There is no restriction in the number of input/output channels required for each Functional Block. However, MobileBIT distinguishes three types:

- **Source Blocks** (No input channel and 1 to N output channels) are data acquisition devices (for instance, the Bitalino Board) and they wrap the connection with external devices (e.g. Bluetooth communication protocol). As a consequence, networking issues are hidden to developers. In addition all sensors can be used and activated in the same way, achieving a homogeneous access to the data of the sensors from MobileBIT.
- **Sink Blocks** (1 to N input channels and no output channel) are aimed at recording, communicating and visualizing the data.
  - When it comes to record/storage of the data, these are used to create logs of the outputs of some Blocks in the flash memory of the device.
  - With regard to communication, Sink Blocks are used for remote storage or for feeding with data services in the cloud. For this purpose, they use network protocols such as WebSockets.
  - There are also blocks devoted to data visualization for the final users. These Sink Blocks usually require some code in the web layer to properly adapt the visualization to the mobile application.
  - Finally, Sink Blocks can be used to deliver context information to the Web layer with an event driven approach. In this case the developer has to deal with this information and adapt the behaviour of the mobile application accordingly.
- **Normal Blocks** (1 to N Input and 1 to N output channels) are designed for processing the sensor data streams and combining properties of source and sink blocks. They can be programmed to use an external server to process the data, or use the same device to do this task. When there is enough data in the input channels they are processed and the results are delivered to the next Functional Block. On the one hand, several inputs can be used to perform complex data processing (e.g. Machine Learning algorithms), on the other hand, several outputs can be used to provide different feature extraction at the same time to the next block.

Functional Blocks are the representation of the Sensors as described in the Layered Conceptual framework of Chapter 3. Physical Sensor are implemented as Source Blocks. Virtual Sensors are implemented as Normal Blocks. The layer of abstract sensors is implemented by PervasiveBIT.

Functional Blocks are programmed once and then reused for any mobile applications when required. For this purpose, MobileBIT includes a set of

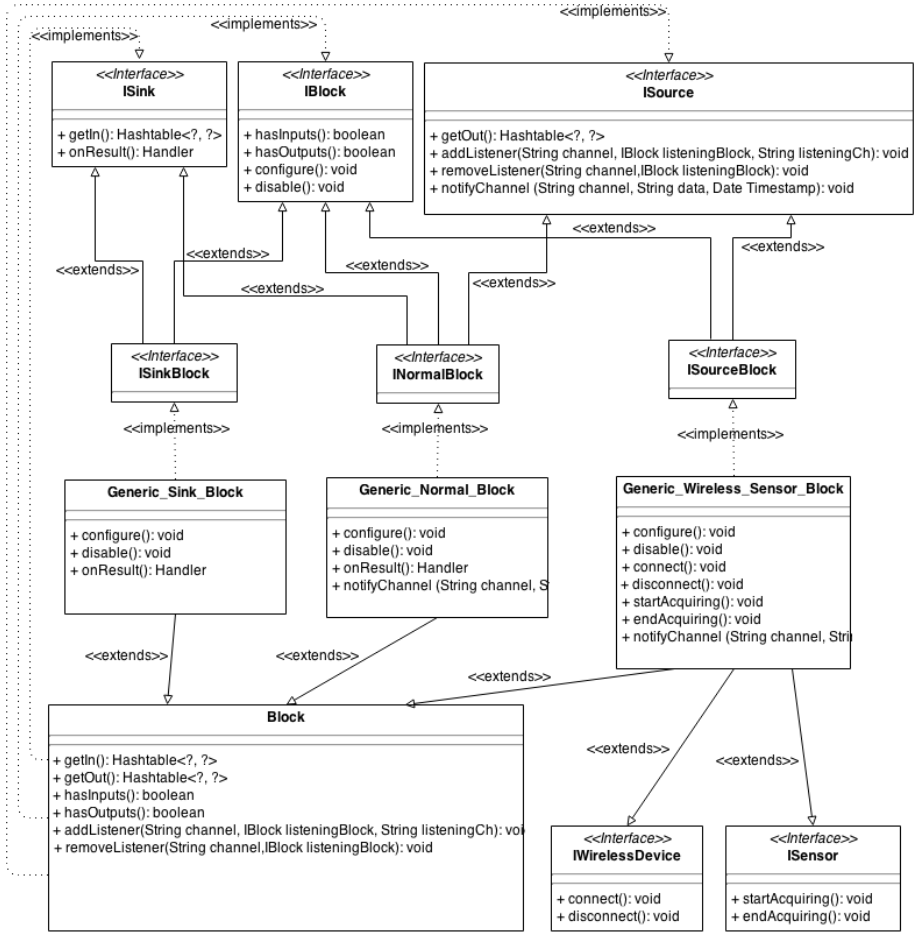


Figure 4.4: Functional Block class and interface hierarchy described in UML

Java interfaces and classes that helps developers to quickly create a Functional Block. The UML design of these classes is depicted in Figure 4.4. The developing of new Functional Blocks is straightforward using these interfaces and the already programmed Java code.

The following example contains a block network designed for an application devoted to monitor patients health using a smartphone (see Figure 4.5). There are four blocks in the system:

1. BITalino (Source Block). This block is connected to a BITalino board that has an ECG sensor.

2. Heart Beat Detector (Normal Block). This block is in charge of processing ECG waves and obtains diverse parameters such as the Heart Rate.

3. Flot (Sink Block). The Flot is in charge of data visualization. It is based in the Flot library provided by Google for JavaScript.

4. Storage (Sink Block). It takes the processed values from the Normal Block and stores them in a Log file on the smartphone.

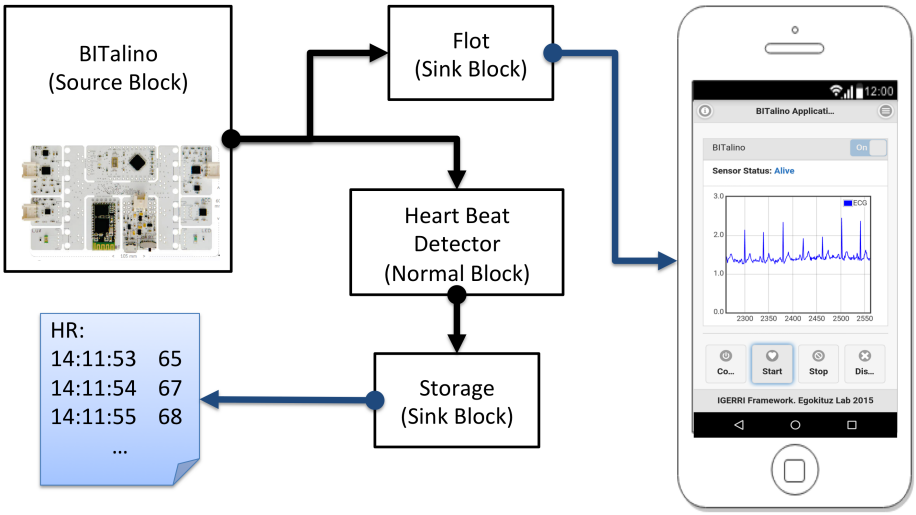


Figure 4.5: Example block application for health monitoring

This application was developed on top of the MobileBIT and did not require further programming concerns than the design of the user interface and the description of blocks. Functional blocks deal with sensors and are ready to be used with other blocks through the input and output channels. This way, the development time is reduced by just coding the expected behaviour of the application. In terms of software components, the Functional Blocks are decoupled from MobileBIT in independent libraries (.jar) that are included in the main application alongside the mobilebit.jar library. In the following subsection, the mechanism to put together the Functional Blocks is described.

Data Processing Language

In order to configure the parameters for each block and to establish connections between input/output channels, a JSON based description language, known as Data Processing Language (DPL), has been defined. With this language, the list of Functional Blocks that take part in an application are described (Listing 4.1).

```
1 {
2   ...
3   "<CLASS:LABEL>": {
4     "config":{"<PARAMETER>";"<"VALUE">",... },
5     "in":{"<CHANNEL>":"<CHANNEL_LABEL>"|"<LABEL>/<CHANNEL>",...}",
6     "out":{"<CHANNEL>":"<CHANNEL_LABEL>"|"<LABEL>/<CHANNEL>",...}",
7   }
8   ...
9 }
```

Listing 4.1: Generic code definition for the Data Processing Language

For every Functional Block four elements have to be defined:

- **Identification labels**. Every Functional Block is identified using the class name of the library which implements that Block. There is also a label to allow calls to the Functional Block from the Web layer.
- **Configuration parameters**. Functional Blocks are configurable for some parameters. For instance, when there is a Source Block for a Sensor, critical values are: the sampling rate, the number of samples required to obtain a frame (window size) and parameters related to scaling and the amplitude of the data. For other blocks, e.g. Sink block for Storage, the name of the file to store the log or whether it uses internal or external memory (flash vs. microSD) are specified.
- **Output channels**. These specify the paths from the sensors to the selected inputs of Functional Blocks. For instance, a Source Block for a Sensor board can have as many output channels as sensors. In the case of Normal Blocks for feature extraction of a signal, they can have different outputs for all the available features. Output channels are labelled with a number and a text. While the number is used only to show the channel data addressed, the text label is an identification for the rest of the blocks to refer to that output channel. Sink Functional Blocks do not have Output channels.
- **Input channels**. They represent the input data required to activate a task in a Normal or Sink Block. They refer to other Functional Blocks Output channels that are connected to the present block. Input channels, like the outputs, have a number and a text. Although the number has a similar meaning, the label refers to other blocks channels.

The behaviour of the sensor data can be described using this language. The complexity of the DPL file expands when the number of sensors and tasks to be performed increases. The three types of Functional blocks are differently designed and are combined together to set up the network. In the following example, Listing 4.2 (shortened in Table 4.1), the DPL file is described. These Functional Blocks were presented in the previous subsection Figure 4.5.

When the DPL File is loaded in the MobileBIT, the file is parsed. If there is no problem, the behaviour of the sensors and communication channels between blocks is established. In the next subsection, the module in charge of the correct creation and maintenance of the block network is described. Additional DPL Files used during the evaluation and testing of the framework can be found in Appendix A.

Workflow Manager

The Workflow Manager (WFM) is the core of the MobileBIT framework. The main goal of the WFM is to automatically handle the subscription between blocks following the Observer Pattern [**?**]. To this end, WFM requires a DPL File to access the libraries of all the Functional Blocks used in the DPL. The

DPL file has to be properly formatted and it has to follow the design rules mentioned in the previous subsection. Otherwise WFM will raise an error warning. Additionally, the WFM allow access to every instance of the system, allowing direct manipulation of the Blocks from the applications user interface. By this mechanism, the users can trigger the acquisition of sensor data on demand or modify some parameters of the blocks.

```
{
  "BitalinoBlock:Sensor":{
    "config":{
          "MACaddress": 11:22:33:44:55:66,
          "Mode":"LIVE",
          "SamplingRate":100",
          "FrameNumber":30"
    },
    "in":{},
    "out":{
      "3":"ECG"
    }
  },
  "HeartRateAlgorithm:HeartBeatDetector":{
    "config":{},
    "in":{
      "1":"ECG"
        },
    "out":{
      "1":"HR"
    }
  },
  "FlotBlock:ProgressionChart":{
    "config":{
      "FileName":"HR_log",
          "Frames":30
    },
    "in":{
      "1":"ECG"
    },
    "out":{}
  },
  "StorageBlock:HeartRateLog":{
    "config":{
      "FileName":"HR_log",
          "Frames":1
    },
    "in":{
      "1":"HR"
    },
    "out":{}
  }
}
```

Listing 4.2: Example JSON DPL for an eHealth application

Table 4.1: Information of the Functional blocks contained in the DPL used for the eHealth application example

Functional Block	Type	Input	From	Output	To
BITalino	Source	-	-	3: ECG	HeartBeatDetector ProgressionChart
HeartBeatDetector	Normal	1: ECG	BITalino	1: HR	HeartRateLog
HeartRateLog	Sink	1: HR	HeartBeatDetector	-	
ProgressionChart	Sink	1: ECG	BITalino	-	

Regarding the network setup from the DPL, as soon as the MobileBIT is created and launched, the DPL file is parsed obtaining a list of tuples. Every tuple contains the following information for each block: identification names, configuration parameters, input channel list, output channel list. Using this information, MobileBIT creates the network of blocks following a two step algorithm (see Algorithm below).

Data:
block is a tuple $\{name, Config, In, Out, Listeners\}$
In is a map of tuples $\{ch\_number, ch\_label\}$
Out is a map of tuples $\{ch\_number, ch\_label\}$
Listeners is a map of tuples $\{ch\_input, \{block\_name, ch\_output\}\}$
NetworkMap is a map of tuples $\{block\_name, block\}$
OutputList is a map of tuples $\{ch\_label, \{block\_name, ch\_num\}\}$
Input: *DPL* format *JSON* String with $name, Config, In, Out$ information for all the blocks in the application
Result: $NetworkMap\{block\_name, block\}$

1 $NetworkMap \leftarrow \emptyset$
2 $OutputList \leftarrow \emptyset$
`/* First, get all the blocks and outputs in the maps */`
3 **foreach** $\{name, Config, In, Out\}$ *in DPL* **do**
4 $block\_instance \leftarrow$ create $block\{name, Config, In, Out, \emptyset\}$
5 set up $Config$ parameters for *block_instance*
6 $NetworkMap \leftarrow NetworMap \cup \{name, block\_instance\}$
`/* Get the output list from the current` *block* `*/`
7 **if** $block\_instance\{-, -, Out\} \wedge Out \neq \emptyset \wedge Out = \bigcup_{i=1}^{n} output_i$ **then**
8 **foreach** $\{ch\_number, ch\_label\}$ *in Out* **do**
9 $OutputList \leftarrow OutputList \cup \{ch\_label, \{name, ch\_number\}\}$
`/* Second, assign the channel listeners to the blocks */`
10 **foreach** $\{block\_name, block\}$ *in NetworkMap* **do**
11 $block\_listeners \leftarrow Listeners$ from *block*
12 **if** $block\{-, In, -\} \wedge In \neq \emptyset \wedge In = \bigcup_{i=1}^{n} input_i$ **then**
13 **foreach** $\{ch\_number, ch\_label\}$ *in In* **do**
14 $output \leftarrow$ get $\{block\_name, ch\_num\}$ for *ch_label* in *OutputList*
15 $block\_listeners \leftarrow block\_listeners \cup \{ch\_number, output\}$

Algorithm 4.1: Algorithm used to create the network of Functional Blocks

This algorithm assumes that all the blocks connections follow the observer Pattern. In the first step, it takes the information about the blocks from the DPL file and fills two data structures, the *NetworkMap* and the *OutputMap*. Then, in the second step, the blocks of the *NetworkMap* are connected using the information of the *OutputMap*, filling the *Listeners* data structure of each block. In this way, when the applications activate a block, a chain of tasks is activated in the proper order: First source blocks, then normal blocks and

finally sink blocks.

After creating the blocks network, sensor data acquisition has to be triggered to begin the data streaming in the blocks. When the Source blocks are activated, the data flows along the rest of Functional Blocks providing the expected result in the Sink Blocks. This activation is enabled using a bridge between the user interface and the blocks in the WFM. As mentioned above, this mechanism allows direct manipulation of the block instances through the WFM and it is further described in the next subsection.

User Interfaces in MobileBIT

MobileBIT is designed to take advantage of hybrid applications approach [41]. Currently, all the mobile platforms (e.g. Android, iOS, etc.) have the possibility to embed a web browser in Native applications, enabling the use of Web based technologies. This component is commonly called WebView and allows the visualization of local or remote Websites and also the execution of JavaScript code. The use of Webview allows Web developers to approach the sensor enhanced applications without having to learn Android programming language. In addition Webview benefits from the latest JavaScript frameworks.

MobileBIT is running in the Android Native layer, and it hides the sensor programming issues. The rest of the application is part of the Web layer. Due to this fact, the management of the sensors is encapsulated in the MobileBIT library and separated from the application itself. This simplifies operations regarding the sensors and avoids common mistakes made by beginners in native applications development. The user interface and the behaviour of the application is coded in HTML, JS and CSS. Anyway, both layers must be connected to communicate with each other. For this purpose a bridge known as JavaScript-Interface (JSI) has been used. The JSI works in both ways, from Native layer to the Web layer and vice versa:

- **From Native to Web**. When Sink Blocks send data to the user interface it uses the JavaScript function listed in Figure 4.6 to evaluate the received data. If it is part of one of the JavaScript functions used in the application, this code runs. For instance, this method is used to show the data collected by a sensor in a progression chart, or to provide new context-information to be handled by the web application.

```
function onmessage(e){
  eval(e.data);
}
```

Figure 4.6: Code for receiving information from the MobileBIT in the JavaScript layer

- **From Web to Native**. When the application calls a method of the functional blocks it uses a similar code to the one listed in Figure 4.7. In the case of this Figure, calling *send('Sensor.startAcquiring()');* a function starts acquiring data from a Source Block labelled as Sensor in the DPL of the application. The commands are received by the WFM and parsed to call the specified method in the Functional Block. This method's calls are asynchronous. Return values and error information are obtained with the callbacks listed in Figure 4.8. Asynchronous behaviour is required to grant a fluent interaction with the user interface of the applications. The sequence to send a command and receive a response is depicted in Figure 4.9.

```
window.wv.send('Sensor.startAcquiring()');
```

Figure 4.7: Code for calling a method in a Functional Block named Sensor

```
function done(cmd, val);
function error(cmd, val);
```

Figure 4.8: Callbacks to get the result for the Functional Blocks functions

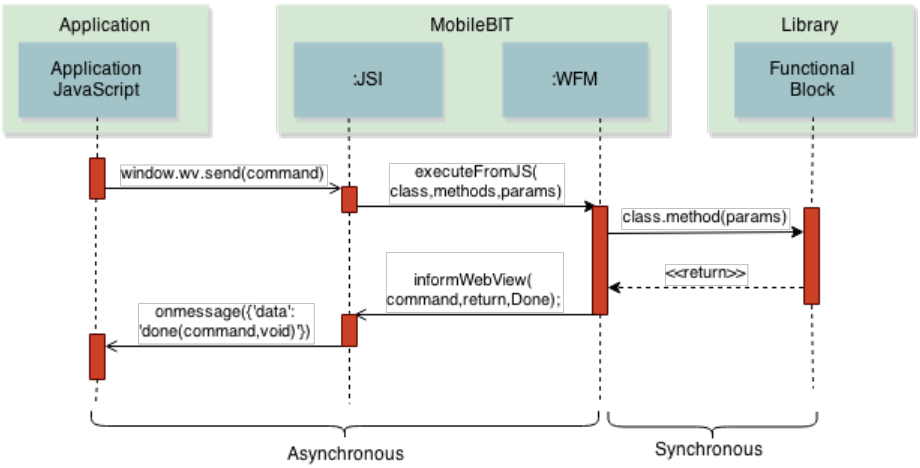
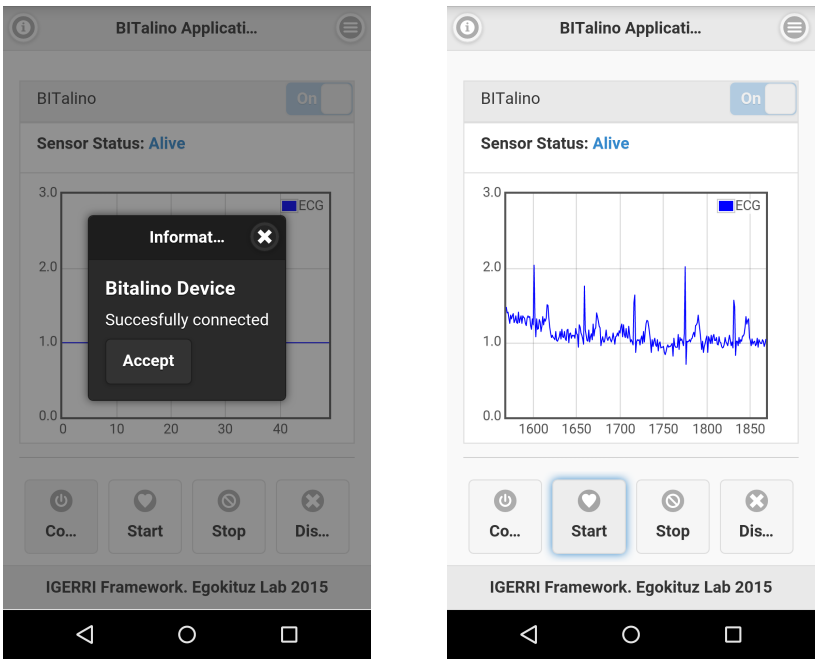


Figure 4.9: Sequence diagram for the call to functions

JSI is designed to be compatible with the semantics of WebSockets [29]. At developing time, the user interface can be instrumented and accessed remotely from the a desktop computer, making it easier to debug from a conventional Web Browser rather than the embedded browser.

Some Sink Blocks structure and organize data from sensors for plotting li-

braries. Although it would be expected to have the visualization as an exclusive task for applications developers, this is very useful for rapid prototyping sensor-driven applications that need real time progress charts. Sink Blocks related to the visualization of the data in the user interface are designed to use the functionalities of JavaScript frameworks like JQuery Mobile[1] and other libraries to improve the presentation of the data to the end user. For instance, figure 4.10 shows two screenshots of the example application for heart rate monitoring. A notification message based on JQuery Mobile is displayed in *Screenshot a*. In a similar way *Screenshot b* shows the Flot library[2] for plotting charts.



Screenshot a **Screenshot b**

Figure 4.10: User interface of the example application

4.2.4 Context-Aware Support for MobileBIT

Context Aware applications use the context information for many purposes. For instance, to adapt the graphical user interface or to consider new interaction modalities. MobileBIT has been extended so as to be useful for the creation of Context-Aware applications.

[1] `http://jquerymobile.com/`
[2] `http://www.flotcharts.org/`

Rationale for the Adoption of MobileBIT

As was explained in Chapter 3, the Igerri framework considers a series of layers in order to deal with sensors. These concepts are represented in the MobileBIT framework by means of the Functional blocks and the DPL. MobileBIT Functional Blocks are the implementation of the different types of sensors described in Igerri:

- Igerri's Physical Sensors are built by means the Source Blocks. The main difference with the conceptual framework is that the Source Block represent all the sensors grouped by device instead of presenting them one by one. Nevertheless, this difference is not relevant for the final application, since the DPL allows only the sensors required for the application to be described.

- Igerri's Virtual Sensors are built using Normal Blocks. By means of processing techniques other physical and virtual sensors can be transformed into a new Virtual sensor.

- Igerri's Abstract Sensors are not explicitly built as a different Functional Block but as a *simplification* of the above mentioned Source and Normal Blocks.

- Igerri's Context Services are built using Sink Blocks. Sink Blocks communicate with the application and deliver context-information to the application.

It is possible to have the same context aware application running for different DPL files with MobileBIT. This feature is necessary to obtain Abstract sensors because it allows the interchange of a DPL file by another one maintaining the context aware application up to date with equivalent context information.

Besides the encapsulation of sensor abstraction in Functional Blocks, the design of MobileBIT allows the decoupling of the user interface. The separation of the sensor abstraction concerns from the Context-Aware application development is implicit in the design. MobileBIT is in charge of sensor issues and the Web layer contains the methods necessaries to create Context-Aware applications. Additionally the use of a Web layer makes MobileBIT compliant with the approach followed in Egoki meeting the requirements stated in Chapter 1 for this thesis.

Nonetheless a mechanism to have access to the context information from the Web layer has to be specified. But a new type of block that can handle the context in a similar way as the Context Services specify can be easily added. The following lines explains how to deliver context to the MobileBIT application.

Context Delivery Mechanism

The delivery of context information to the Web layer is the first step in order to design and develop new Context-Aware applications. We modelled the context-information as the tuple:

$$\texttt{context-information} \equiv \{\ \texttt{Entity, Context, Value, TimeStamp}\ \}$$

To achieve this MobileBIT uses a special Sink Block called Context Block. Context Blocks are the implementation of the Context Services in the Conceptual Framework. This Block receives from an input channel the information to be delivered to the Web Layer as context. It has two main features:

- Specific configuration parameters to deal with context information from other Normal blocks.
- A callback function to be used with the JSI that has to be implemented by the Context-Aware application.

Functional blocks can produce context information that is not valid for a specific Context-Aware application. This happens when a Normal Block is used for several purposes. For instance, EMG electrodes can be attached to many muscles but the Normal Block to process the EMG signals have to be reconfigured each time to indicate which muscle has been measured or which movement has taken place. It also happens with general purpose sensors such as accelerometers. In order to make the Blocks more reusable, this block renames and formats the context information for each application. The following configuration parameters were used for the Context Block:

- **Entity** refers to the person, object or place which characterizes the context information.
- **Context** refers to the action or event taking place for the above mentioned entity.
- **ValuesMap** are the specific values that will be delivered to the Web layer. They are mapped taking into account the values that are obtained in the input channel. If there is not a values-map specified for the block, the Context Block just conveys to the Web layer the input values enriched with the Entity - Context values.
- **Notification** refers to the way the Context Block and the Web layer communicate:
  - **All**. Every time new data arrives to the Context Block using the input channel it is delivered to the Web layer.
  - **Updates**. The Web layer is notified only when there is a change in the context-information values. Sometimes, the previous notification mode (All) may deliver exactly the same context information values. To avoid this unnecessary repetition, this mode will only

notify the Web layer in the case of changes in the context information.

- **None**. This mode is useful in scenarios where the Context Blocks does not send values to the Web Layer. Applications require on-demand access to the context-information. The block has a method that has to be called from the WebLayer to get the last value for the context information.

These parameters are an extension for the DPL language and follow JSON syntax. They provide a configurable layer for the context-information in order to be appropriate for the Web layer.

Example 1: Environmental Light Context Application

The DPL of a context-aware application is depicted in Listing 4.3.

```
{
  "Smartphone:Sensor":{
    "config":{...},
    "in":{},
    "out":{
      "10":"Light"
    }
  },
  "ThresholdCompare:normal":{
    "config":{
      "compare":"greater_than",
      "threshold":"1000"
    },
    "in":{
      "1":"Light"
    },
    "out":{
      "1":"Brightness"
    }
  },
  "ContextBlackboard:Ctx-information":{
    "config":{
      "Entity":"Room",
      "Context":"Environmental_light",
      "ValuesMap":{
        "Light":true,
        "Dark":false
      },
      "Notification":"Updates"
    },
    "in":{
      "1":"Brightness"
    },
    "out":{}
   }
}
```

Listing 4.3: Example JSON DPL using Context Block

In this application, a smartphone light sensor is used to measure the level of light in a room. The Source block sends the data from the light sensor to a Normal block that simply checks if the value of the sensor is above a certain threshold. The output of that block is a boolean value, true when the light

value is over the threshold and false in other case. Thus, in the Context Block, the input values are mapped to *Light* when the value of the sensor is over the threshold and *Dark* if the value is under the threshold. Finally, the Context Aware application only needs to adapt its behaviour to the context information. This is a straightforward use of the Context block to better explain how it works. However, the real importance of the context renaming comes when there is need to reuse the same Functional Block for different purposes.

Example 2: Arm Gestures Context Application

The activity in the right arm biceps is measured in the application described in Listing 4.4. For that purpose a EMG sensor from the BITalino board is used. EMG data are filtered and processed to obtain values periodically. These values have an implicit semantic regarding the muscle state in the output of the Normal Block. The application returns the values *contracted* or *relaxed* depending on the input. Application developers aimed to use this muscle contraction detector block to associate it to two different muscles, the biceps of the right arm and the thenar eminence of the right hand. Therefore they had to relabel the context information of the Normal Block and obtain different values in the Web layer.

All the context-information is delivered to the web application using the JSI bridge. For this purpose the application developer has to include and complete the following function in the JavaScript code.

```
function perceived(context , value){
  // Developers code for context aware application ...
}
```

The first parameter (context) contains a complete description of the information required to understand 'value' and the second parameter is the context-information value itself.

4.2.5 Guidelines to Improve the Performance

The performance of the MobileBIT applications is an important concern and depending on the sensor sampling rate when dealing with data intensive applications the streams of data flowing through the chain of blocks can suffer performance drops and bottlenecks. Usually, sampling rates over 100Hz are problematic in modest or inexpensive smartphones if the application doesn't address the performance issues properly. The main evidence for these problems is that the user interface lags and in the most extreme cases, is blocked, which has an undesirable impact on the usability of the applications.

```
{
  "BitalinoBlock:Sensor":{
    "config":{...},
    "in":{},
    "out":{
      "1":"EMG_1",
      "3":"EMG_2"
    }
  },
  "MuscleContraction:Muscle_biceps":{
    "config":{...},
     "in":{
      "1":"EMG_1"
     },
    "out":{
      "1":"Arm_folded"
    }
  },
  "MuscleContraction:Thenar_eminence":{
    "config":{...},
     "in":{
      "1":"EMG_2"
     },
    "out":{
      "1":"Hand_closed"
    }
  },
  "ContextBlackboard:Ctx_arm":{
    "config":{
      "Entity":"User",
      "Context":"Right:Arm",
      "ValuesMap":{
        "Folded":"contracted",
        "Unfolded":"relaxed"
       },
      "Notification":"Updates"
    },
    "in":{
      "1":"Arm_folded"
    },
    "out":{}
   },
   "ContextBlackboard:Ctx_hand":{
     "config":{
       "Entity":"User",
       "Context":"Right:Hand",
       "ValuesMap":{
         "Closed":"contracted",
         "Opened":"relaxed"
       },
       "Notification":"Updates"
     },
     "in":{
       "1":"Hand_closed"
     },
     "out":{}
    }
}
```

Listing 4.4: Example DPL file for Arm Context

These issues could be expected to disappear with the increase of computational power in mobile devices. However, this cannot be taken for granted because the software (or operative system) requirements will increased in a similar way. On the other hand, the sensor task processing requires higher sampling rates than 100Hz. For instance Lourenço et al. 2011 [52] propose a

method to analyse the ECG signal acquired at 1000Hz for biometry applications.

These concerns take special importance for Real-Time applications, when all the data must be processed and transformed in short periods of time. Some guidelines have to be followed when using the MobileBIT framework to address these problems.

- **Tweak window size according to the Sampling Rate**. The sampling rate is a critical parameter for a sensor driven application. The amount of sensor data processed is stored in intermediate data structures and delivered over the rest of the application. Functional blocks separate concerns and send the data periodically to the next block in the chain, to facilitate its manipulation. However, the window size has to be considered to optimize this. For instance, with a 100Hz Sampling Rate, a window size of 100 samples for a visualization would require waiting 1 second from update to update. In some cases, this is not a problem (for instance updates in the HeartRate) but can create a feeling of a lack of fluency (for instance, when showing updates of an Electrocardiograph in a progress chart). On the contrary, smaller data chunks, for instance 1 sample window size can overload the Functional block input/output channels with an excess of small size messages. Functional blocks have to allow mechanisms to set up these parameters in order to find the balance between those two different scenarios.

- **Discard extra information as soon as possible**. Sometimes the Sampling Rate is higher than required but it cannot be modified. For instance, the sampling rate in BITalino can be between 1, 10, 100 or 1000 Hz. To calculate the Heart Rate from the ECG, a 50Hz sampling rate is enough with some algorithms (see the difference of sampling rate for the ECG in Figure 4.11).

  If the redundant information is not used elsewhere, the functional block can avoid delivering it to other blocks. This reduction can be done in this block or in the following ones, but the sooner the better. Anyway, if different blocks are fed with the information of a source block, unexpected behaviour can appear if the sampling rate is not appropriate for all the blocks. Developers must be careful when modifying the sampling rate. There are different techniques to carry out this reduction like decimation or downsampling, that can be implemented as Normal Blocks.

- **Use threads to manage input/output channels**. MobileBIT uses the concept of channel to transfer information between blocks. When a block subscribes the output channels it must manage the amount of data incoming from the channels properly. This mechanism is prone to bottlenecks, to avoid them, it must be managed using threading. Usually the reception of the data using the Input channel is implemented in one thread and the delivery of the data is implemented in other separate thread. In a similar way, the use of threads is crucial when dealing with external sensors to avoid blocking the application.

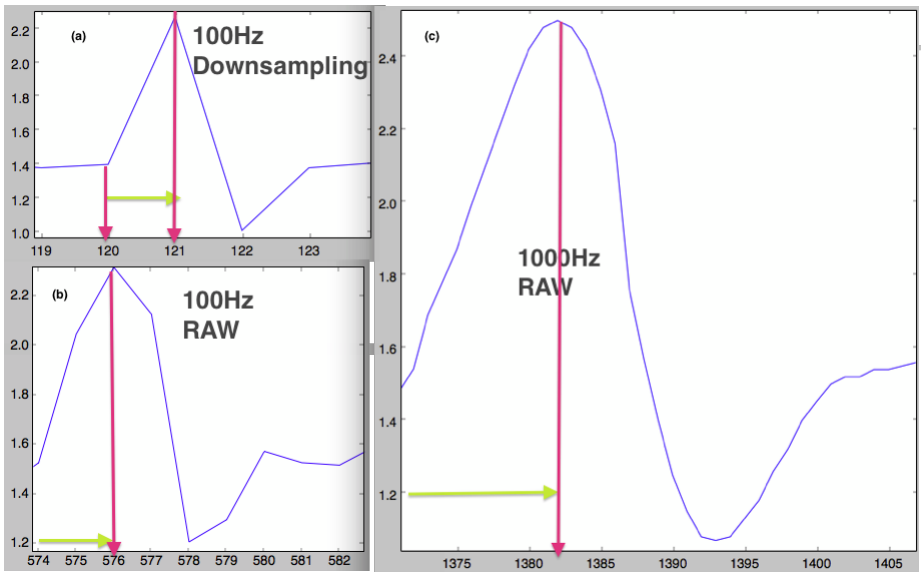


Figure 4.11: In these pictures, the ECG signal is acquired using two different sampling rates. In the left side, the rate is 100Hz (the top one (a) is downsampled to 50Hz and the bottom one (b) is the original). On the right side (c) the sampling rate is with 1000Hz.

- **Be careful with the performed tasks**. Some tasks require several iterations and loops and a considerable amount of time to be performed, in addition this can have a negative effect in the battery charge and processor performance. Sometimes, it is better to send the values to an external server than to try to process all the data in a Normal Block. A careful evaluation must be made concerning which option is more adequate for processing heavy tasks.

- **Avoid overloading the user interface**. When the Sink blocks are connected to the user interface, a mechanism to deliver the information from the native layer to the web layer is used periodically to update the user interface using the event driven approach. There can be problems if the number of JavaScript events per seconds are too high. To reduce the likelihood of this problem, the data to be delivered to the user interface should be accumulated in the same event.

If poor performance is detected in an application that implements MobileBIT, these recommendations should be carefully followed.

4.3 PervasiveBIT

4.3.1 Introduction

So far the implementation has dealt with device internal issues in development time. Nevertheless, Context-Aware applications often requires to interoperate between several devices. For this purpose, the PervasiveBIT component was designed.

PervasiveBIT works as a component to ease the connection between devices and external sensors in run time. Mobile Context-Aware applications can work in an opportunistic way: depending on the number of devices and sensors in the network some applications will work or not. Additionally, information that is not available in development time is required. For instance, the MAC address for a Bluetooth sensor or the IP address of a device. Therefore, this component implements the transformation from the virtual sensor layer to the physical layer explained in the Conceptual Framework.

4.3.2 SensorHub: Automatic Discovery of Sensors

PervasiveBIT includes a module called SensorHub that was developed to gather information about the different types of sensors available in a device. This module works for both internal sensors and externally connected sensors (e.g. via Bluetooth). This mechanism is divided in two parts:

- A library for each device that stores all the information and organizes it.
- A server application in the local network that receives the information about all the devices using PervasiveBIT.

In this way, it is possible to list the devices and sensors available, as was mentioned in Chapter 3. With this information is also possible to know which context-information will be available on every mobile device and to know if an application will work with the information available.

These features are necessary to discover all the sensors available to the Context-Aware application. This information is used to create the repository of elements of the Physical Layer of Physical Sensors (PLPS) as defined in the Conceptual Framework.

4.3.3 SENSONTO: A Knowledge Base for Context Perception

A knowledge base was designed in order to store the parameters to enable the sensor abstraction and the creation of virtual sensors described in Igerri. This knowledge base contains concepts that are relevant to discover the available context-information. The following elements required for ubiquitous computing are conceptualized:

- **Device** contains a set of sensors that have network interfaces to be connected to a network and also provide information of the **user** and its **location**. The Devices run the context-aware **applications**.
- **Sensors** include the three categories considered for the categorization of sensors proposed by Egoki: Physical, Virtual and Abstract sensors.
- **Network interfaces** to communicate the data with other **devices** using **Networks**.
- **Users** are entities to characterize the **context-information**. He/she is also interacting with the **application** and the **device**. The **location** of the user is also a relevant concept.
- **Context-Information**, is the relevant information to be perceived in the Context-Aware **application**. It contains all the lexicon for the Context.

To enable the perception of context-information these concepts must be part of the system. The relation of the concepts is depicted in Figure 4.12.

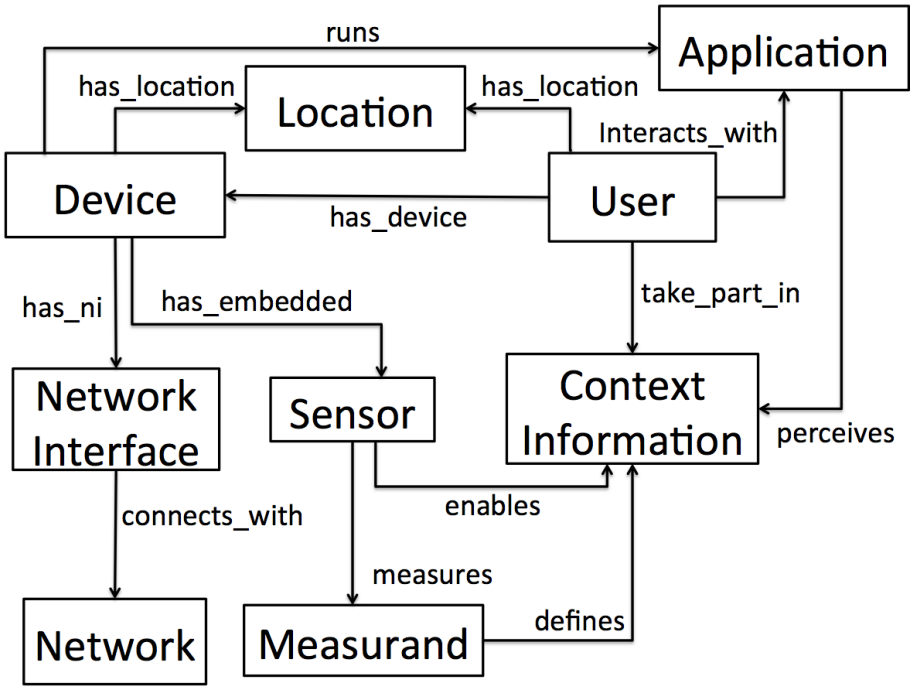


Figure 4.12: Conceptualization of elements in a Ubiquitous System

These elements are implemented in an Ontology called SENSONTO, which is populated with the information gathered by the SensorHUB. It also contains the lexicon to be used by the applications developers. With the additional information about the modules collected by MobileBIT, it is possible to know which Physical Sensors can be used to create new Virtual Sensors.

4.3.4 DPL Generation to Instantiate the Conceptual Framework

PervasiveBIT has two modules with well defined functionalities. Firstly, SensorHub to gather information about the current status of the ubiquitous system (devices, sensors, networks). Secondly, SENSONTO to discover parameters to obtain the context services available for the applications. This process is mainly done by a Server, through the transformations processes detailed in the following paragraphs.

Bottom-up Approach to Igerri Translation Transformations

PervasiveBIT obtains a list of Sensors and Context Services for every abstraction layer in the Conceptual Model (See Figure 4.13).

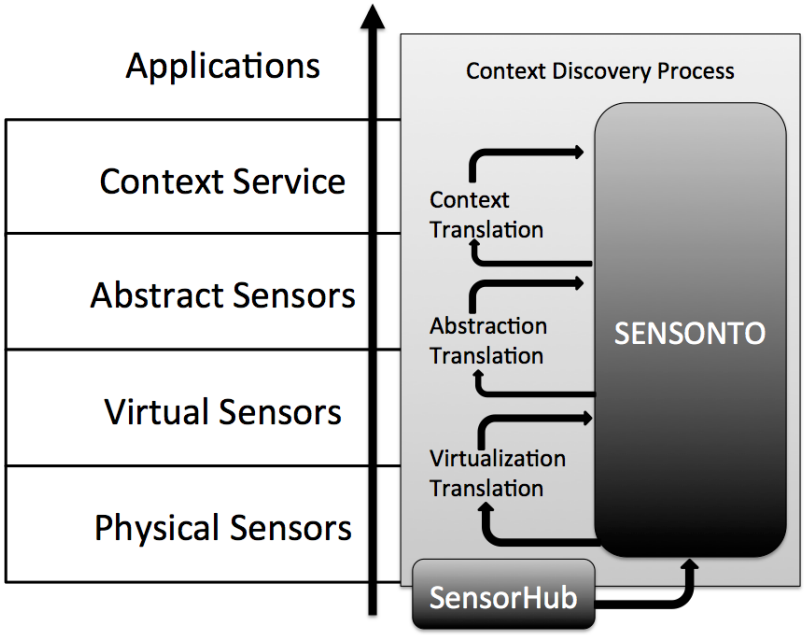


Figure 4.13: A bottom up perspective of the implementation comparing it to the conceptual framework

The lists are filled out following a bottom-up path in a four-step process:

1. The SensorHUB gathers information about all the devices and sensors in the networks. This information is shared with the PervasiveBIT server. It is used to build a list of physical sensors.
2. For every physical sensor in PervasiveBIT, a virtual sensor is created. During this process new virtual sensors from combinations of input from simpler virtual sensors are also created. These Combined Virtual Sensors are discovered using the information available in SENSONTO. It maps with the Virtualization Translation of the conceptual framework.

3. The abstract sensors matched to each virtual sensors are listed. This step requires the use of SENSONTO to discover how virtual and abstract sensors are matched. It maps with the Abstraction Translation of the conceptual framework.

4. The context service is generated based on the list of available Abstract Sensors, again with the help of SENSONTO. The context information will enable the use of words from the Lexicon. It maps with the Context Translation of the conceptual framework.

When the process is completed all the information about the sensors is available. Thus when a Context Aware application requires it, the DPL generator using this information creates the correct Workflow between blocks.

Top-down Approach to Igerri Request Transformations

The main function of PervasiveBIT is the generation of valid DPL code for applications following the Igerri Conceptual Framework layered model. In order to do this, all the elements available in the model are identified starting with the Physical Layer and ending with the Context Service layer. This is a bottom up approach to discover all the perceivable context in the system. Then, a DPL file is created by requesting information to all the layers created in the previous step. The creation of the DPL follows a top-down approach to check which elements are available.

When the developer knows which context information is required for the application he or she calls the PervasiveBIT to obtain a valid DPL for the devices available in the system. This process is opposite to the previous one. It is also composed of four steps:

1. Developers indicate which context names from the Lexicon are used. A pseudo DPL File is generated using this information with the Context Blocks necessary to obtain the context data.

2. After that, the Context in the pseudo DPL is transformed into a set of Abstract Sensors with the help of the SENSONTO. In terms of MobileBIT, the DPL adds a set of Normal and Source Blocks coordinated to work together with the rest of information. It maps with the Contextualize Request of the conceptual framework.

3. In the next step, more information about the Functional Blocks is added to the pseudo DPL File. New blocks are added if required and specific parameters are included. It maps with the Precision Request of the conceptual framework.

4. Finally, the information provided by the SensorHUB is added to complete a valid DPL File. It maps with the Actualize Request of the conceptual framework.

When this process is completed, the DPL is sent back to the Device and MobileBIT is launched.

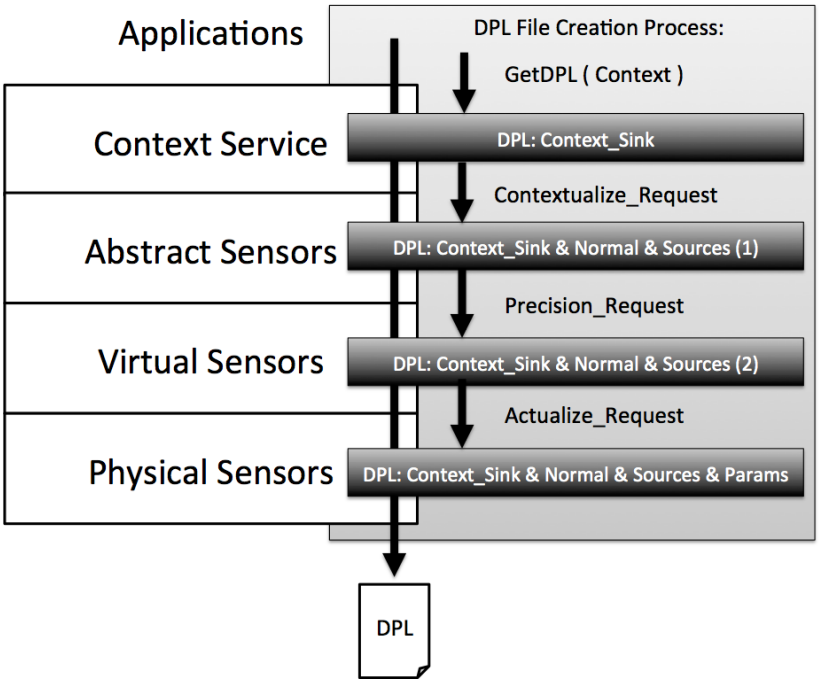


Figure 4.14: Top down perspective of the implementation fulfilling the conceptual framework

4.4 Conclusion

This chapter presents an implementation of the Igerri conceptual framework that relies in two components: MobileBIT and PervasiveBIT.

MobileBIT is a general purpose framework for the development of sensor-driven applications [15]. It was adopted during a research stay in the PIA group in Lisbon. The rationale for this adoption is based on the suitability of MobileBIT to implement the different virtual layers of the conceptual framework and the possibility to use it in design time for the development of new Virtual sensors and Context-Aware applications.

When MobileBIT is combined with PervasiveBIT, the Conceptual Framework Described in the chapter 3 is fully instantiated.

Regarding the challenges enumerated in the ubicomp paper [33], MobileBIT contributes to mitigate the effect of these issues in their applications:

- **Sensor heterogeneity**. With MobileBIT, all the sensors are used in the same way. If they return the same data type, common Functional Blocks can be shared to process sensors.
- **Platform heterogeneity**. Although the MobileBIT is implemented for the Android platform, the use of the Web layer to separate the application

from the sensor abstraction engine makes it possible to run the same context aware applications on new platforms such as iOS or Windows Phone if they have a version of MobileBIT implemented.

- **Encapsulation for complex data processing**. Normal Blocks provide a quick way to implement complex data processing. Some generic filters can be applied by means other than Normal Blocks before and after the processing, thereby easing the programming task. Heavy processing tasks can use an external server to obtain faster results without affecting the performance of the mobile device.
- **Bad performance**. A number of design tips to improve the performance of the Functional Blocks have been described. If the blocks are well programmed, the developer does not need to know inner details of the Android system to improve the performance of their applications.

Regarding the original motivation for this thesis, the lack of an adaptability mechanism in the Egoki system (as stated in Chaper 1), applications based on this implementation have the possibility of improving a Egoki-based Ubiquitous System. Due to the possibility of loading web applications in the Web layer and to updating them using the sensor abstraction layer described in this chapter, minor updates should be required to improve the accessibility and interactions of Egoki user interfaces. In this way, this approach addresses the first approach to this thesis, described in the MOBACC workshop [32].

The design of PervasiveBIT is similar to Egoki: it relies on a Knowledge base and uses middleware for automatic discovery in the network. Although working for different purposes, both (Egoki and Igerri) can share the same infrastructure (a server connected in the network), and also share information between both Knowledge bases EGONTO and SENSONTO.

Chapter 5

Evaluation

In this chapter, an evaluation of the Igerri framework is carried out using representative Context-Aware applications. These applications are based on the concept of Animatronic Biofeedback and examine the movements of the arm to move a mobile robotic platform. The evaluation was carried out with a total of 29 participants obtaining good results in terms of the usability of the applications.

5.1 Introduction

Usability testing comprises a set of practices with which to test the user interfaces of prototypes with representative users attempting representative tasks [51]. With usability testing we can check an applications' functionality, asses the user's experience and identify specific problems in applications [26]. In Chapter 1, we enumerated a series of questions and the hypothesis concerning the usability for Context-Aware applications developed following the Igerri approach. The results of the usability testing are essential to obtain an answer to these research questions:

> **Research Question 1**. *Can Igerri produce functional and usable applications with the mobile Context-Aware framework?*
>
> **Research Question 2**. *Are the users able to control sensor enhanced applications following the Igerri approach?*
>
> **Research Question 3**. *Do the users perceive sensor enhanced Igerri applications as being appealing, engaging and/or of added value?*

On the whole, these questions lead to the following hypothesis: *The abstraction and virtualization of sensors as presented in Igerri are valid techniques with which to develop usable Context-Aware applications.*

In the Context-Aware computing field, from a Human Computer Interaction perspective, researchers use prototypes and case studies to evaluate and

test frameworks and systems. Usually a part of the whole system is tested, rather than the whole system [69]. The evaluation carried out for Igerri is focused on the usability of a representative application. We wanted to obtain evidences whether the above mentioned hypothesis is valid or not from the results of the evaluation (See Figure 5.1).

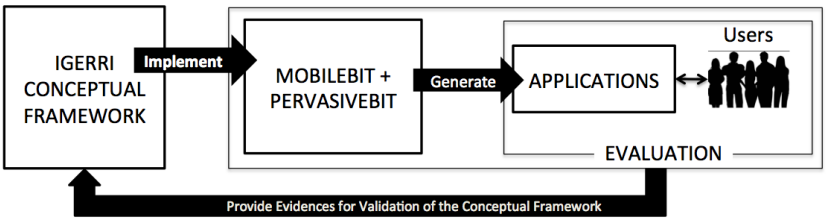


Figure 5.1: Evaluation approach followed for Igerri

The applications proposed in this chapter addresses two functionalities:

- **The generation of context information from a developer perspective.** The tested applications use information relevant to the developers obtained by means of sensors. This information helps to characterise the situation of an entity, in this case the arm of the user. Thus this information can be considered context following Dey's definition [24] mentioned in Chapter 2.
- **The creation of virtual sensors.** Physical sensors are involved in the creation of high-level context information. In this case, the arm context is processed using a Virtual sensor defined in 5.3.

Besides these two functionalities the rest of the framework is also indirectly involved in the evaluation.

To obtain evidence on this matter, representative applications developed with MobileBIT, based on the Igerri framework, were tested using different usability techniques.

5.1.1 Description of the Experimental Evaluation

For the experimental evaluation, quantitative and qualitative assessment were considered. For quantitative evaluation, it was necessary to design and create representative tasks to obtain objective and measurable values. These tasks were designed to combine all the functionalities available in the application. Two types of parameters were studied: First, the task performance, represented by the number and type of errors (if any). To this end, the ability of the participant to finish the task or not is measured. The time performance is also measured, providing a benchmark with which to compare participants and assign them with a reference value. This result can be used to determine if a user had trouble or was unable to use the application properly.

With regard to qualitative assessment, the user perception of the applications is measured. This is a subjective value that can provide indications about the suitability of the interaction with the applications.For this purpose from among the numerous types of Usability tests, the System Usability Scale (SUS) [10, 13] was chosen. SUS is a predefined Likert Scale with ten items with an appropriate design to detect if the participants understood the questions. It was designed as a rapid usability assessment tool and the results obtained can be interpreted in Figure 5.2 (Figure from Bangor et al. 2009 [10]). Values over 68 show good application usability. In addition to the SUS, ad-hoc questionnaires and semi-structured interviews were carried out to obtain more information about the perceived usability of the applications.

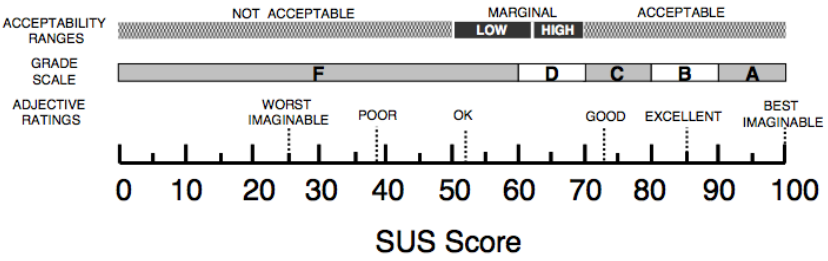


Figure 5.2: System Usability Scale evaluation criteria from Bangor et al. 2009

Different environments to test the applications were considered for the experimental evaluation. Laboratory conditions offer high internal validity of experiments but a lack of evidence regarding how the applications would perform under real conditions. Because of this we also tested the application in the field, where users can be affected by other external and non quantifiable factors. Good results in both settings improves the validity of the applications being assessed.

In the rest of the chapter, the evaluation of two Context-Aware applications with 29 participants is described.

5.2 Tested Applications

Two representative Context-Aware applications were developed: the *ToBITas Case Study* [35] and the *Rehabilitation Exercise System* (RESapp) [38]. The former is a smartphone application designed to control a mobile robot in real time with movements of the right arm. The latter was designed to exercise the right arm in rehabilitation therapy tested in two scenarios: under laboratory conditions and in a retirement day center. For this last application, there are two different possibilities, with and without the help of the robot. The number of participants for each application can be seen in Figure 5.3.

The two applications tested in this chapter are based on the same set of ideas:

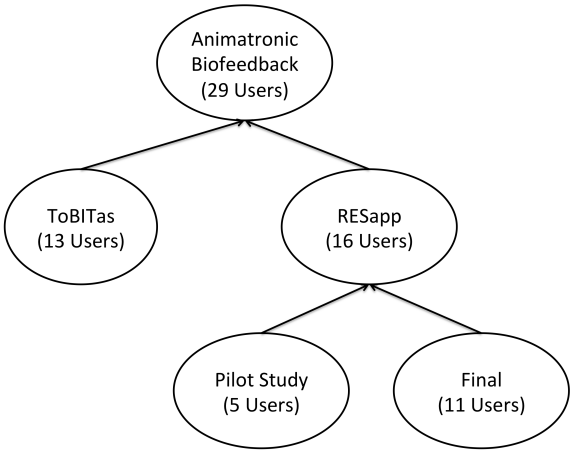


Figure 5.3: Summary of users for each application

- Firstly, the use of an Animatronic Biofeedback approach (informally defined as the use of pre-programmed actions in a robot that are triggered in response to certain changes detected in the users biomechanical or electrophysiological signals).
- Secondly, the use of the right arm monitorization using a BITalino sensor platform as an input interface [73].
- Finally, performing remote control of a mobile robotic platform known as Bot'n Roll [62].

A key element in biofeedback is the ability to measure how a given movement or exercise is being performed. The most common exercises involve either the musculoskeletal system, biomechanical activities, or a combination of both. We used the Electromyography (EMG) and Accelerometry (ACC) sensors to measure movements. EMG provides a direct measurement of the recruitment of one or several groups of muscles, while ACC can be used for biomechanical assessment (e.g. range of motion, limb tilt, etc.).

Although these sensors can be used for a wide range of movements, in the proposed applications, the perceived context information is focused on the contraction of muscles and tilt movements of the right arm. The user must wear a band with the BITalino attached to the wrist to acquire those signals. These experiments focus on muscle contraction of the Biceps and the Thenar eminence and on the wrist tilt movement. The arrangement of the electrodes are shown in Figure 5.4. From left to right: EMG 1 (biceps), EMG 2 (thenar eminence) and the board placement on the wrist with incorporated ACC sensor.

The application is notified regarding every new context information acquired and as a result it sends a command to move the robot as a biofeedback response. In this way the Animatronic Biofeedback is achieved. The set of programmed movements for the robot are the same for both applications: forward

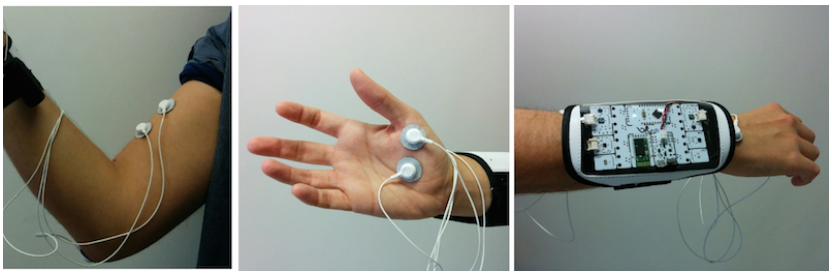


Figure 5.4: Electrodes and sensor placement for the right arm

or backward motion; turn left or right; and closing and opening the claw.

Each application will activate this movement in a different way depending on the underlying semantics of the application: the completion of a task in real time for ToBITas and the carrying out of rehabilitation exercises for the RESapp. The overall setup can be seen in Figure 5.5.

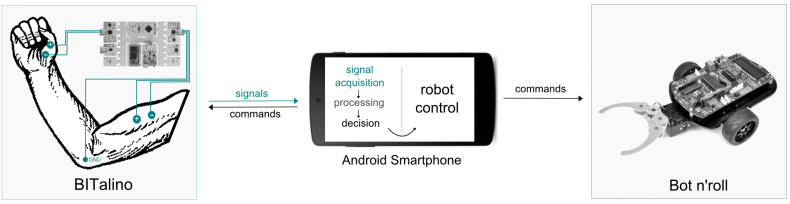


Figure 5.5: Main elements of the proposed Context-Aware biofeedback applications

5.3 Virtual Sensors

Two new functional blocks were developed to connect the right arm movements with the BITalino with the Bot'n Roll movements. BITalino combines EMG and ACC sensors with a wireless communication module that uses Bluetooth technology, providing both biosignals acquisition and connectivity to a base station (in our case a smartphone). We used the BITalino Board in a configuration that acquires EMG and ACC (Z-axis) signals (10 bits @ $fs = 100$ Hz), streaming the raw data via Bluetooth with a baud rate of 115.200 bps to the smartphone. Given that BITalino only outputs raw data, we devised a set of algorithms to convert this data into meaningful events, following the virtual sensor creation strategy.

5.3.1 Muscle Contraction Detection

For EMG data, we used an onset detection algorithm consisting of two stages, namely a processing block to filter the signal and compute the envelope of the EMG signal, and a decision rule block (see Fig. 5.6). The processing block uses a sliding window of $M = 40$ samples to perform a moving average filtering of the signal as described by Equation 5.1 $x[n]$ is the input signal (EMG), $s[n]$ is the filtered signal, and M is the sliding average window size. For a sampling frequency of 100 Hz, the -3 dB cut-off frequency is approximately 30 Hz and the gain is 2 at 0 Hz. The phase shift is 20 ms (two samples):

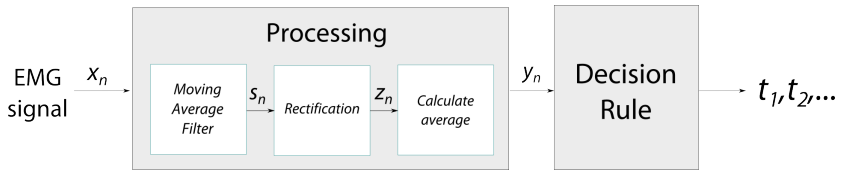


Figure 5.6: Signal Processing for the EMG

$$s[n] = \frac{1}{M} \sum_{j=0}^{M-1} x[n-j]. \tag{5.1}$$

Afterwards, signal rectification is performed by subtracting the DC component $s[n]$ and computing the absolute value of each point within the sliding window, as defined by Equation (5.2).

$$z[n] = \left| s[n] - \frac{1}{M} \sum_{j=0}^{M-1} s[n-j] \right|. \tag{5.2}$$

The processing stage finishes by computing the average value of the sliding window, as described in Equation 5.3 (where $M = 40$). Finally, we obtain a resultant value, $y[n]$ for each M samples of the raw input signal. An example, biceps contractions, can be seen in Fig. 5.7.

$$y[n] = \frac{1}{M} \sum_{j=0}^{M-1} z[n-j]. \tag{5.3}$$

Finally, a simple decision rule is applied to $y[n]$ to determine if there is a muscle contraction when $y[n] > threshold$. The *threshold* value is defined as a percentage of what is called the Maximum Voluntary Contraction (MVC). In the calibration stage, the user is asked to voluntarily contract each muscle being monitored in order to determine the maximum muscle activation amplitude he can produce. An expert, for instance the therapist in the rehabilitation scenario, defines the threshold manually as a percentage of the MVC. The fact that this is a relative value makes it insensitive to variable factors between exercising sessions (e.g. skin moisture, different electrodes, different devices, etc.).

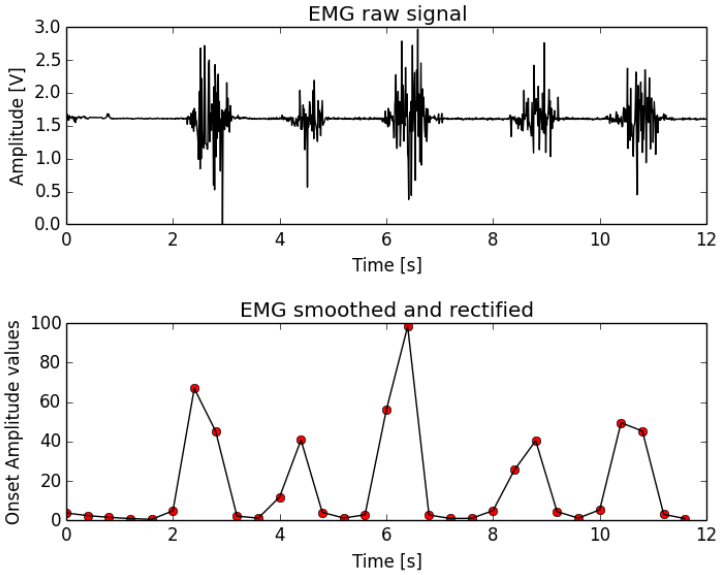


Figure 5.7: EMG signal used to evaluate the adopted algorithm. The algorithm facilitates the onset detection

This functional Block is called Muscle contraction. The input is an EMG raw signal from a muscle and returns "contracted" or "relaxed" values if a certain *threshold* is passed or not. The block has the following Normal Block structure:

```
"MuscleContraction:<label>":{
  "config":{
        "threshold":<number>
   },
   "in":{
    "1":<input_label>
   },
  "out":{
    "1":<output_label>
  }
}
```

Listing 5.1: DPL for muscle contraction

5.3.2 Limb Tilt and Motion Detection

The strategy used for ACC data processing follows the same approach. However, the processing stage consists only of a low-pass filter implemented with a moving average filter, as described by Equation 5.1. Similarly to the EMG approach, the ACC decision rule is applied to determine the ACC position in each instant and compare it with a threshold value interval. The thresholds for the ACC were calculated by the analyzing the values for the processed data and mapping them with the limb positions. As an example of limb tilt, Figure 5.8 shows three positions of an accelerometer attached to the wrist.

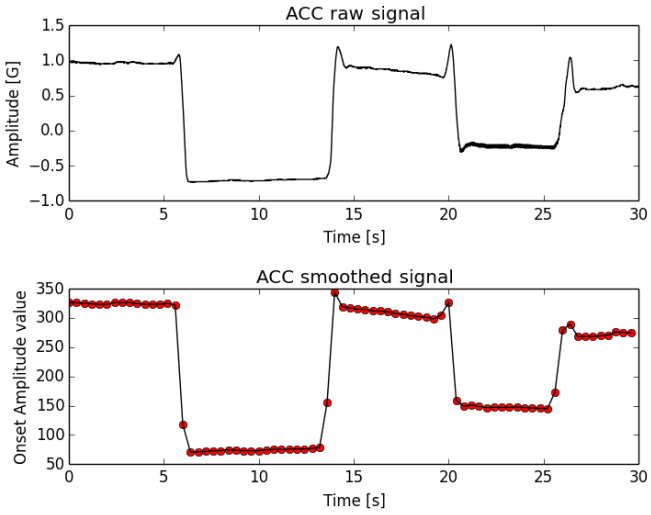


Figure 5.8: ACC signal used to evaluate the adopted algorithm

This functional Block is called TiltMovement, the Accelerometer signal from a joint produces an input if the joint is in a specific zone delimited by the values *threshold_zone*1 and *threshold_zone*2. The block returns the Strings "low", "mid" or "high" values.

```
1  "TiltMovement:<label>":{
2    "config":{
3          "threshold_zone1":<number>
4          "threshold_zone2":<number>
5     },
6     "in":{
7       "1":<input_label>
8     },
9    "out":{
10      "1":<output_label>
11   }
12 }
```

Listing 5.2: DPL for tilt movement

5.4 Application 1: ToBITas

5.4.1 Motivation

ToBITas [35] is a proof of concept for a Context-Aware application that controls a mobile robotic platform using physiological sensors. Users can activate different responses in the robot in real time using two EMG channels of the BITalino and one ACC channel. For this purpose the context information gathered with MobileBIT is used to decide which command operates the robot. Table 5.1 contains a detailed description of the interaction.

Table 5.1: Relationship between the acquired signals, context information and system behaviour

Signal	User Action	Context Information	Robot Command
EMG_1	The user folds his arm	Action_detected: Right_arm_folded	Move Forward
EMG_2	The user closes his hand	Action_detected: Hand_Closed	Open/Close the Claw
	Tilt the wrist to the left	Position_detected: Wrist_up	Move Right
ACC	Tilt the wrist to the right	Position_detected: Wrist_down	Move Left
	Wrist in central position	Position_detected: Wrist_side	Don't Move

5.4.2 Methods

The ToBITas Case Study was created to test the adaptation of the user to physiologically-enhanced sensor input methods. In addition to this, user satisfaction was also measured. For that purpose we performed an exploratory study evaluating the interaction of the participants with ToBITas completing a simple robot control task. We had two research questions:

- Are the users able to control our system?
- Do users feel comfortable with this kind of control?

For the former, the times to complete the task were measured for each participant in order to compare the learning effect between participant groups. For the latter, the participants were asked to complete the System Usability Scale questionnaire to obtain insights into the user satisfaction.

Participants

We recruited thirteen volunteers (four females) from the surrounding research laboratories of the IST-UL university campus. The participants ranged from 21 to 39 years old and all of them were right-handed. They were divided in three groups by their level of expertise and equipped with similar devices and applications:

- Group A (Novices). For seven participants for whom it was the first time that they had tried this kind of user interface.
- Group B (Experienced). Four participants who had reported to some previous experience controlling similar systems but for whom it was the first time they had used this system.
- Group C (Experts). Two participants who were involved in the design and development of the system. They were included in order to serve as a performance reference.

Apparatus

- A BITalino board was used for the EMG and ACC data acquisition.

- Bot'n Roll One was the mobile robotic platform.
- The smartphone was an LG Optimus F5 with Android 4.1.2, a Dual-Core 1.2GHz processor and 1GB RAM.

Procedure

To start, the demographic background of each user was noted. Subsequently, the BITalino device was set up and the electrodes were placed as mentioned in Figure 5.4. Afterwards, the calibration phase was carried out with the help of the researchers. Then, each user was asked to carry out the following routine:

1. Move the robot one meter to approach the cylindrical object
2. Grab the cylindrical object with the claw
3. Move the robot one meter to the target
4. Release the cylindrical object at the given position

The tasks is depicted in Figure 5.9. In addition, a demo video of the task can be seen in `http://sipt07.si.ehu.es/bgamecho/ToBITas/Demo_video.mp4`

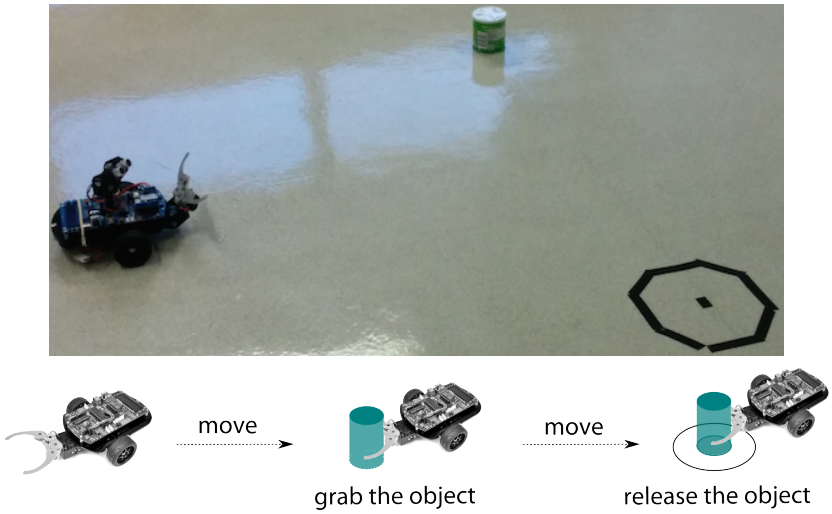


Figure 5.9: Experimental set-up and task description for the evaluation of ToBITas use case

Each participant performed the task three times. The performance was recorded on video and stored altogether with log data from the application. Finally, participants of Groups A and B completed the SUS questionnaire, Group C participants did not complete the SUS questionnaire to avoid a conflict of interests.

Results and Discussion

All the participants in the experiment were able to finish the proposed task. The completion times for the task are listed by group in Table **??**. The time required to complete the task was under 100 seconds for all the users in Group B, while in group A four users spent more than 100 seconds.

Table 5.2: Summary of the task results measured in seconds (T_n refers to the attempt)

Group	T_1 [s]	T_2 [s]	T_3 [s]	μ [s]	σ [s]
A	140	93	57	96	59
B	32	45	48	42	7
C	25	38	24	29	6
Average	89	70	49	69	16

There was no learning effect for participants of Group C. This fact is explained because those participants learned how to use ToBITas during the development and testing time. For this reason we decided to use the 6 values of the Experts group as a baseline reference for the results of Group A and the Group B. The value of the baseline is 29 ± 6 seconds. In the Figure 5.10 the learning effect is noticeable for Group A participants.

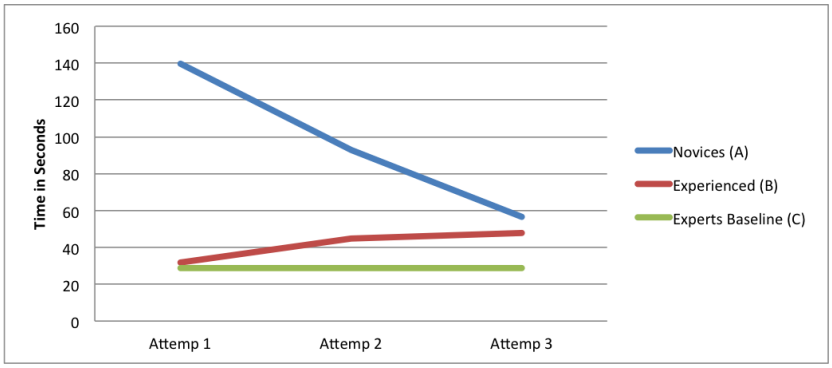


Figure 5.10: Progression chart for the results time

Depicting the times for each group in a box plot visualization, it also can be seen that each group differs from the others when considering the $\mu \pm \sigma$ values. This can be seen in the Figure 5.11. Thus, in a first approach, the previous experience with this kind of interaction techniques does have an effect on the completion time.

With regard to the SUS questionnaire, the average score is $73,86 \pm 12,58$ which is over the 70 required to consider the usability of the system as good. Four of the participants marked the system with a lower score than 70 and only one of those four scored the system lower than 60. It was noticed that this participant had lower thresholds during the calibration process compared to

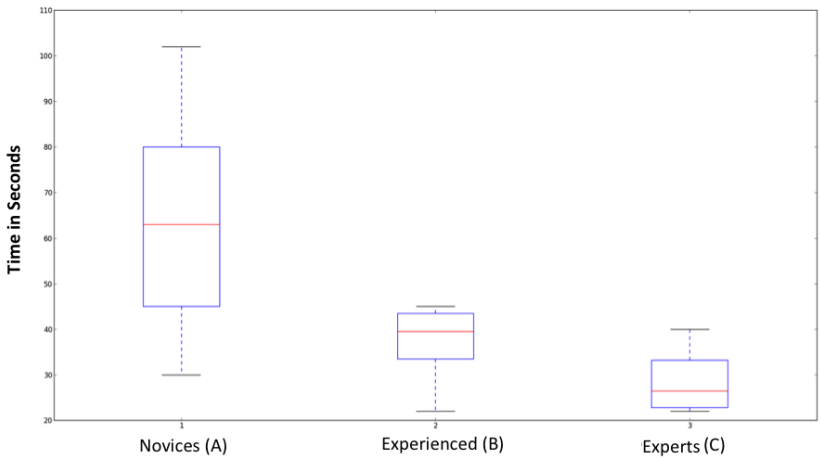


Figure 5.11: Box plot for the times in each phase

other participants, which made the system more sensitive to noise. Therefore, on some occasions, the processing algorithm did not follow the participant's demands and expectations. However, the bad result might be due to noise within the acquisition scenario, or bad electrode placement. Surprisingly, in the third run all the participants from both groups A and B spend approximately around 55 seconds to perform the task. This suggest that after two trials user Group A obtained a similar level of experience as user Group B.

On the other hand, some flaws were detected in the system that must be addressed in future versions. For instance, it was reported that the tilt movement of the wrist associated to a right turn by the robot sometimes also activated the thresholds for the biceps and the thenar eminence muscles. This unintended movement was noticed by some of the participants who started to move their arm more accurately. Others instead followed a strategy of "only turning to the left" to avoid the undesired movement. New DSP algorithms combining three signals (EMG1, EMG2, ACC) should be tested to fix this issue.

The results showed that users were able to understand and quickly learn how to use our approach. ToBITas is a functional and usable Context-Aware application, proving that Igerri is able to provide valid Context-Aware applications. The input method based on EMG and ACC sensors performed well for the control of a mobile robot. Moreover, since the users were able to shorten the experience time in each repetition, this seems to indicate that the users were able to adapt themselves to the system quickly.

5.5 Application 2: Rehabilitation Exercise System (RESapp)

The second application developed with the Igerri Framework RESapp [38] was designed as a tool for rehabilitation therapy.

5.5.1 Motivation

Every year, a great number of people worldwide undergo physical rehabilitation due to work related injuries [1], disability [2], and other conditions. Within the portfolio of tools that therapists currently have at their disposal, biofeedback has become particularly popular [40, 71], with clinical evidence showing that it is an engaging technique with multiple benefits for the patient [12, 40, 47, 71].

As a way of improving the effectiveness of treatments, the extension of the rehabilitation process to people's homes with biofeedback exercises has been proposed. These exercises would be designated by the therapist to be performed autonomously by the patient at home between sessions at the clinic [20, 59, 60].

A major advantage of biofeedback is the possibility of deriving objective performance indicators from direct physiological measurements in real time, and using them for live progress monitoring and guidance of the rehabilitation exercises to maximize the potential outcomes. A critical aspect in any biofeedback system are positive and negative reinforcement cues, provided to the patient in real time when a given exercise is being executed. These are used as a way of confirming whether the goals set by the therapist are being met or not.

As recently argued by Chandra, Oakley and Silva [16], physical rehabilitation at home using biofeedback is hindered by multiple factors, user engagement being the most important. Positive and negative reinforcement cues provided to the user during the exercises play a major role in user engagement.

Reinforcement cues used in biofeedback are generally visual, acoustic, and/or haptic signals delivered through a computer or mobile device acting as the interface with the patient. However, aspects such as screen size, input interfaces, lack of technological proficiency, reduced sight or hearing abilities, and other factors associated with the more traditional feedback methods can pose difficulties to specific user groups (such as elderly patients), ultimately leading to poor compliance in home exercises.

5.5.2 Proposed Approach

The animatronic biofeedback presented in ToBITas was adapted for a telerehabilitation scenario with elderly users. The main change is the substitution from

the continuous control mode to a step by step one, which proved to be more appropriate for the performance of rehabilitation exercises.

In this second approach, only one movement at a time is allowed, and consequently the mobile application is waiting to finish the exercise to allow the next movement. In this way we can overcome the problems regarding the multiple activation of the different movements mentioned in the discussion of the previous evaluation. Another difference with the previous experiment is that the EMG signal from the thenar eminence muscle was eliminated because it was not a good choice for the type of exercises envisioned. Both applications use the BITalino board for the acquisition of EMG and ACC.

The system recognizes two kinds of exercises performed with the right arm and triggers different actions in the animatronic:

- Activating the biceps (e.g. folding the arm) for T seconds moves the robot forward for a distance of 40 cm.
- Tilting the wrist to the left or right for T seconds animates the robot to turn in the same direction for 90 degrees.

Ideally physiotherapists define routines which chain these exercises. Usually the same exercise would be repeated a number of times. If the exercise is interrupted, the system will activate the negative biofeedback cue. After the negative biofeedback cue, the user must always repeat the last exercise again. We consider the following events as interruptions:

- For muscles: When the contraction value is under the threshold set up in the calibration phase.
- For tilt positions: When the value of the accelerometer is outside the region defined for an exercise.

The application was developed following a life cycle of two iterations. Additionally, in every iteration, the animatronic biofeedback was compared to an alternative method. The objective was to ascertain if the animatronic biofeedback presented advantages over other feedback methods.

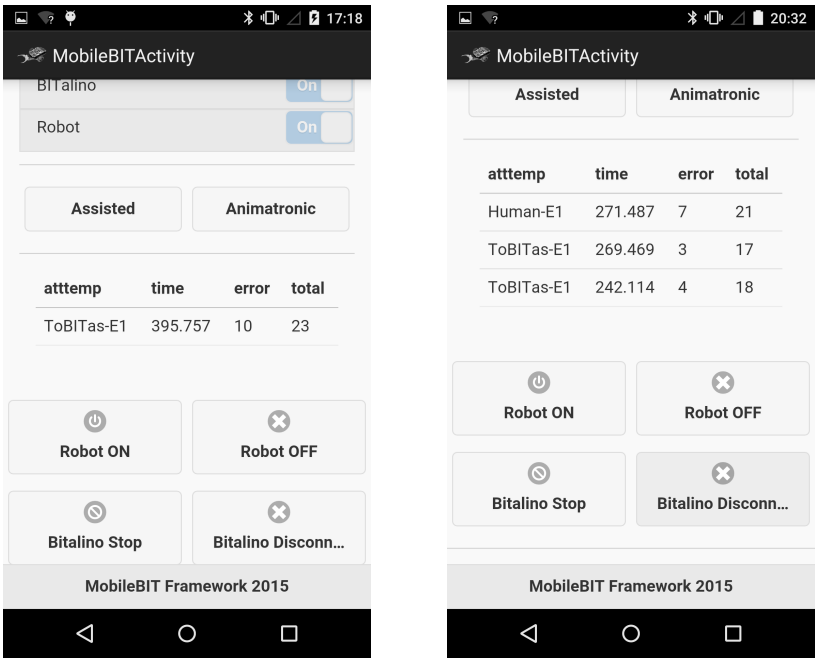
- Iteration 1 was a pilot test under laboratory conditions with 5 users. The animatronic biofeedback application was compared to human user feedback.
- Iteration 2 was improved with the feedback from the first iteration. The animatronic biofeedback positive and negative cues were changed. Then it was tested in an association of retired people with 11 participants. For this iteration the animatronic biofeedback was compared with a visual biofeedback.

In order to design the exercises, both biceps contraction and wrist movements were designed. They were chosen following the previous experience of the researchers of IST-UL who have experience in this area. It must be clearly stated that no medical benefits can be claimed using this evaluation, since it has been designed to check the usability of the approach rather than check if

the animatronic biofeedback can be used as a valid medical therapy. All the participants were informed about this concern before starting the experiments. Following this rationale, the value of variable T was chosen regarding that the exercises would be repeated several times in order to minimize the fatigue of the participants in the experiment.

5.5.3 Iteration 1

In the first iteration, the researcher in charge of the evaluation controls the test using the smartphone device that controls the BITalino and the Bot'n Roll. After the test, a semi-structure interview was recorded to obtain more information about the participants opinion. In Figure 5.12 some screenshots are included from the user interface.



Screenshot a **Screenshot b**

Figure 5.12: User interfaces for the first iteration

The positive animatronic biofeedback cue consisted of waiting until the exercise was adequately performed to move the robot depending on the exercise. The biceps contraction required a $T = 20s$ and the turn left/right a $T = 10s$ to be considered as well performed.

The negative animatronic biofeedback cue for this iteration was to remain still without any movement. If the users do not see the robot moving it means that the exercise is not being performed well.

Due to the fact that the goal for the first iteration was to obtain a preliminary version for the animatronic biofeedback, the results are summarized in the following lines. The evaluation of the second iteration is fully described in the Section 5.5.5.

Pilot Testing Results

The pilot testing was conducted with 5 participants (2 males) from an age group of 70 ± 4 years old. All of them were able to complete a representative routine of exercises. The average completion time was 296.1 ± 75.6 seconds for the user feedback and 322.9 ± 89.55 seconds for the animatronic biofeedback. Regarding the number of errors, for the user feedback an average of 8 ± 6 errors were obtained and on the other hand an average of 8 ± 7 errors were obtained. The SUS evaluation obtained and average value of 84 ± 4.64 meant that the usability of the applications was perceived as excellent.

Discussion and Conclusion

The pilot testing was useful to understand that the animatronic feedback could be more effective as far as the positive and negative cues were concerned. A series of conclusions and lessons learnt were obtained from this pilot evaluation:

- Comparing the Human feedback with the animatronic biofeedback is pointless. Generally, human feedback is expected to be more effective and understandable than other biofeedback methods.
- The time required to perform the exercises should be lower. A period of 20 seconds for the biceps contraction is not adequate for this evaluation because it is quite hard for the users to maintain since fatigue and tiredness is accumulated during the evaluation session. Considering that this is not a real physiotherapy session the tiredness could be confounded with poor usability of the system.
- Animatronic Positive and negative reinforcement cues could be misunderstood. On the one hand, positive biofeedback should be represented as a continuous movement of the robot. On the other hand, negative biofeedback should not be merely that the robot remains still without moving the robot.

Regarding the use of Igerri and the appropriateness of the user interfaces and interaction methods, it shows that the developed applications are usable and functional for the 5 users.

5.5.4 Iteration 2

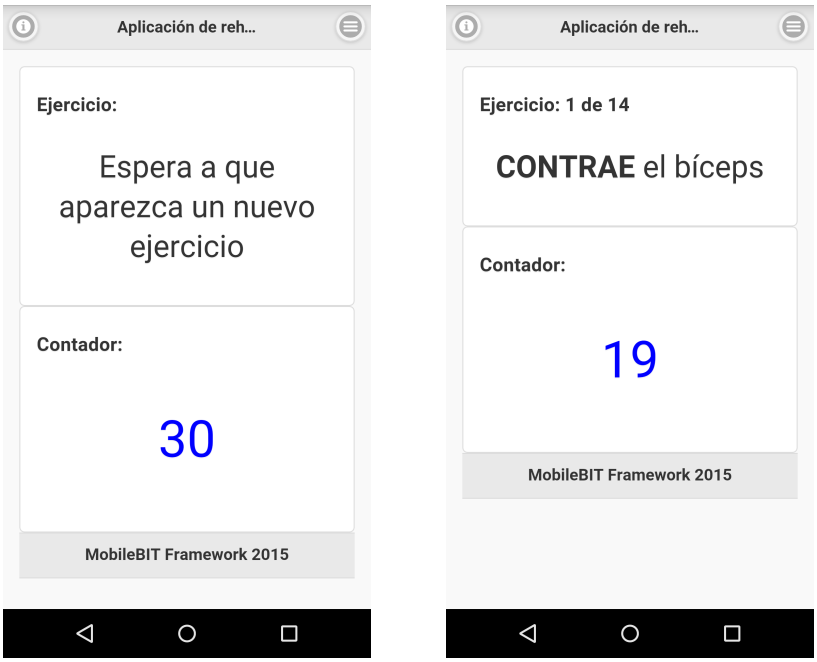
For the second iteration, the researcher carrying out the experiment, controls remotely the activation of the exercise routine and monitors how it is working (see Figure 5.13). We call this user interface the Therapist panel. For this iteration, two devices were used, one for the visual biofeedback test and another one for the animatronic biofeedback test.

Figure 5.13: Web interface to start and control the experiment from a remote Device

Visual Biofeedback Application

The Visual biofeedback application developed using MobileBIT relies on the indications in the screen of the mobile device to perform the exercises (see Figure 5.14). The screen showed a counter from 30 to 0. Visual biofeedback was implemented in the following way:

- Positive Visual cue: while the exercise was being performed well the counter counts down. Every 400*ms* the counter decreases by 1. Whenever



Screenshot a **Screenshot b**

Figure 5.14: User interfaces for the Visual Biofeedback application

the countdown reached 0 the exercise was considered to be successfully performed and the next exercise was prompted on the screen.

- Negative Visual cue: when an error was detected the counter resets to 30.

Animatronic Biofeedback for Rehabilitation

Using the Bot'n Roll features we programmed animatronic feedback actions that can be triggered as positive reinforcement cues when the user is correctly performing a given exercise. These movements can be to move forward for some centimeters, or to turn to the right or to the left for 90º. The negative reinforcement cue is common to all exercises, and is expressed by the robot going backwards to the place it was before the last movement. If the robot remains still without moving it could mean different things:

- After positive feedback: the current exercise has been performed well.
- After negative biofeedback: the last exercise can be tried again.
- In the remaining cases: the current movement has not been correctly performed yet.

5.5.5 Methods

For the evaluation of the RESapp scenarios, a comparative approach has been adopted between the two biofeedback methods:

- Method A: Visual Biofeedback application.
- Method B: Animatronic Biofeedback application.

We want to know what the differences are for both of methods regarding the time to complete the rehabilitation routines and the number of errors performing the exercises. We also want to study how the users perceive the usability of these applications and if they are functional and usable. We believe that this Context-Aware animatronic biofeedback perspective can greatly contribute to further enhance the set of tools available for therapists, especially in with regard to methods for increasing compliance to home exercises in elderly patients. Therefore the evaluation of RESapp scenarios were performed with retired people from 64 to 80 years.

Participants

11 volunteers (4 males) were recruited from an association of retired people. The participants age ranged from 64 to 78 years ($73.2 \pm 4, 4$). Informed consent was obtained from all individual participants included in the study.

Apparatus

- A BITalino board was used for the EMG and ACC data acquisition.
- Bot'n Roll One was the mobile robotic platform.
- For the Method A a Rikomagic MK902II mini-PC with Android 4.4 was used connected to a 14 inch monitor.
- For Method B, a Nexus 5 smartphone with Android 5.0.1

Procedure

First, demographical data were gathered by means of a short questionnaire. After that, the participants were helped to adjust the BITalino to the right arm. Subsequently a calibration phase was carried out to find out the MVCs and to define the activation threshold. The EMG electrodes were placed over the muscle fibres of the biceps, at a distance of approximately 2 cm. The reference electrode is placed on a bone area on the elbow. The ACC sensor is on the BITalino board, which is placed on the right wrist of the user, as shown in Fig. 5.4.

In our experimental evaluation we will consider an activation threshold around 20% of the MVC, this threshold should be changed to a lower one if the

participants shows fatigue while performing the experiment. Then, the users were required to perform a specific task. The task consists in a 14-exercise routine described in Table 5.3. Each exercise had to be maintained for $T = 12$ seconds to be considered successful. This task is repeated two times for each Method, A and B.

Table 5.3: The routine is composed of 3 different exercises in a sequence of 14 exercises

Type	Exercise description	Sequence order	Repetitions
1	Biceps contraction	1, 3, 5, 7, 8, 10, 12, 14	8
2	Turn wrist to the left	2, 4, 6	3
3	Turn wrist to the right	9, 11, 13	3

The descriptions are the following ones, in Figure 5.15 the experimental setup for both methods can be seen.

- In Method A, the participant had to follow the indications of the Android TV application to perform the task and complete the exercises. The screen showed the above mentioned counter from 30 to 0.
- In Method B, the participants were provided with a sheet on which the routine was described and they had to perform the exercises watching the robot. When required, the researcher indicated the next movement to the participants.

Method A **Method B**

Figure 5.15: Experimental Set-up for both methods. Notice the screen switched on for Method A and the Robot in the ground for Method B.

During the experiment, the time required to complete the routine and the number of errors made were recorded. After completing the tasks the participants were interviewed using two questionnaires: the System Usability Scale

(SUS) [13] and the preferences questionnaire described in Table 5.4. The questionnaire was filled in by a research assistant to avoid misunderstandings of questions and likert scales. After finishing the experiment every participant was rewarded with a ticket for a meal in the restaurant of the association.

Table 5.4: The second questionnaire is an 8 items likert scale with 7 answer options (1 totally disagree and 7 totally agree). Three categories are evaluated: User Satisfaction (US), User Awareness (UA) and the Location to apply the system (Loc).

Item	Category	Factor	Statement
1	US	Lack of difficulty	I'm satisfied with the ease with which I completed the task.
2	US	Perceived time to complete the task	I'm satisfied with the time I spent to complete the task.
3	US	Comfortability	I've felt comfortable using the system.
4	US	Amusement	I've enjoyed using this system.
5	UA	Biceps movements	I was aware that the biceps contraction exercise performed well.
6	UA	Wrist movements	I was aware that the wrist tilt exercise performed well.
7	Loc	Rehab. center	I would like to use this system in a rehabilitation center.
8	Loc	Home	I would like to use this system at home.

Design

The experiment followed a within-subject design. Biofeedback Method condition was counterbalanced using a Latin square. For the same routine two biofeedback Methods were tested, visual biofeedback and animatronic biofeedback, and two attempts are performed to mitigate the learning effect. The routine was composed of three different types of exercises: 8 biceps contraction, 3 right wrist movements and 3 left wrist movements. Aside from calibration, the amount of expected entry was 11 participants $\times$ 2 Methods $\times$ 2 attempts $\times$ 14 exercises/attempt = 616 exercises (352 biceps, 132 left, 132 right). For each routine, a minimum time of 14 exercises/routine $\times$ 12 seconds/exercise = 168 seconds/routine are expected.

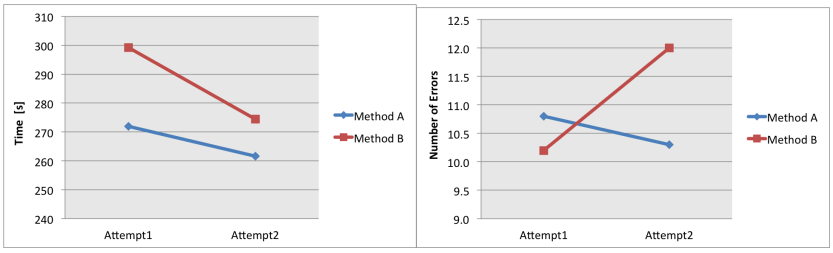
Results and Discussion

One of the volunteers did not finish the experiment because she could not maintain the required biceps contraction for $T = 12$ seconds. Since that participant did not complete the routine, the final questionnaires were not filled in. The rest of the participants ($N = 10$) successfully completed the experiment.

Concerning the perceived usability of the system, the obtained SUS scores were 88.5 ± 7.2. Therefore, we can conclude that participants considered our system to be usable. Regarding the required time to finish all the exercises and the error rate, we found that the results were similar for both Methods (see Table 5.6 and Table 5.7).

With reference to the learning effect, in the second attempt, the time decreased by 11 seconds (4%) for Method A and 25 seconds (8%) for Method B.

Conversely, the number of errors decreased by 0.5 (5%) for Method A but increased by 1.8 (12%) for Method B (see Fig. 5.16). The reduction in time by 25 seconds along with the slight increase by 2 in the number of errors, could suggest that the participants learnt how to deal quickly with the errors after the animatronic biofeedback during the second attempt.



(a) Time to complete the routine (b) Number of errors

Figure 5.16: Progression charts for the two methods regarding the time and number of errors

To further compare both methods, we consider the second attempt as a repetition. Therefore, we increased the sample to $N = 20$ values for each method. Afterwards, using the Shapiro-Wilk Normality Test we found that the data did not follow a normal distribution for Method A. Consequently we apply a Wilcoxon Signed-Rank test to analyse differences for the completion times and the number of errors between Method A and B. Regarding the time required to complete the routine, the average values of Method A and Method B were 249.93 and 291.56 respectively. A Wilcoxon Signed-rank test suggests that there is no significant effect for Method ($W = 72$, $Z = -1232$, $p > 0.05$). With respect to the number of errors, the average values of Method A and Method B were 7.5 and 8.5 respectively. A Wilcoxon Signed-rank test shows that there is no significant effect for Method ($W = 94.5$, $Z = -0.392$, $p > 0.05$). These results, while not conclusive, do suggest that for this experiment the participants performance was identical in terms of completion time and errors for both animatronic and visual feedback.

Finally, the results of the interview and questionnaires show that both systems are perceived similarly (see Fig. 5.17). Nevertheless there is a noticeable difference for the *Amusement* item of the animatronic biofeedback. For this case, the medians of Method A and Method B were 5.5 and 7 respectively. A Wilcoxon Signed-rank test was applied to a sample of $N = 7$ data (the test ruled out 3 participants for scoring both methods with the same value). Under these conditions, the Wilcoxon Signed-rank test shows that the Method does have a significant effect ($W = 2$ and the critical value of W for $N = 7$ at $p \leq 0.05$ is 2). This result is reinforced by some comments made by participants in the interview: "*The robot is a toy*", "*It's funny*". In addition, one participant also said that he felt more comfortable with the screen but that it was boring comparing to the robot. As the main drawback, most of the participants believed that the Visual feedback was more appropriate for home due to the space required

for the movements of the robot. A viable solution for this concern can be the use of a different robot instead of the Bot'n Roll. For instance, Rodriguez et al. 2014 [63] propose the use of NAO humanoid robots to mimic the movements of a human operator. This solution is more efficient because it does not need more space than the required by the robot.

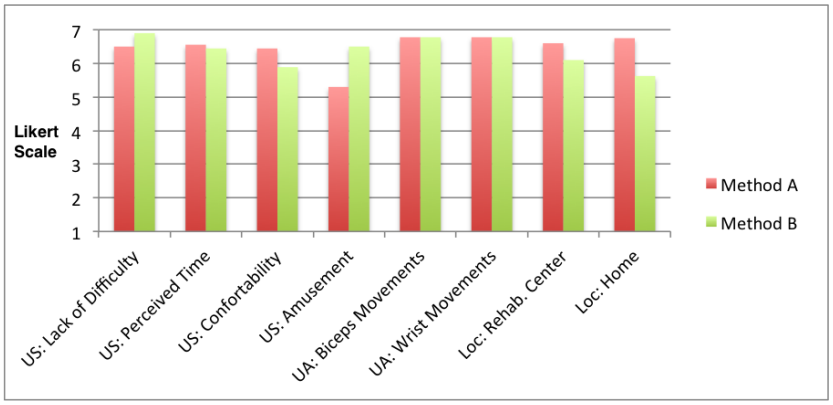


Figure 5.17: Mean values obtained for the results of the second questionnaire (see Table5.4)

5.6 Conclusion of the Usability Testing

This chapter describes a series of experiments for two context aware applications developed using the Igerri conceptual framework and their implementation. Quantitative and qualitative results were obtained and overall, the applications were revealed to be fully functional and usable by the participants during the experiments.

Regarding the task performance and the errors, on the whole 27 of 29 participants finished the evaluation without problems. From a total of 13 participants, only one reported problems using ToBITas and another one had problems using RESapp. In both cases, the thresholds were very low and this made it impossible to use the applications. This means that the functional blocks described in section 5.3 were valid for 93% of the participants.

With regard to the task completion time using ToBITas, novice users were able to reduce the completion time of the proposed task over three attempts. With regard to the RESapp, despite the fact that a comparative study was conducted, both scenarios were developed with MobileBIT and they obtained a similar completion time and error rate which indicate that the addition of a new element in the MobileBIT such as the robot did not lower usability.

With regard to the qualitative evaluation the SUS obtained acceptable results as can be seen in Table 5.5. The SUS questionnaire was carried out by 26 participants.

Table 5.5: A summary of SUS evaluation results

Application	Evaluation	Participants	Age	Environment	SUS Value
ToBITas	Continuous control	11	20-40	Lab conditions	$73,86 \pm 12,58$
RESapp	Step by Step Pilot	5	64 - 80	Lab conditions	84 ± 4.64
RESapp	Step by Step Final	10	64 - 80	In the field	88.5 ± 7.2

Overall, the evaluation shows good results and personal interviews and questionnaires reported that the participants very quickly understood the concepts regarding the proposed control mode. It seems that the approach followed in this thesis produce:

- Functional Applications: All the functionalities tested regarding with sensors worked for 93% of the participants.
- Usable Applications: Participants graded ToBITas application with good usability and RESapp with excelent usability.
- Appealing Applications: Participants of RESapp application experiment reported high values for different categories regarding user satisfaction and awareness of using the rehabilitation application.

This way research questions stated at the beginning of the chapter are answered. With regard to the hypothesis, we conclude that the implementation of the Conceptual Framework tested in this chapter is adequate to produce usable Context-Aware applications.

Table 5.6: Mean times for complete the routines in seconds

Participant	Attempt 1		Attempt 2	
	A (Visual)	B (Robot)	A (Visual)	B (Robot)
1	209.3	242	226.9	214.8
2	230.1	3144	220.1	351.2
3	235.7	258.8	230.1	228.4
4	343.4	289.2	274.2	328.9
5	319.7	300	338.5	250.4
6	259.7	360	247.4	310.8
7	373.8	241.6	279.3	294
8	280.4	408	345.7	300.4
9	252.5	239.2	212.5	232.8
10	214.9	339.3	241.7	232.4
μ	272	299.2	261.6	274.4
σ	56.4	57.06	47.6	48.3

Table 5.7: Number of errors registered in the experiment

Participant	Attempt 1		Attempt 2	
	A (Visual)	B (Robot)	A (Visual)	B (Robot)
1	1	0	5	3
2	7	12	2	28
3	4	8	6	3
4	26	7	14	23
5	24	12	24	9
6	9	17	7	15
7	25	2	8	11
8	8	31	28	20
9	3	5	1	4
10	1	8	8	4
μ	10.8	10.2	10.3	12
σ	10.2	8.8	9.1	9.1

Chapter 6

Conclusion & Future Work

In this chapter, the results obtained in this thesis are summarized and discussed. The contributions to sensor abstraction and virtualization in the field of mobile Context-Aware computing are highlighted. Finally, future lines of research are presented.

6.1 Conclusion

This PhD presents Igerri, a conceptual framework for sensor abstraction and virtualization. This framework is composed of a set of layers with different levels of abstraction. From a bottom up perspective, context information is transformed from raw data into context information of a higher level of abstraction. This is achieved thanks to virtual and abstract sensors. This conceptual framework was designed with the goal of facilitating the design of usable context aware applications for developers. In addition to this, the framework layers were designed to conceal from developers the difficulties of designing applications for mobile Context-Aware computing.

The implementation shares concepts with the Igerri framework and facilitates the creation of Context-Aware applications. To test our hypothesis, the conceptual framework was instantiated in two components named, MobileBIT and PervasiveBIT. These software components were implemented for a mobile platform and used an external server to achieve seamless integration across the devices of the ubiquitous system. Igerri can be implemented in many ways, and it was decided to follow this approach because it provides enough mechanisms to deal with the objective of the framework, the sensor abstraction and virtualization.

Two sensor enhanced applications were developed for the evaluation of the resulting implementation. Both use external sensors in a wearable platform to test the generation of context, the abstraction and the virtualization of sensors, in terms of perceived usability. The use of usability testing techniques

provides not only a way to measure the performance of the applications (bad performance leads to bad usability) but also show whether the applications are appealing or not for the users. A set of experiments were conducted with 29 participants producing good results.

The evaluation for the framework was carried out testing the applications. However, the results obtained cannot be generalized to all the applications implemented following our framework. Anyway, the good results in the perceived usability of the applications show that Igerri was useful to create context-aware applications using the proposed implementation.

Consequently, the hypothesis presented in the research questions was satisfied (partially validated) by the implementation of Igerri and the proposed applications: It is possible to create usable (1), controllable (2) and appealing (3) sensor-enhanced context-aware applications based on the mechanisms described by Igerri for sensor abstraction and virtualization. Although this fact doesn't ensure to every application created with Igerri to be valid for the properties (1 to 3), it evidences that these properties can be provided using Igerri. In the remaining sections, the relationship with Egoki, contributions, limitations on the followed approach, and possible future work are described.

6.2 Igerri as an Extension for Egoki

In the first chapter, it is suggested that the motivation for this research was an extension for the Egoki user interface generator. The original design of Egoki was not intended for Context-Aware computing but for the generation of user interfaces tailored to people with special needs using different adaptation techniques. Four problems or challenges were proposed to stretch Egoki regarding its limitations. How Igerri can be used to deal with these limitations is explained below:

- With regard to the use of **new input modalities**, Igerri, as was probed in the evaluation, can be used to gather information from sensors such as EMG or ACC to present a gesture based interaction. In the same way, other sensors can be taken into consideration to extend the input modalities available in Egoki.
- On the topic of the **adaptability** of Egoki user interfaces, the use of data from sensors (located in the environment and user devices) can improve the existing adapting techniques. Igerri and its implementation provide access to the context information generated using sensors. This context information is a valuable input for Egoki in order to generate adapted user interfaces available through Igerri.
- In relation to **proactive** applications and the interaction without the use of graphical user interfaces, ToBITas is a good example of the capabilities of Igerri. It is possible to use Igerri-based applications to deal seamlessly

with different devices (i.e. robots, wearables, mobile devices), and allow the user to control them without directly manipulating the mobile device.

- Concerning the use of **mobile applications** instead of web applications, the proposed implementation of Igerri is based on a Hybrid application approach. The proposed user interfaces rely on Web technologies but also use all the resources of the mobile devices. Therefore, it is possible to generate Egoki web based user interfaces and to create code to interact with the libraries implemented in MobileBIT and PervasiveBIT.

Igerri was not only designed as an extension to Egoki but Egoki was its main motivation. With the proposed implementation and its evaluations it can be concluded that the work presented in this thesis contributes to improve context awareness, suitable for the adaptation of the user interfaces generated by Egoki.

6.3 Contributions

As a result of the research and development work carried out for this thesis, the following contributions can be outlined:

- The **conceptual framework** itself is a useful abstraction to deal with sensors in mobile context-aware environments. The transformations between layers is suitable for describing and understanding how a hierarchy of sensors can be created to manage context information. The access to context information is simplified to a request call to the top layer of the framework.

- This thesis also proposed an **implementation** of Igerri composed of two separate software components working together, namely MobileBIT and PervasiveBIT. They facilitate the creation of applications and provide methods to extend the context information. In order to provide context awareness to applications these methods conceal programming issues regarding the sensors.

- MobileBIT was originally proposed in Cânovas et al. [15] as a framework intended to create sensor-enhanced applications for eHealth domain. This thesis introduces the **MobileBIT extension for context aware applications** as a novel contribution. In addition to this, the existing implementation of MobileBIT was refined and optimized in order to be useful for context-aware applications. Finally, guidelines to improve the performance of MobileBIT applications were provided.

- The **evaluation results obtained for the usability testing** of the applications suggest that the implementation of Igerri can produce useful context-aware applications. This does not fully validate the framework however results gives evidence that it is possible to build the intended applications. Applying the usability techniques considered in Chapter 5, it

would be possible to evaluate additional context-aware applications created with Igerri.

- Another result of the evaluation is the **animatronic biofeedback** that was tested as an example of a sensor enhanced application. This contributes to the research in the area of biofeedback, introducing a way to interact with the physiotherapy applications in an unattended setting.

These results, led to the following publications:

1. A Context-Aware Application to Increase Elderly Users Compliance of Physical Rehabilitation Exercises at Home via Animatronic Biofeedback. **Gamecho B.**, Silva H., Guerreiro J., Gardeazabal L., Abascal J. Selected and submitted to the Journal of Medical Systems. [38]

2. Automatic Generation of Tailored Accessible User Interfaces for Ubiquitous Services. **Gamecho B.**, Miñón R., Aizpurua A., Cearreta I., Arrue M., Garay-Vitoria N., Abascal J. In: Human-Machine Systems, IEEE Transactions on, PP(99):1–12. [37]

3. Evaluation of a Context-Aware Application for Mobile Robot Control Mediated by Physiological Data: The ToBITas Case Study. **Gamecho B.**, Guerreiro J., Alves A.P., Lourenço A., Silva H.P., Gardeazabal L., Abascal J., Fred A. In: Proceedings of 8th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI'14). [35]

4. Design Issues on Accessible User Interface Generation for Ubiquitous Services through Egoki. **Gamecho B.**, Miñón R., Abascal J.In: 12 European AAATE Conference (2013). [36]

5. Combination and Abstraction of Sensors for Mobile Context-Awareness. **Gamecho, B.**, Gardeazabal, L. and Abascal, J. In: Adj. Proc. UbiComp'13, Ubiquitous Mobile Instrumentation (UbiMI'13). [33]

6. A Context Server to Allow Peripheral Interaction. **Gamecho B.**, Gardeazabal L. and Abascal J. Peripheral Interaction: Embedding HCI in Everyday Life. Workshop at INTERACT 2013 the 14th IFIP TC13 Conference on Human-Computer Interaction. [34]

7. Augmented Interaction with Mobile Devices to Enhance the Accessibility of Ubiquitous Services. **Gamecho B.**, Gardeazabal L., Abascal J. In: Mobile Accessibility (MOBACC 2013). Workshop at CHI 2013. [32]

8. Extending In-home User and Context Models to Provide Ubiquitous Adaptive Support Outside the Home. Aizpurua A., Cearreta I., **Gamecho B.**, Miñón R., Garay-Vitoria N., Gardeazabal L. and Abascal J. In: Martín E, Haya PA, Carro RM (eds) User Modeling and Adaptation for Daily Routine, Springer-Verlag. [7]

9. Automatically Generating Tailored Accessible User Interfaces for Ubiquitous Services. Abascal J., Aizpurua A., Cearreta I., **Gamecho B.**, Garay-Vitoria N. and Miñón R. In: The 13th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2011. [6]

10. Testing A Standard Interoperability Framework In An Ambient Assisted Living Scenario. **Gamecho B.**, Abascal J. and Gardeazabal L. In: Proceedings of 5th International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI'11). [31]

11. Model-Based Accessible User Interface Generation in Ubiquitous Environments. Miñón R., Abascal J., Aizpurua A., Cearreta I., **Gamecho B.**, Garay-Vitoria N. In: INTERACT 2011, the 13th IFIP TC13 Conference on Human-Computer Interaction. [54]

12. Generación de interfaces de usuario accesibles para entornos ubicuos, basadas en modelos. Miñón R., Abascal J., Aizpurua A., Cearreta I., **Gamecho B.**, Garay N. Interacción 2011 (XII Congreso Internacional de Interacción Persona-Ordenador). Actas del Congreso, Ibergarceta S. L., pp. 145-154. ISBN: 987-84-9281-234-9. Lisboa (Portugal). 2011 [53]

13. Some Issues Regarding the Design of Adaptive Interface Generation Systems. Abascal J., Aizpurua A., Cearreta I., **Gamecho B.**, Garay-Vitoria N. and Miñón R. In: The 14th International Conference on Human-Computer Interaction (HCII 2011). [5]

6.4 Limitations of this Thesis Work

This thesis addresses an important challenge in the area of ubiquitous computing, the heterogeneity of devices, sensors and platforms. This work focuses on mobile context-aware applications. This challenge is far from being fully resolved. There are still open issues and limitations in our approach.

To test our hypothesis Igerri was proposed to overcome the problems stated in Chapter 1. Nevertheless, due to the general purpose of the conceptual framework, Igerri lacks of specific mechanisms to deal with concerns that must be considered in the implementation. Other limitations are consequence of decisions took when the implementation was designed and programmed. This implementation has a clear goal stated in the first chapter, to demonstrate the usability of the applications and deal with programming issues concerning sensors in context-aware applications. The implementation has the following limitations:

- It is not possible to share context-information across different apps in the same device. The implementation only allows one application each time to get the context-information. For instance, this makes not possible to have background applications in the smartphone using the same context information.

- The use of an external server to implement PervasiveBIT makes Igerri applications dependant of the availability of the server to obtain the information about the specific sensors. On the other hand, once the DPL file is obtained, PervasiveBIT is not more required to obtain the additional information.

- Regarding the number of sensors available in the system, our implementation considers only the sensors discovered when context aware applications are launched. Changes in the sensor list has no effect in previously launched context-aware applications. This limitation, for instance, does not allow to switch Virtual sensors if a sensor crashes.
- There is no mechanism to deal with situations where two virtual sensors with the same information are presented. So far, the implementation chooses always the first sensor available that suits the restrictions in order to instantiate the virtual sensor.

Nevertheless, these limitations were not critical for testing the proposed approach.

6.5 Future Work

Even though the work introduced in this thesis covers a considerable number of topics and research challenges there remain a number of research lines that could be undertaken:

- **Testing Igerri implementation using more than one output for a Virtual Sensor**. So far, the evaluation of this work includes digital signal processing techniques to produce virtual sensors regarding one input sensor (See Section 5.3). It would be interesting to implement Virtual sensors that needs techniques such as Sensor Fusion, Machine Learning or Fuzzy Logic.
- **Combining with Egoki to create accessible smartphone applications**. That is, using it alongside Egoki to generate applications with multimodal user interfaces. As argued in Section 6.2, Egoki can benefit from Igerri in numerous ways to improve the accessibility of the user interfaces. In the same way, Igerri can take advantage of Egoki and use it to present different versions of the user interfaces in the same application. For instance, swapping the visual and animatronic biofeedback of RESapp to an audio modality when the user requires it.
- **Extension for distributed context-aware applications**. Context-Aware applications described in this thesis are currently running on a single device. Even if they are able to use different sensor devices and platforms, the interaction is limited to only one user. In order to fit better the definition of ubiquitous computing, distributed context-aware applications for many users should be studied.
- **Use two or more wireless sensing devices at the same time**. This scenario can lead to desynchronised data from sensors. It would be interesting to check how these scenarios can affect the performance of the implementations.
- **Lack of tools to create context-aware applications**. Both the conceptual

framework and the implementation are depicted in this thesis. The conceptual framework contributes to explain and understand how to achieve context-aware methods based on a layered approach. A graphical editor that translates this layered approach into code would be of great interest to facilitate the development of new virtual sensors or context services.

- **Test the implementation with developers**. The implementation of Igerri and more precisely the MobileBIT component is oriented to the rapid prototyping of context-aware applications. Nevertheless it has not been tested with developers. A formal evaluation involving Web designers and developers would be interesting to fully validate the mechanism implemented to solve programming issues regarding sensors.

References

[1] European agency for safety and health at work. musculoskeletal disorders: Key facts. https://osha.europa.eu/en/topics/msds/index_html/facts_html. Accessed: 31 March 2015. (Cited in page 79)

[2] UN world health organization (who): World report on disability. who/nmh/vip/11.01. http://www.who.int/disabilities/world_report/2011/report.pdf. Accessed: 31 March 2015. (Cited in page 79)

[3] AARTS, E., AND WICHERT, R. Ambient intelligence. In *Technology Guide*, H.-J. Bullinger, Ed. Springer Berlin Heidelberg, 2009, pp. 244–249. (Cited in pages 2 and 15)

[4] AARTS, E. H. L., AND DE RUYTER, B. E. R. New research perspectives on ambient intelligence. *JAISE 1*, 1 (2009), 5–14. (Cited in page 1)

[5] ABASCAL, J., AIZPURUA, A., CEARRETA, I., GAMECHO, B., GARAY, N., AND MIÑÓN, R. Some issues regarding the design of adaptive interface generation systems. In *Universal Access in Human-Computer Interaction. Design for All and eInclusion - 6th International Conference, UAHCI 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings, Part I* (2011), C. Stephanidis, Ed., vol. 6765 of *Lecture Notes in Computer Science*, Springer, pp. 307–316. (Cited in pages 5 and 97)

[6] ABASCAL, J., AIZPURUA, A., CEARRETA, I., GAMECHO, B., GARAY-VITORIA, N., AND MIÑÓN, R. Automatically generating tailored accessible user interfaces for ubiquitous services. In *The 13th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS '11, Dundee, Scotland, UK, October 24-26, 2011* (2011), K. F. McCoy and Y. Yesilada, Eds., ACM, pp. 187–194. (Cited in pages 5, 6, 11, and 96)

[7] AIZPURUA, A., CEARRETA, I., GAMECHO, B., MIÑÓN, R., GARAY-VITORIA, N., GARDEAZABAL, L., AND ABASCAL, J. Extending in-home user and context models to provide ubiquitous adaptive support outside the home. In *User Modeling and Adaptation for Daily Routines*, E. Martín, P. A. Haya, and R. M. Carro, Eds., Human–Computer Interaction Series. Springer London, 2013, pp. 25–59. (Cited in pages 5 and 96)

[8] AVCI, A., BOSCH, S., MARIN-PERIANU, M., MARIN-PERIANU, R., AND HAVINGA, P. J. M. Activity recognition using inertial sensing for health-

care, wellbeing and sports applications: A survey. In *ARCS Workshops* (2010), VDE Verlag, pp. 167–176. (Cited in page 18)

[9] BANAVAR, G., BECK, J., GLUZBERG, E., MUNSON, J., SUSSMAN, J., AND ZUKOWSKI, D. Challenges: An application model for pervasive computing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking* (New York, NY, USA, 2000), MobiCom '00, ACM, pp. 266–274. (Cited in page 14)

[10] BANGOR, A. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies 4*, 3 (2009), 114–123. (Cited in page 69)

[11] BANOS, O., GARCIA, R., HOLGADO-TERRIZA, J., DAMAS, M., POMARES, H., ROJAS, I., SAEZ, A., AND VILLALONGA, C. mhealthdroid: A novel framework for agile development of mobile health applications. In *Ambient Assisted Living and Daily Activities*, L. Pecchia, L. Chen, C. Nugent, and J. Bravo, Eds., vol. 8868 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 91–98. (Cited in pages 18 and 24)

[12] BØ, K., BO, K., BERGHMANS, B., AND MORKVED, S. *Evidence-based Physical Therapy for the Pelvic Floor: Bridging Science and Clinical Practice*. Elsevier Health Sciences, 2007. (Cited in page 79)

[13] BROOKE, J. SUS: A quick and dirty usability scale. In *Usability evaluation in industry*, P. W. Jordan, B. Weerdmeester, A. Thomas, and I. L. Mclelland, Eds. Taylor and Francis, London, 1996. (Cited in pages 69 and 87)

[14] BROWN, P. J., BOVEY, J. D., AND CHEN, X. Context-aware applications: from the laboratory to the marketplace. *Personal Communications, IEEE 4*, 5 (1997), 58–64. (Cited in page 16)

[15] CÂNOVAS, M., SILVA, H., LOURENÇO, A., CANENTO, F., AND FRED, A. MobileBIT: A framework for mobile interaction recording and display. In *Proc. of the 6th Conference on Health Informatics (HEALTHINF)* (2013), pp. 366–369. (Cited in pages 42, 65, and 95)

[16] CHANDRA, H., OAKLEY, I., AND SILVA, H. User needs in the performance of prescribed home exercise therapy. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2012), CHI EA '12, ACM, pp. 2369–2374. (Cited in page 79)

[17] CHON, J., AND CHA, H. Lifemap: A smartphone-based context provider for location-based services. *IEEE Pervasive Computing 10*, 2 (2011), 58–67. (Cited in pages 4 and 17)

[18] COOK, D. J., AUGUSTO, J. C., AND JAKKULA, V. R. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing 5*, 4 (2009), 277 – 298. (Cited in pages 2 and 15)

[19] COSTA, D., AND DUARTE, C. From one to many users and contexts: A classifier for hand and arm gestures. In *Proceedings of the 20th International Conference on Intelligent User Interfaces, IUI 2015, Atlanta, GA, USA, March*

29 - April 01, 2015 (2015), O. Brdiczka, P. Chau, G. Carenini, S. Pan, and P. O. Kristensson, Eds., ACM, pp. 115–120. (Cited in page 18)

[20] COULTER, C. L., SCARVELL, J. M., NEEMAN, T. M., AND SMITH, P. N. Physiotherapist-directed rehabilitation exercises in the outpatient or home setting improve strength, gait speed and cadence after elective total hip replacement: a systematic review. *Journal of Physiotherapy 59*, 4 (2013), 219 – 226. (Cited in page 79)

[21] COUTAZ, J., CROWLEY, J. L., DOBSON, S., AND GARLAN, D. Context is key. *Commun. ACM 48*, 3 (Mar. 2005), 49–53. (Cited in pages 2 and 15)

[22] DAVIES, N., AND GELLERSEN, H.-W. Beyond prototypes: challenges in deploying ubiquitous systems. *Pervasive Computing, IEEE 1*, 1 (Jan 2002), 26–35. (Cited in page 14)

[23] DEY, A. K. *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology, 2000. (Cited in page 3)

[24] DEY, A. K. Understanding and using context. *Personal and Ubiquitous Computing 5*, 1 (2001), 4–7. (Cited in pages 16 and 68)

[25] DEY, A. K., ABOWD, G. D., AND SALBER, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction 16*, 2-4 (2001), 97–166. (Cited in pages 2 and 22)

[26] DIX, A., FINLAY, J., ABOWD, G., AND BEALE, R. Human-computer interaction. *England: Pearson Education Limited* (2004). (Cited in page 67)

[27] DOURISH, P. What we talk about when we talk about context. *Personal and Ubiquitous Computing 8*, 1 (2004), 19–30. (Cited in page 1)

[28] FERREIRA, D. Aware: A mobile context instrumentation middleware to collaboratively understand human behavior. In *Ph.D. dissertation, University of Oulu, Faculty of Technology.* (2013). (Cited in pages 3 and 23)

[29] FETTE, I., AND MELNIKOV, A. The websocket protocol. (Cited in page 52)

[30] FONTECHA, J., NAVARRO, F. J., HERVÁS, R., AND BRAVO, J. Elderly frailty detection by using accelerometer-enabled smartphones and clinical information records. *Personal and Ubiquitous Computing 17*, 6 (2013), 1073–1083. (Cited in page 2)

[31] GAMECHO, B., ABASCAL, J., AND GARDEAZABAL, L. Testing a standard interoperability framework in an ambient assisted living scenario. In *of the 5th International Conference on Ubiquitous Computing and Ambient Intelligence, UCAMI 2011, Cancún, México, December 2-5, 2011, Proceedings* (2011). (Cited in pages 6 and 97)

[32] GAMECHO, B., GARDEAZABAL, L., AND ABASCAL, J. Augmented interaction with mobile devices to enhance the accessibility of ubiquitous

services. In *Mobile Accessibility (MOBACC 2013). Workshop at CHI 2013* (2013). (Cited in pages 8, 11, 66, and 96)

[33] GAMECHO, B., GARDEAZABAL, L., AND ABASCAL, J. Combination and abstraction of sensors for mobile context-awareness. In *The 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '13, Zurich, Switzerland, September 8-12, 2013 - Adjunct Publication* (2013), F. Mattern, S. Santini, J. F. Canny, M. Langheinrich, and J. Rekimoto, Eds., ACM, pp. 1417–1422. (Cited in pages 9, 11, 65, and 96)

[34] GAMECHO, B., GARDEAZABAL, L., AND ABASCAL, J. A context server to allow peripheral interaction. In *Peripheral Interaction: Embedding HCI in Everyday Life. Workshop at INTERACT 2013 the 14th IFIP TC13 Conference on Human-Computer Interaction.* (2013). (Cited in pages 11 and 96)

[35] GAMECHO, B., GUERREIRO, J., ALVES, P., LOURENÇO, A., DA SILVA, H. P., GARDEAZABAL, L., ABASCAL, J., AND FRED, A. Evaluation of a context-aware application for mobile robot control mediated by physiological data: The tobitas case study. In *R. Hervás et al. (Eds.): UCAmI 2014* (2014), vol. LNCS 8867, pp. 147–154. (Cited in pages 69, 74, and 96)

[36] GAMECHO, B., MIÑÓN, R., AND ABASCAL, J. Design issues on accessible user interface generation for ubiquitous services through egoki. In *12 European AAATE Conference* (2013), IOS Press, pp. 1304–1309. (Cited in pages 6, 11, and 96)

[37] GAMECHO, B., MINON, R., AIZPURUA, A., CEARRETA, I., ARRUE, M., GARAY-VITORIA, N., AND ABASCAL, J. Automatic generation of tailored accessible user interfaces for ubiquitous services. *Human-Machine Systems, IEEE Transactions on PP*, 99 (2015), 1–12. (Cited in pages 6, 11, and 96)

[38] GAMECHO, B., SILVA, H., GUERREIRO, J., GARDEAZABAL, L., AND ABASCAL, J. A context-aware application to increase elderly users compliance of physical rehabilitation exercises at home via animatronic biofeedback. *Submited to Journal of Medical Systems Under revision* (2015). (Cited in pages 69, 79, and 96)

[39] GHIANI, G., MANCA, M., PATERNÒ, F., AND PORTA, C. Beyond responsive design: Context-dependent multimodal augmentation of web applications. In *Mobile Web Information Systems*, I. Awan, M. Younas, X. Franch, and C. Quer, Eds., vol. 8640 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 71–85. (Cited in page 25)

[40] GIGGINS, O. M., PERSSON, U., AND CAULFIELD, B. Biofeedback in rehabilitation. *J Neuroeng Rehabil 10*, 60 (2013), 0003–10. (Cited in page 79)

[41] GOK, N., AND KHANNA, N. *Building Hybrid Android Apps with Java and JavaScript: Applying Native Device APIs*. " O'Reilly Media, Inc.", 2013. (Cited in pages 43 and 51)

[42] GOLDSTEIN, E. *Sensation and perception*. Cengage Learning, 2013. (Cited in page 14)

[43] HAAG, A., GORONZY, S., SCHAICH, P., AND WILLIAMS, J. Emotion recognition using bio-sensors: First steps towards an automatic system. In *Proc. ADS* (2004), Springer, pp. 36–48. (Cited in pages 4 and 18)

[44] HERVÁS, R., BRAVO, J., AND FONTECHA, J. An assistive navigation system based on augmented reality and context awareness for people with mild cognitive impairments. *IEEE J. Biomedical and Health Informatics 18*, 1 (2014), 368–374. (Cited in page 2)

[45] JANG, E.-H., PARK, B.-J., KIM, S.-H., CHUNG, M., PARK, M.-S., AND SOHN, J.-H. Classification of three negative emotions based on physiological signals. In *Proc. INTELLI 2013* (2013), pp. 75–78. (Cited in pages 4 and 18)

[46] KALANTAR-ZADEH, K. *Sensors, An Introductory Course*. Springer US, 2013. (Cited in pages 18 and 19)

[47] KOH, C. E., YOUNG, C. J., YOUNG, J. M., AND SOLOMON, M. J. Systematic review of randomized controlled trials of the effectiveness of biofeedback for pelvic floor dysfunction. *British Journal of Surgery 95*, 9 (2008), 1079–1087. (Cited in page 79)

[48] KORPIPÄÄ, P., MALM, E.-J., SALMINEN, I., RANTAKOKKO, T., KYLLÖNEN, V., AND KÄNSÄLÄ, I. Context management for end user development of context-aware applications. In *Proceedings of the 6th International Conference on Mobile Data Management* (New York, NY, USA, 2005), MDM '05, ACM, pp. 304–308. (Cited in page 17)

[49] KORPIPÄÄ, P., MÄNTYJÄRVI, J., KELA, J., KERÄNEN, H., AND MALM, E.-J. Managing context information in mobile devices. *IEEE Pervasive Computing 2*, 3 (2003), 42–51. (Cited in page 17)

[50] KUKKONEN, J., LAGERSPETZ, E., NURMI, P., AND ANDERSSON, M. Betelgeuse: A platform for gathering and processing situational data. *IEEE Pervasive Computing 8*, 2 (Apr. 2009), 49–56. (Cited in page 23)

[51] LAZAR, J., FENG, J. H., AND HOCHHEISER, H. *Research methods in human-computer interaction*. John Wiley & Sons, 2010. (Cited in page 67)

[52] LOURENÇO, A., SILVA, H., AND FRED, A. Unveiling the biometric potential of finger-based ecg signals. *Intell. Neuroscience 2011* (Jan. 2011), 5:1–5:8. (Cited in page 58)

[53] MIÑÓN, R., ABASCAL, J., AIZPURUA, A., CEARRETA, I., GAMECHO, B., AND GARAY, N. Generación de interfaces de usuario accesibles para entornos ubicuos, basadas en modelos. Interacción 2011 (XII Congreso Internacional de Interacción Persona-Ordenador). Actas del Congreso, Ibergarceta S. L., pp. 145-154. ISBN: 987-84-9281-234-9. Lisboa (Portugal). (Cited in pages 5 and 97)

[54] MIÑÓN, R., ABASCAL, J., AIZPURUA, A., CEARRETA, I., GAMECHO, B., AND GARAY, N. Model-based accessible user interface generation in ubiquitous environments. In *Human-Computer Interaction - INTERACT 2011*

- 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part IV (2011), P. Campos, T. C. N. Graham, J. A. Jorge, N. J. Nunes, P. A. Palanque, and M. Winckler, Eds., vol. 6949 of *Lecture Notes in Computer Science*, Springer, pp. 572–575. (Cited in pages 5, 6, 11, and 97)

[55] MIÑÓN, R., MORENO, L., AND ABASCAL, J. A graphical tool to create user interface models for ubiquitous interaction satisfying accessibility requirements. *Universal Access in the Information Society 12*, 4 (2013), 427–439. (Cited in page 6)

[56] MORRIS, D., SCOTT, T., AND TAN, D. S. Emerging input technologies for always-available mobile interaction. *Foundations and Trends in HCI 4*, 4 (2011), 245–316. (Cited in page 18)

[57] PARKKA, J., ERMES, M., KORPIPAA, P., MANTYJARVI, J., PELTOLA, J., AND KORHONEN, I. Activity classification using realistic data from wearable sensors. *Information Technology in Biomedicine, IEEE Transactions on 10*, 1 (Jan 2006), 119–128. (Cited in page 4)

[58] PASPALLIS, N., AND PAPADOPOULOS, G. A. A pluggable middleware architecture for developing context-aware mobile applications. *Personal and Ubiquitous Computing 18*, 5 (2014), 1099–1116. (Cited in page 24)

[59] PETROFSKY, J. The use of electromyogram biofeedback to reduce trendelenburg gait. *European Journal of Applied Physiology 85*, 5 (2001), 491–495. (Cited in page 79)

[60] RAO, S. S. Biofeedback therapy for constipation in adults. *Best Practice & Research Clinical Gastroenterology 25*, 1 (2011), 159 – 166. Chronic Constipation. (Cited in page 79)

[61] REDDY, S., MUN, M., BURKE, J., ESTRIN, D., HANSEN, M., AND SRIVASTAVA, M. Using mobile phones to determine transportation modes. *ACM Trans. Sen. Netw. 6*, 2 (Mar. 2010), 13:1–13:27. (Cited in page 4)

[62] RIBEIRO, A. F., LOPES, G., PEREIRA, N., CRUZ, J., AND COSTA, M. F. Bot'n roll robotic kit as a learning tool for youngsters. In *9th International Conference on Hands on Science (HSCI'2012)* (2012), Universidade do Minho, pp. 192–192. (Cited in page 70)

[63] RODRIGUEZ, I., ASTIGARRAGA, A., JAUREGI, E., RUIZ, T., AND LAZKANO, E. Humanizing nao robot teleoperation using ros. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on* (Nov 2014), pp. 179–186. (Cited in page 89)

[64] RYAN, N. S., PASCOE, J., AND MORSE, D. R. Enhanced reality fieldwork: the context-aware archaeological assistant. In *Computer applications in archaeology* (1998), Tempus Reparatum. (Cited in page 16)

[65] SAHA, D., AND MUKHERJEE, A. Pervasive computing: A paradigm for the 21st century. *Computer 36*, 3 (Mar. 2003), 25–31. (Cited in pages 2 and 14)

[66] SATYANARAYANAN, M. Pervasive computing: vision and challenges. *IEEE Personal Commun. 8*, 4 (2001), 10–17. (Cited in pages 1, 2, 13, and 14)

[67] SCHILIT, B., ADAMS, N., AND WANT, R. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on* (1994), IEEE, pp. 85–90. (Cited in page 16)

[68] SCHMIDT, A. Implicit human computer interaction through context. *Personal Technologies 4*, 2-3 (2000), 191–199. (Cited in page 14)

[69] SCHMIDT, A. *Ubiquitous computing-computing in context*. PhD thesis, Lancaster University, 2003. (Cited in pages 22 and 68)

[70] SCHMIDT, A., AIDOO, K. A., TAKALUOMA, A., TUOMELA, U., LAERHOVEN, K. V., AND VELDE, W. V. D. Advanced interaction in context. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing* (London, UK, UK, 1999), HUC '99, Springer-Verlag, pp. 89–101. (Cited in pages 2, 4, and 16)

[71] SCHWARTZ, M. S., AND ANDRASIK. *Biofeedback: A practitioner's guide* . Guilford Press, 2005. (Cited in page 79)

[72] SILVA, H., FRED, A., AND MARTINS, R. Biosignals for everyone. *Pervasive Computing, IEEE 13*, 4 (Oct 2014), 64–71. (Cited in page 18)

[73] SILVA, H., GUERREIRO, J., LOURENÇO, A., FRED, A., AND MARTINS, R. Bitalino: A novel hardware framework for physiological computing. In *International Conf. Physiological Computing Systems - PhyCS* (January 2014), pp. 246–253. (Cited in page 70)

[74] SWAN, M. Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0. *Journal of Sensor and Actuator Networks 1*, 3 (2012), 217–253. (Cited in page 2)

[75] VILLARREAL, V., FONTECHA, J., HERVÁS, R., AND BRAVO, J. Mobile and ubiquitous architecture for the medical control of chronic diseases through the use of intelligent devices: Using the architecture for patients with diabetes. *Future Generation Comp. Syst. 34* (2014), 161–175. (Cited in page 18)

[76] WANT, R., HOPPER, A., FALCAO, V., AND GIBBONS, J. The active badge location system. *ACM Transactions on Information Systems (TOIS) 10*, 1 (1992), 91–102. (Cited in page 15)

[77] WEISER, M. The computer for the 21st century. *Scientific American 265*, 3 (January 1991), 66–75. (Cited in pages 1 and 16)

[78] WEISER, M., AND BROWN, J. S. Beyond calculation. In *Beyond Calculation*, P. J. Denning and R. M. Metcalfe, Eds. Copernicus, New York, NY, USA, 1997, ch. The Coming Age of Calm Technolgy, pp. 75–85. (Cited in page 1)

[79] WIESE, J., SAPONAS, T. S., AND BRUSH, A. B. Phoneprioception: Enabling mobile phones to infer where they are kept. In *Proceedings of the*

SIGCHI Conference on Human Factors in Computing Systems (New York, NY, USA, 2013), CHI '13, ACM, pp. 2157–2166. (Cited in pages 2, 4, and 17)

[80] WINOGRAD, T. Architectures for context. *Human-Computer Interaction 16*, 2 (2001), 401–419. (Cited in pages 17 and 23)

[81] WOOD, S. E., WOOD, E. G., AND BOYD, D. *The World of Psychology, 7th Edition*. Pearson, 2011. (Cited in page 14)

[82] ZIMMERMANN, G., AND VANDERHEIDEN, G. C. The universal control hub: An open platform for remote user interfaces in the digital home. In *Proc. HCI* (2007), Springer, pp. 1040–1049. (Cited in page 6)

Appendix A

Glossary

BITalino Wireless sensor platform to use in physiological computing.

Bot'n Roll Mobile robotics platform used in the evaluation.

CSS Cascading Style Sheet is a style sheet language used for describing the look and formatting of a document written in a markup language.

DPL Data Processing Language is a description language build on JSON for describing the relationships between blocks in MobileBIT.

Egoki User interface generator for ubiquitous environments. It generates accessible and user tailored user interfaces.

EGONTO Knowledge base for Egoki.

Functional Block Software component of MobileBIT where the Igerri sensors are represented.

HTML Hyper Text Markup Language is the standard language to define the Webs.

Igerri Conceptual framework for sensor abstraction and the creation of virtual sensors for mobile Context-Aware computing.

JS JavaScript programming language is used to describe client-side scripts in a web browser.

JSI JavaScript Interface is a module of the MobileBIT that connects Web application layer with the native layer in the Hybrid application schema.

JSON Text format that facilitates structured data interchange between all programming languages.

MobileBIT Part of the implementation of the Igerri Framework dedicated to the development of sensor applications .

PervasiveBIT Part of the implementation of the Igerri Framework dedicated to network and context management.

RESapp Rehabilitation Exercise System application created to test the Framework.

SENSONTO Knowledge base for Igerri.

SensorHub A module for the discovery of sensors in the implementation of Igerri.

SUS System Usability Scale is a likert questionnaire to measure the usability.

ToBITas Context-Aware application to command Bot'n Roll via BITalino.

UCH Universal Control Hub is the reference implementation for the URC Standard.

UIML User Interface Modelling Language is a User Interface Description Language based on XML.

URC Universal Remote Console is a ISO standard to address the accessibility and heterogeneity across services and devices.

WFM WorkFlow Manager is a part of MobileBIT dedicated to the creation of the Functional Block newtork.

XML Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable.