PhD-FSTC-2016-63
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defense held on 06/12/2016 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN *INFORMATIQUE*

by

Remus Alexandru DOBRICAN
Born on 13 November 1988 in Timişoara, (Romania)

COLLABORATIVE RULE-BASED PROACTIVE SYSTEMS: MODEL, INFORMATION SHARING STRATEGY AND CASE STUDIES

Dissertation defense committee

Dr Denis ZAMPUNIERIS, dissertation supervisor
Professor, Université du Luxembourg

Dr Michael Ignaz SCHUMACHER
Professor, University of Applied Sciences Western Switzerland

Dr Pascal BOUVRY, Chairman
Professor, Université du Luxembourg

Dr Steffen ROTHKUGEL
Professor, Université du Luxembourg

Dr Anthony CLEVE, Vice Chairman
Professor, University of Namur

Abstract

The Proactive Computing paradigm provides us with a new way to make the multitude of computing systems, devices and sensors spread through our modern environment, work for/pro the human beings and be active on our behalf. In this paradigm, users are put on top of the interactive loop and the underlying IT systems are automated for performing even the most complex tasks in a more autonomous way.

This dissertation focuses on providing further means, at both theoretical and applied levels, to design and implement Proactive Systems. It is shown how smart mobile, wearable and/or server applications can be developed with the proposed Rule-Based Middleware Model for computing pro-actively and for operating on multiple platforms.

In order to represent and to reason about the information that the proactive system needs to know about its environment where it performs its computations, a new technique called Proactive Scenario is proposed. As an extension of its scope and properties, and for achieving global reasoning over inter-connected proactive systems, a new collaborative technique called Global Proactive Scenario is then proposed.

Furthermore, to show their potential, three real world case studies of (collaborative) proactive systems have been explored for validating the proposed development methodology and its related technological framework in various domains like e-Learning, e-Business and e-Health.

Results from these experiments confirm that software applications designed along the lines of the proposed rule-based proactive system model together with the concepts of local and global proactive scenarios, are capable of actively searching for the information they need, of automating tasks and procedures that do not require the user's input, of detecting various changes in their context and of taking measures to adapt to it for addressing the needs of the people which use these systems, and of performing collaboration and global reasoning over multiple proactive engines spread across different networks.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Professor Dr. Denis Zampunieris, for his support, for his patience, for his professionalism and for giving me the opportunity to do research inside a very nice team. Without his help, this PhD would have been a much more difficult road to take. I appreciate a lot the liberty which was offered to me to do research by exploring many different directions that I was not aware of.

I would also like to thank the members of my CET committee for their advices, guidance and encouragements since the beginning of this work.

Moreover, I would like to thank Dr. Daniel Theisen, head of the research unit "Sports & Health at the Public Research Centre for Health" (CRP-Santé) in Luxembourg, and to his team, and to Dr. Patrick Feiereisen, part of the physiotherapy group at the CHL (Centre Hospitalier de Luxembourg) for helping me with the e-Health application.

A special thanks goes to my parents, Alexandru and Elena Dobrican, and to my entire family, which offered me moral support during the whole time of my PhD. Also, I would like to thank Victoria for her continuous effort to support me and her permanent smile, which brought joy and inspiration.

Also, a special thanks to my colleagues, Gilles Neyens and Sandro Reis, for their help, especially with the technical side, i.e., with the development of the server-side and mobile applications.

Finally, I would like to thank God.

Abbreviations

AI Artificial intelligence. 13, 16, 19, 22, 23, 25, 28

API Application Programming Interface. 32, 41

AT Anaerobic Threshold. 104, 105

BC Backwards Chaining. 24, 60

BPM Beats Per Minute. 101

CE Communication Engine. 37, 41

CoPs Communities of Practice. 65

CQ Current Queue. 36, 61

CR Cardiac Rehabilitation. 92–97, 99, 100, 102–104, 106–108

ECA Event Condition Action. 17, 18

ECG Electrocardiogram. 94–96

ES Expert System. 25–27

ET Exercise Training. 92, 97

FC Forward Chaining. 24, 25, 60

FIFO First-in, First-Out. 36

GCM Google Cloud Messaging. xiii, 43, 44, 46, 50

GPaS Global Proactive Scenario. 9, 10, 54–57, 59, 64, 78, 81, 83–85, 87, 89–92, 100, 111

GPS Global Positioning System. 51, 83

GUI Graphical User Interface. xii, xiv, 34, 37, 58, 72, 81, 82, 97, 98

HR Heart Rate. xiv, 92, 94–99, 101, 102, 104, 105, 107

IT Information Technology. 65

Jess Java Expert System Shell. 26, 27, 62

JSON JavaScript Object Notation. xiii, xiv, 41, 42, 44, 45, 55, 57, 88, 89

KE Knowledge Engineering. 23

KR Knowledge Representation. 13–15, 18, 19, 25, 26

LMS Learning Management System. xi, xiv, 9, 65–69, 72–77

LPaS Local Proactive Scenario. 9, 10, 54, 55, 59, 64, 65, 68–71, 76, 77, 92, 100

NM Notification Manager. 37, 41

NQ Next Queue. 36, 61

O-O Object-Oriented. 14, 15, 18, 22, 27, 31, 32, 34, 40

OCoP Online Community of Practice. 66, 68, 70, 71, 73

OCoPs Online Communities of Practice. i, 65, 66, 68, 72–77

ORMLite Object Relational Mapping Lite. 42

OS Operating System. 29, 32, 33, 37, 39–41, 81

P2P Peer-to-peer. 62

PaC Proactive Computing. 2, 4, 5, 7, 8, 32, 64–66, 68, 73, 77, 79, 81, 91, 107

PaS Proactive Scenario. 7–9, 30, 36–38, 41, 46, 49, 51, 53–55, 58–61, 64, 65, 70, 88, 92, 100–102, 107

PC Personal Computer. 40

PE Proactive Engine. 29, 30, 32–42, 45–47, 49–56, 60, 61, 68, 83, 85–88, 99, 101, 104, 106, 111

PPG Photoplethysmogram. 94, 97, 101, 107

PR Proactive Rule. 33–39, 45–47, 49–51, 53–61, 70, 85, 86, 91, 101, 103

PS Proactive System. 5–7, 9, 30, 32, 34, 36–41, 43, 55–61, 64, 68–70, 92, 104, 110

QM Queue Manager. 32, 34, 36, 41

RBES Rule-based Expert System. 27

RBPS Rule-based Proactive System. 7, 9, 43–46, 58, 64, 111

RBS Rule-based System. 5, 23–25, 31, 36, 42, 60, 110

RE Rule Engine. 32–34, 36, 41, 46, 47, 49

SMS Short Message Service. 96

SNS Social Networking System. xi, 67

SQL Structured Query Language. 17, 24, 42

XML Extensible Markup Language. 16, 18, 27, 43

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A couple of decades ago, in 1988, Mark Weiser introduced the idea of ***ubiquitous computing***, which, at that time, was considered a revolutionary paradigm [6]. The essence of this paradigm consisted of the vision of having an environment where old and new networking and computing technologies will complement each other and will work together. Nowadays, this dream has become a fact in certain computing environments. Embedded devices capable of making advanced computation in everyday objects are everywhere. These devices spread through various places, e.g., at home, at school, at work and even when we are on the move, becoming ***pervasive***. As smart devices, computing systems and software systems become a bigger part of our daily life and of our daily activities, it becomes highly significant for these systems to perform and to provide stable, continuous and complex services.

Shortly after, in 2000, David Tennenhouse proposed a new paradigm, ***Proactive Computing (PaC)***, which gave computer scientists the opportunity to tackle a new research direction [7]. More precisely, this would lead researchers beyond *ubiquitous computing* and *pervasive computing*, towards a world full of networked devices capable of anticipating human users' needs and of acting on their behalf. This newly proposed concept would extend and augment the realm of ubiquitous devices by giving them a purpose and by providing a modern way of controlling all the devices around us.

Current computing challenges require advanced software systems capable of addressing the full needs of the users and of tackling new problems in an efficient and flexible manner. And, by referring to advanced software systems, it is meant to include those systems that satisfy multiple key properties. This work narrows down the scope of how to design such systems by focusing on certain key properties such as ***context-awareness***, ***adaptiveness*** and ***collaboration*** as a consequence of using *proactive computing*. Furthermore, as each of these properties constitute separate broad research fields of their own, it is aimed to show how better outcomes are achieved by

linking them together rather than by considering them as separated properties.

1.1 Motivation

Technology is evolving at an incredible speed and its impact on our society is happening faster than it takes us to adapt to it. Recent discoveries such as growing atomically thin circuits and transistors [8] are aiming at keeping pace with Moore's Law [9], which stated back in 1965 that the number of transistors in integrated circuits will double almost every two years. Sensors and actuators continue the trend of getting smaller and smaller, more powerful and cheaper than their previous generations. Small embedded computers offer everyday objects, such as clothes, cans, mugs, mobile devices or wearable devices, new computing capabilities. Mobile phones, with a market of 8,2 billion devices in 2016 [10], and wearables, with an estimated number of 110 million shipped devices in 2016 [11], are undoubtedly a major part of a growing computing environment that not only includes other physical objects, like smart screens, but also virtual objects.

Not only that computing objects increase their numbers but they augment their connectivity, thus creating an **increasingly networked environment**. Some estimations indicated that, by 2020, an incredible number of almost 50 billion devices will be connected to the Internet [12], with an average of 6.5 connected devices per person. Thanks to the latest advances in communication technologies such as wireless communication networks or wireless sensor networks, enable objects that were normally considered as passive physical objects with computing and communicating capabilities. Short-range wireless communication technologies used intensively by smartphones and wearable devices such as Bluetooth, ZigBee or IEEE 802.15.6 offer low-energy communication for low data rate applications. Wireless hotspots are so widely spread that people have access to the Internet almost everywhere they are, e.g., in airplanes, in buses, in trains. Thus, information is available from multiple sources and knowledge about the surrounding environment is more accurate. This context creates great collaboration opportunities for all the interconnected devices and their applications, and this can lead to solving more complex problems which could not be addressed until now by single computing software systems [13].

However, as almost everything comes at a cost, creating collaboration mechanisms for applications inside these networks of smart devices is not an easy task due to the **increasing complexity** of exchanging and handling data between computing systems which come with different operating systems, different communication protocols and different computing capabilities. The key of addressing high complexity for collaborative systems is to **automate** procedures, processes and tasks to reduce human interven-

tion only to the most important decision, keeping, nevertheless, humans in a supervising role. This involves enhancing devices and their applications with decision-making capabilities, also referred to as delegation of control. Nowadays, examples where human judgement and control has been passed to computing systems are present even in important industry sectors like aviation or healthcare.

Society's desire to be better informed and to make better decisions is well-known since a long time ago. For example, people that are better informed about traffic conditions can take alternative routes in order to avoid being stuck for hours or people knowing important information about public transportation could reduce significantly their travel time. In time of crisis, when natural disasters or industrial accidents occur, it is crucial to have accurate information from multiple trusted sources to be able to take proper actions. If various devices and sensors are coordinated to work for humans in such cases the benefits would be enormous, ending up with saved time, money and even human lives.

Existing tools that are taking advantage of the current settings are often inadequate. They are either too *user-driven*, meaning the full control is given to the users, or too *reactive*, waiting for input, either from the user or from an internal event, that will trigger some kind of action. Using software tools that are supposed to be *user-friendly* has become difficult to use due to the variety of user interfaces and numerous functions they offer. Users are often overloaded with information and this might prevent them to take the right action at the right time. In order to use them properly, users need to become highly specialised by doing special training lessons to learn how to handle such tools.

On the other hand, in rapidly changing ubiquitous environments, devices and their applications, in order to achieve their goals, need to be capable of noticing all the changes and of taking rapid measures in response to what is happening. This requires the applications to be constantly monitoring the environment and not just to wait for the user or for another process to activate them. They should actively look for the information they need, for the firing of a particular event and, even, for the lack of an event. Even more, they need to anticipate and predict certain situations to better provide solutions to the people that use them. Gathering data only from local sensors can lead to information that is not complete or to having partial knowledge about a certain situation. By collaborating with other applications, the knowledge becomes more complete and the information easier to be checked. These minimum set of requirements need a new generation of innovative software systems that solve more complex computational problems. In order to reach their full potential, applications for smart devices need to be remodelled.

One solution, in this case, can be offered by *PaC*, which can accurately represent our needs and preferences. By using this particular computing

mode, software systems can be delegated to operate in our behalf, they can be automated for performing even the most complex tasks and they can collaborate and exchange information to offer the best solutions in foreseen situations.

1.2 Objectives of the thesis

Conventional hard-coding methods for building modern applications tend to become extremely complicated even for the most skilled programmers. Rule-based Systems (RBSs) are an alternative for provide methods for expressing and reasoning about knowledge in a clear manner. However, this approach presents several challenges and issues that need to be tackled. Besides this, the importance of being able to use PaC needs to be addressed. This raises the first research question (RQ1):

RQ1. *What are the advantages of using rule-based reasoning when building modern software applications and how will this approach be able to employ Proactive Computing on different platforms?*

Following the initial direction of developing a RBS for enhancing a software system with proactive capabilities [14] and the availability of a multitude of devices capable of conducting complex computations, presented in the previous subsection, the second main objective of the thesis is to design a rule-based model, capable of computing proactively, that could be used for building proactive applications on various platforms and devices. The purpose was not to create a universal model capable of operating at the same time on different platforms but to develop an architecture that could be used by different developers for implementing a Proactive System (PS) on top of different operating systems.

Building such a system raises new questions in terms of its properties. These properties or characteristics are important to determine if this system is addressing most of the current concerns of ubiquitous applications. Among these concerns are the ability of determining different contexts, or being context-aware, the ability to anticipate multiple possible situations that can happen in the future, or the ability of being able to adapt to multiple platforms, e.g., such as distinct operating systems of various devices. Developing such a model is addressing the second research question (RQ2), which focuses on establishing a set of common core components, on the architecture of the model and on its properties:

RQ2. *What are the key components necessary to build a functional rule-based proactive system, what are their functions as part of the model, what is the relationship between them, how to build the model and what are the*

main properties of such a model?

Following the design guidelines to build multiple PSs will lead to a diversified environment, where multiple proactive applications will operate. Apart from opening new computational perspectives, the answers to RQ1 and RQ2 provide an additional opportunity and immediately fire a new question: what if these PSs would be able to exchange information between them? This leads to the third research question (RQ3), which states the following:

RQ3. *How should multiple operational rule-based proactive systems, dispersed over various networks, communicate and what kind of novel collaborative mechanism should they use for exchanging information?*

Establishing a communication protocol for PSs provides the possibility of connecting multiple environments rich in information, thus extending the perspectives of proactive applications. This brings many new challenges and points that also need to be addressed, such as determining at what level should the communication take part, how it should happen, e.g., in the background, without the user's involvement, or still asking the user's explicit permission in some essential steps, and what should actually be exchanged during the communication process. This third objective of the thesis is to find a method for a PS to take advantage of the diverse information available in ubiquitous environments that is normally accessible only by the native applications and restricted to outside software systems. The answer to RQ3 provides not only a technique for allowing collaborative proactive applications to be developed but will also take into account other common challenges in rule-based application design like *correctness*, *termination* and *response time* [15]. *Correctness* refers to the fact of producing the correct output for a chain of rules for all the correct inputs, *termination* to the idea of producing an output at the end of a chain of rules and *response time* to the amount of time a chain of rules needs to finish its execution or, in different words, to produce an output.

Developing basic rules for single software systems is a classical task for rule-based systems but developing sets of special rules for PSs, that activate on different platforms and are able to exchange information, is a different challenge. These special rules are describing more complex situations, which are quite hard to address with a couple of basic rules. This requires a certain model, according to which rules need to be conceived. This leads to the following research question:

RQ4. *How should various distributed sequences of events be expressed under the form of rules and sets of rules capable of computing dynamically complex situations?*

The idea is that breaking down a complex situation into multiple smaller, manageable fractions helps the developers to reason easier and to design the actions to be taken in each case. This leads to a logical, straightforward and natural way of representing each situation.

The solution to the next and last research question (RQ5) is addressing the last objective of this thesis, which is, to study the impact of collaboration in diverse networks where various proactive devices are connected and to validate the theoretical models created as solutions for the previous research questions. More precisely, the goal is to develop domain-specific applications that are using the Rule-based Proactive System (RBPS) and its collaboration technique to solve existing real-world issues, in multiple domains. RQ5 is formulated as follows:

RQ5. *How would rule-based collaborative PSs impact real-world actual situations and which domains would benefit from such an approach?*

In many ways, this last research question is the most compelling of the research questions discussed in this thesis as it shows, by using real examples, the potential of PSs. It also emphasizes, giving concrete examples, in which domains would PSs be a suitable solution to solve existing issues and to address current challenges.

The research questions that are addressed in this thesis contribute to a better understanding of the concept of PaC and of the dynamics of an environment where multiple PSs operate and collaborate. They also indicate how this technology can be beneficial for both, the users and the developers, at the same time.

1.3 Methodology

The research methodology conducted in this dissertation is following the lines of constructive research [16]. Initially, based on previous theoretical studies, analysed and discussed in chapter 2, key practical challenges and relevant problems are identified. This first part is designed to shed some light on both explicit and implicit reasons for considering certain aspects as challenges. Then, a solution is designed to address these issues. More specifically, the core components of a rule-based model are identified and guidelines for building a functional rule-based proactive engine are provided. Following these guidelines a new middleware architecture for mobile devices is implemented, a new principle for converting existing situations into Proactive Scenarios (PaSs) and a collaboration protocol between PSs are proposed. This part addresses the first four research questions. The proposed model is then tested with standard quantitative and qualitative approaches to show how it behaves in certain conditions and to establish

the performance boundaries under which it can easily operate without having any issues. And finally, to demonstrate the solution's feasibility and usefulness, several real examples and prototype applications are developed and tested. They are conceived and implemented in an iterative manner, which starts with a basic example that is benefiting from the proposed solution and finish with a complex system, in the last case study, where all the parts are working together. Adopting multiple applications addresses and answers the last research question.

1.4 Contributions

The main aim of this dissertation is to contribute to the support of both the end users, which are using advanced software applications to solve part of their daily tasks, and the developers, which are building these applications, by employing PaC. This dissertation brings both a scientific and a practical contribution. Through the rest of this section the most important contributions are shortly described.

The first main contribution was to bring arguments in favour of using rule-based system when dealing with software systems which involve many decision-making procedures.

The second contribution was to design and conceive a generic rule-based model that could compute proactively in an object-oriented environment and to implement it on top of multiple platforms, e.g., as a middleware architecture for Android-based mobile devices and for Linux-based servers. This second step includes a prior phase dedicated for identifying the main core components of such a generic proactive system.

The third important contribution was to plan and implement a communication mechanism for the middleware model to be able to exchange information with similar models functioning on different devices.

Another key contribution is the method of decomposing complex tasks and situations, part of the overall set of situations that a developer needs to address when creating a specific application, into multiple collections of rules called **PaS**. This decomposition process allows the developer to be more structured and to have a more logical format for analysing facts, for taking a decision in each case and for taking appropriated actions. Furthermore, a collaboration technique for proactive systems is proposed, which gives the possibility to perform computations in a coordinated and distributed manner.

Three cases studies, together with their prototype applications, are proposed to test, evaluate and validate the proposed theoretical solutions. Each case study can also be seen as a separate contribution as they also addresses various objectives and challenges in domains like e-Learning, e-Business management and e-Health or, more precisely, telemedicine.

1.5 Dissertation outline

This dissertation is composed of two main parts besides the introductory chapter, which contains a brief introduction, the motivation for this work, the research questions, the methodology used and the main contributions, and the last chapter, where the main conclusions are drawn. Each part consists of several chapters, closely related. **Part I** starts by offering background information relevant to this dissertation. It represents the theoretical foundation on which this work relies. It focuses on the fundamental concepts behind the design and development of RBPSs and it is comprised of the following chapters:

Chapter 2 contains the relevant background information necessary to understand the main concepts in this dissertation. Key challenges and missing links between existing problems are identified based on previous research studies conducted in the related research domains.

Chapter 3 is dedicated to presenting a middleware model that would address issues and challenges previously identified in chapter 2. The model comes as a solution for static and passive systems, which need to become proactive, a change of paradigm which was presented in [17] and awarded with a best paper award [18]. The core components and the architecture of the model are discussed, and, the properties of software systems implementing the proposed model are outlined. Advantages and the downsides of the proposed architecture are also discussed in this chapter and prior linked research approaches are analysed.

Chapter 4 introduces a structured technique for breaking down and organising the situations and the goals of a software application, i.e., ***PaSs***. Then, a collaborative technique, called Global Proactive Scenario (GPaS), with which RBPSs, based on the model proposed in chapter 3, are extending their local operations. Furthermore, in this chapter it is shown how this collaborative technique is a novel method for computing proactively in a distributed way. Additionally, other research studies that contain similar approaches to exchange information are examined.

Part II is focusing on the practical implementation of the solutions proposed in the first part of this dissertation. Starting from a more basic example of application of Proactive Scenarios, several case studies are given to exemplify the applicability of PSs in different domains. Each of the case study, presented in a separate chapter, has a corresponding paper published in a peer-reviewed conference. Part II is divided into the following chapters:

Chapter 5 introduces an application which uses the first type of PaSs, i.e., *Local Proactive Scenarios (LPaSs)*, to enhance a static e-Learning software platform with proactive functions. More specifically, a RBPS is designed to create, maintain and support social communities of students inside a LMS. This initial application is used next to a single system to transform this system into a flexible environment. To prove the added value and to

check to role played by the proactive system in increasing the students' online engagement and in facilitating the learning process a quantitative study and several observations are conducted.

In **chapter 6**, with the help of both *LPaSs* and *GPaSs*, a prototype mobile application is developed and discussed. Using group-driven collaboration and location-based collaboration, computed proactively, the mobile devices of several users are automatically switched into silent mode during a meeting and/or an important event. It is shown how both types of Proactive Scenarios run on top of the middleware model proposed in chapter 3.

In **chapter 7**, the implementation and evaluation of a complex e-Health proactive system which encompasses all the proposed theoretical aspects proposed in this dissertation is presented. Multiple aspects regarding the cardiac rehabilitation of patients are discussed, including multiple layers of feedback for patients provided by the Proactive System, and several applications for wearable devices, mobile devices and server side are presented.

And, finally, **chapter 8** summarises the contributions of this work and draws the main conclusions. In addition, several research perspectives are suggested.

Part I

Theoretical foundations

Chapter 2

Background information

In day to day life, happenings that have a meaning to somebody or in relation with someone are generally referred to as **events**. Events are used to describe certain things that happen in a particular situation. A student receiving his/her assignment in school, the birth of a child or receiving a postcard can be labelled as events because something has changed according to the previous moment in time. There has been a huge amount of studies in computer science over a long period of time on how to detect, analyse, describe, process and react to events. Three important aspects when speaking about events are *data*, *information* and *knowledge*. They were initially characterised as a chain [19], where each element would represent the transition from the previous element. Typically, the starting point is *data*, which is collected, classified, corrected and summarised. Then, after a transformation process, data becomes *information*. When context is added to the information, *knowledge* is obtained. Knowledge is the hardest of the three to handle as it normally applies to the minds of human beings, being related to their experience, expertise and judgement. It can be simply possessed by people as well as integrated and kept on a machine. *Wisdom*, a fourth element of the DIKW pyramid [20], derived from knowledge, is added on top of the first three elements. This pyramid represents one of the most popular models for representing the structural and functional relationships between data, information, knowledge and wisdom. Wisdom refers mainly to knowing what to do, what action to perform and why to do it. However, when dealing with computing systems, the difference between knowledge and wisdom is quite subtle, many times authors preferring to stick to the first three elements of the DIKW pyramid.

A basic example of the relationship between these elements, is illustrated in figure 2.1. The *data*, in the given example, represents a set of symbols or entities, i.e., names like X, Y, Z or course, assignment or student. *Information* is actually data processed to be useful, e.g., X is a name which belongs to a student. It also needs to provide answers to the questions

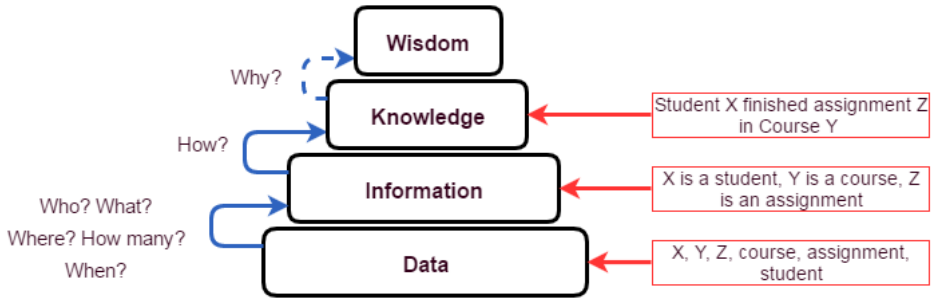


Figure 2.1: Data, Information, Knowledge, Wisdom - DIKW Pyramid.

"Who?", *"What?"*, *"Where?"*, *"When?"*. In the transition step between data and information, raw data is upgraded to information by giving it relevant meaning. *Knowledge* incorporates an extended mix of information that has been assigned to certain contexts, values, expert understanding and experiences. In the portrayed example, *knowing that student X finished assignment Z for course Y* represents knowledge. Two additional levels of abstraction exist on top of the basic form of the DIKW pyramid. They apply to systems where it is important to *understand* why the accumulated knowledge is justified, what does it mean and what to do with it or the ability of applying this knowledge.

When speaking about computing systems, knowledge has to be **represented** and **reasoned** about. These two main requirements are essential for any software system. They are strictly connected and would not work one without the other.

2.1 Knowledge Representation

Knowledge Representation (KR) is a multidisciplinary field which covers multiple disciplines such as *Artificial intelligence (AI)* in computer science or cognitive representation in psychology and philosophy. If knowledge is to be processed by software systems, it must contain an explicit and precise model of representation, including its semantics. As knowledge is one of the key aspects for building intelligent behaviour [21], it is imperative to have a mechanism that would represent it in a well-defined formal manner. To deduce and build knowledge, KR relies on various input events. The events are also used the starting point for deriving complex states and high-level systems.

There are several main types of knowledge, i.e., *declarative* and *procedural* knowledge as pointed out in [22, p. 61-63], or, *declarative*, *procedural* and *metaknowledge* knowledge as listed in [23, p. 182]. The difference between the two, in AI, for example, is better illustrated by the case of a robot that has to navigate inside a building [24]. If the robot's software system is

based on procedural knowledge, it will contain procedures or strategies to execute an action like *move to a certain floor*. On the other hand, if it based on declarative knowledge, the robot will know how to *move*, *turn* or *stop*. It will need an additional algorithm to help it achieve its goals. Later on, additional types of knowledge were mentioned by some authors, including *factual* and *meta-cognitive* knowledge [25, p. 46] or *tacit*, *implicit*, *a priori* and *a posteriori* knowledge [26]. Each type of knowledge has several representations. Choosing a way of representing knowledge is still a big challenge as each method has its own advantages like, for example, declarative representations, which offer standardisation, optimisation and improved methods of acquiring and reusing knowledge [27, p. 89].

2.1.1 Knowledge Representation Approaches

The term of KR approaches or technologies refers to the set of common representation tools like frames, semantic nets, rules, ontologies, logic, Object-Oriented (O-O) or/and conceptual graphs. A KR approach should support at least a couple of different kinds of activities. Selecting an appropriated KR approach is an important step as an inadequate selection could lead to problems later on, in the last stages of the system development.

Mylopoulos and Levesque classified knowledge representing techniques into four main schemas: *logical representation*, *procedural representation*, *networked representation* and *structural representation* [28]. The first category of schemas, which includes first order logic or Prolog, uses mathematical or orthographical symbols to express knowledge in formal logic. The second category, which includes rules of productions systems, employs multiple sets of instructions for solving domain-related problems. The third category, which includes semantic networks and conceptual graphs, employs graphs to represent knowledge. This category of schemas, however, is better suited to model static knowledge in a particular domain. And, the last category or the structural representation schemas, makes use of more complex data structures like scripts, frames and objects to represent knowledge.

Object-Oriented Representation

To handle a bigger number of sentences or procedures in a growing knowledge base a more efficient method of organising them is required. Marvin Minsky came up with the idea of dealing with O-O sets of procedures to handle new situations [29]. *Frames* became one of the first O-O technology for representing knowledge. However, they were more concentrated on how to organise and call procedures than on reasoning on objects and their properties. A more advanced approach to represent knowledge is through objects and the relationship between them. O-O programming languages provide both a declarative approach for specifying objects and a procedural approach

for defining what happens with the objects. In contrast with frames, modern programming languages, like Java, support extra features such as data abstraction, inheritance or encapsulation, a method used to avoid direct manipulation of private instance variables from outside the same class [30]. These languages have the advantage of making it quite straightforward to add new functionalities or to craft new behaviour into existing programs. Several limitations of O-O representation exist, e.g., *negations* like '*the mobile phone is not green*' cannot be expressed uniformly or *disjunctions* such as '*the mobile phone is either red or blue*' cannot be defined naturally. O-O models can be sufficient for quite a few practical systems designed and used nowadays.

Rules Representation

One of the most popular KR approaches is defined under the form of **rules**. Very basic rules can be regarded as condition-action pairs or *'if [condition], then [action]'* clauses which are used for expressing parts of a decision strategy of a system. For example, the simple rule *'IF it is sunny THEN I will go for a walk'* consists of two parts: the antecedent, premise or the IF part and the consequent, conclusion or the THEN part. It defines a condition which, if satisfied, produces a certain action to be carried out.

Rules can be seen as simple forms of KR but they are powerful tools, which encompass both procedural and declarative representations in a unified form. Rules are used to capture the information which is passed from the observations and understanding of a developer. According to this view, a rule is not just another mechanism for KR but a form of actual human behaviour. Rules that are specifically created for a system do not make sense when used in other rules systems [31]. Nevertheless, rules are being used more and more to define a variety of situations and cases like business requirements, conventional behaviour, policies and protocols. This leads to an increasing popularity of rules engines as components of various software systems [27, p. 327].

Rules are a good overall mechanism for representing knowledge because they can express **relations**, e.g., *IF there is no water in the tank THEN the tank is empty*, or **recommendations**, e.g., *IF it is cloudy THEN take an umbrella*, or **directives**, e.g., *IF the patient is training AND there is a high risk for accidents THEN contact the doctor*, or **strategies**, e.g., *IF the device is low on battery THEN activate low battery mode (step1) IF step1 is complete THEN check for minimum level of battery*, or **heuristic**, e.g., *IF a liquid is spilled AND the liquid pH is smaller than 6 AND the it smells like vinegar THEN the liquid is acetic acid.*

Rules are divided into several distinguishable families of rules, as shown in figure 2.2. The main families are *integrity rules*, *derivation rules*, *reaction rules*, *production rules* and *transformation rules*. Figure 2.2 shows an im-

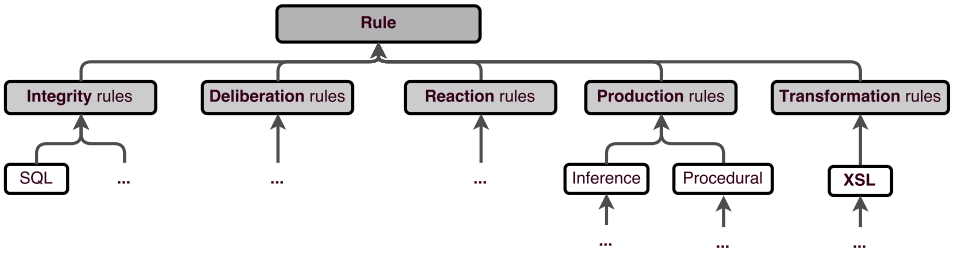


Figure 2.2: The main families of rules [1].

portant subcategory of rules for each family, except for the derivation rules and reaction rules, which are described in detail in figure 2.3.

Among the main families of rules, the most common are is the family of *production rules*. In literature, different classifications of rules are portrayed [23, p. 205], [32, p. 22], [33], [34, p. 237]. In AI, the most common types or rules are *knowledge rules* and *inference rules* [23, p. 205]. Knowledge rules, known as declarative rules, are used for declaring all the facts and the relations regarding a certain situation or a happening, while inference rules, also referred to as procedural rules, are used for recommending approaches for dealing with the situation or the happening. A concrete example of a knowledge rule is *'IF there is to much oil on the market, THEN the price of the oil will go down.'*, while *'IF the required data is not available, THEN ask the user for them'* is an example of an inference rule. In rule-based systems, knowledge rules are designated for the knowledge base, while the inference rules are integrated into the inference engine. More about rule-based systems and their components in the next section. *Meta-rules*, another important type of rules, are used to illustrate how to make use of the other rules. More precisely, they describe the behaviour of other rules. For example, *'IF the patient cannot breathe AND his/her pulse is very low, THEN apply the set of respiratory rules plus the alert rules'* is a classic example of a meta-rule. Turban et al. [23, p. 205] considers meta-rules as inference rules, while Matsatsinis et al. [34, p. 237] argues that meta-rules represent a different type of rules containing knowledge on how to handle the other rules. In [32, p. 22], rules applied in the business domain are divided into multiple types such as *facts*, *definition rules*, *integrity rules*, *production rules*, *reaction rules*, *transformation rules*, *data processing rules*, *control rules* or *meta rules*. They are either used for defining terms and notions in current use or for defining how to utilise other rules.

Another particular type of rules, used by rule systems on the Web, is represented by the rules of the Rule Markup Language (RuleML) family, which comprises a taxonomy of languages, sublanguages and subfamilies expressed in the Extensible Markup Language (XML). RuleML contains two main types of rules, i.e., *deliberation* rules and *action* or *reaction* rules, with

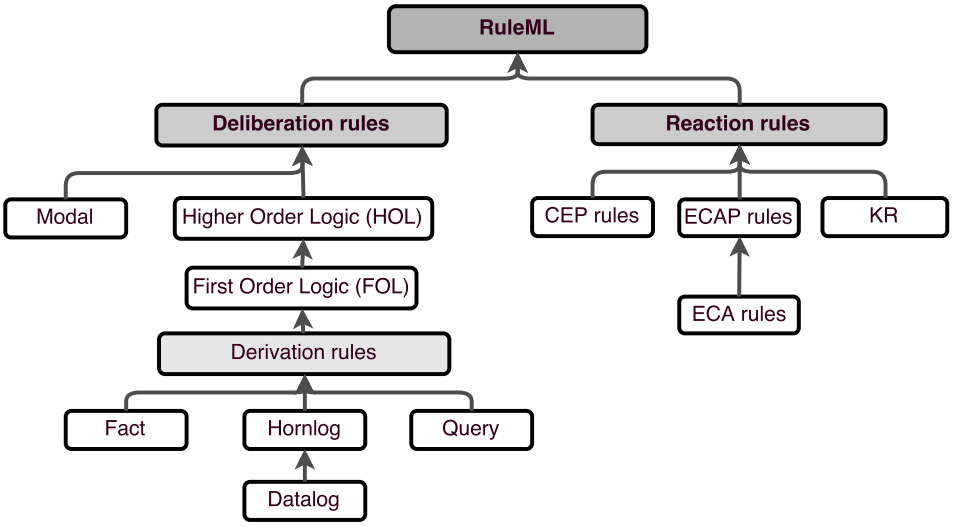


Figure 2.3: A simplified version of the taxonomy of RuleML rules [2]

multiple other subtypes of rules used by different systems, as illustrated in figure 2.3. For example, *Event Condition Action (ECA)* rules, a well-known subtype of action rules, are used intensively in event driven architectures like big data management and in active database systems. The ECA rule shown in figure 2.4a specifies how, for the occurrence of an even, i.e., a new customer registers for a flight, the desired output action is triggered, i.e., a new promotion will be offered, if certain conditions are fulfilled, i.e., the flight is operated by the *Luxair* company, the destination is the city of Luxembourg and the time travel is bigger than 7 hours. However, the behaviour of the ECA rule in this example is reactive, meaning the action of offering the promotion will only be triggered after a customer will buy his/her plane ticket. A proactive behaviour would be offering this promotion when a customer would be searching for a flight which would meet all the requirements.

Another example of a rule of the RuleML family is presented in figure 2.4b. It represents a subtype of derivation rules, which are subtype of deliberation rule. In the presented example, the rule is classified as a business rule. It is a representation of the sentence expressed in English: '*A seller is premium if his/her selling amount has been minimum 1000 euro last month.*'. Datalog, a markup language which combines Structured Query Language (SQL) and Prolog, is the foundation for the kernel of RuleML. Besides being able to define facts for each row of a relational table, it can be used to define rules that correspond to tables. It contains similarities with the Java languages, which defines classes with capital letters and methods with lower letters. In Datalog, the upper-case tags are used for types and the low-case tags for roles. The *seller* is a variable, represented by the mark-up

```
<Implies>
  <head>
    <Atom>
      <Rel>premium</Rel>
      <Var>seller</Var>
    </Atom>
  </head>
  <body>
    <Atom>
      <Rel>selling amount</Rel>
      <Var>customer</Var>
      <Ind>min 1000 euro</Ind>
      <Ind>previous month</Ind>
    </Atom>
  </body>
</Implies>
```

(a) A Datalog rule

```
ON
    AddFlight (Custid,
        Airline,FromCity,
        ToCity, Depart,
        Arrival)
IF
    Airline = 'Luxair'
        and
    ToCity = 'Luxembourg'
        and
    Arrival-Depart >= 7
THEN
      OfferPromotion
        (CustId,
        'ParfumePromotion',
        'LuxParfume')
```

(b) An ECA rule

Figure 2.4: Rules of the RuleML family.

'<Var>seller</Var>', and is assigned a unary relation, i.e., *'<Rel >premium</Rel>'*. This is only one example which shows how RuleML Datalog rule allow complex conditions and variables to be defined. Nevertheless, the XML representation of rules is limited if compared to O-O representations, being less natural and less flexible in representing complex situations and the relations between them.

2.1.2 Additional Challenges in Knowledge Representation

KR involves a couple of particular challenges that are independent of any particular representation or implementation. These challenges point out more general issues in KR like choosing the *granularity level* [35], dealing with *uncertainty* [36, p. 199], handling *information overload* [37] or choosing the right *method for representing set of objects, attributes and their relationship* [38, p. 65].

The central point of KR is to provide a method for a developer to express his/her knowledge in a formalised way. This basic definition contains already one of the important challenge in KR, i.e., choosing the adequate granularity level [35]. More precisely it refers to finding the correct balance for expressing a situation between lightweight methods of KR and more complex methods of formalism.

Situations can be expressed in terms of entities, i.e., variables, objects, their attributes and their values, or in terms of sets of entities. Choosing the right representation can be complicated, especially when dealing with sets

of objects [38, p. 65]. For example, certain properties of objects are valid or true as components of a set or a collection of objects but are not valid if taken as individual entities.

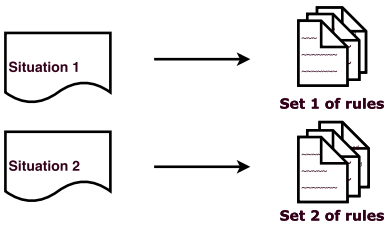
The rapid growth of information available in ubiquitous environments has led to an information overload problem. This is an issue for developers that want to integrate their knowledge into software systems as they are faced with an increasing number of complex situations they have to analyse. Developers have to design advanced structures that will allow them to represent all the entities and their relations related to their applications. This challenge is not particular to KR and is mostly related to software engineering.

Humans have the tendency to apply familiar methods to solve a problem, even if the problem is new. When a new area of knowledge or a new situation in a particular area is involved, people opt to apply common knowledge they already have to deal with them. But, inevitable, there will always be additional impediments, so it is necessary to come up with new approaches to address modern requirements. The representation of new requirements is a major challenge in multiple fields in computer science like AI.

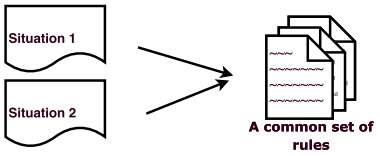
Challenges for Rules Representation

Apart from the more general challenges and issues in KR, there are several challenges specific for each particular approach of KR. For example, choosing to express the knowledge of a systems in terms of rules brings a couple of difficulties [27, p. 327] [39, p. 142] [40, p. 3-20] such as overcoming the traditional thinking model of using algorithms that are defined monolithically, splitting different situations into multiple sets of rules, finding a good mechanism for grouping and classifying rules or finding a balance of which attributes, relations and actions to put inside a rule. These are mainly conceptual problems that have to be solved manually by the developer, which decides which part goes where and how the main goals are achieved with each set of rules. This process tends to be less efficient if the developer does not have a guiding algorithm for making the related calculations or a method for allowing effective rule-grouping.

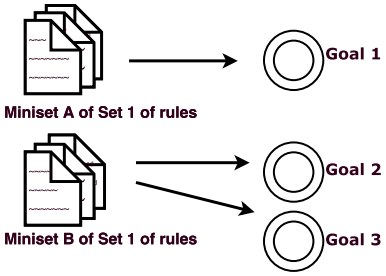
Software engineers and human experts in specific domains are accustomed to adopt a single algorithm to solve a specific problem. However, when it comes to rules or sets of rules, the algorithm is composed of a series of mini algorithms or smaller pieces, independent from each other, that work together to achieve some goals. These smaller pieces develop into the conditions and the actions of the rules. Once this model is adopted by the developers of the rules, it gets quite straightforward to break situations into rules. In spite of adopting this model, the solution brings up a new challenge: should each situation or mini algorithm be grouped in separate sets of rules, as exposed in figure 2.5a? Or should they be all put together and



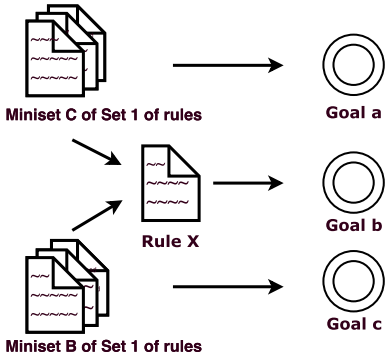
(a) A set of rules for each situation.



(b) A set of rules for multiple situations.



(c) Sets of rules with one to many goals.



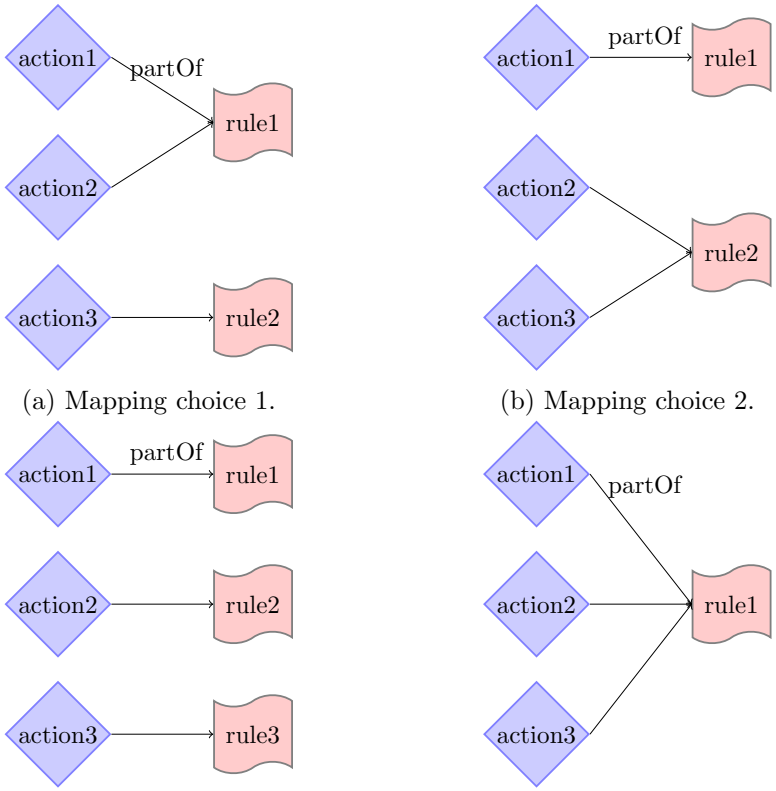
(d) Set of rules that share rules.

Figure 2.5: Conceptual and design challenges in rule representation.

no differentiation made, as illustrated in figure 2.5b? If the rules are not grouped in a structured manner it will lead to big problems when having to handle sets of thousands of rules. But if the rules are grouped into distinct sets of rules, should each set of rules have one particular goal, as shown in figure 2.5a by the first mini-set of rules, i.e., mini-set A, or they could be targeting multiple goals, as the mini-set B of rules in figure 2.5a?

If a solution where each mini-set of rules is assigned to one goal is chosen, it would solve partially the problem because another issue will need to be addressed, i.e., the fact of having common rules. Figure 2.5d presents a sketch of this new sub-challenge. For example, considering an e-Learning scenario, a system would want to notify the students in several different cases like when there is a new assignment or when they forget to complete their assignment. The notification action would be designated to be carried out to a single rule, which would contain the algorithm for sending generic messages. It would make no sense to have multiple rules with the same code but with different names just to simplify the conceptual design phase. But what would happen if the same rule would be called simultaneously by two other rules? This could generate a conflict at the level of the mechanism or engine which processes the rules.

Digging even deeper, at the level of the structure of a rule, several questions arise in terms of how should the mapping of actions and rules be done,



(a) Mapping choice 1.

(b) Mapping choice 2.

(c) Mapping choice 3.

(d) Mapping choice 4.

Figure 2.6: The mapping between functions and rules.

as shown in figure 2.6. For example, should there be only one action per rule like in figure 2.6c or could several actions be implemented in the body of the same rule like in figures 2.6a, 2.6b or 2.6d? And if a rule would contain several possible output actions, up to how many functions or instructions should a rule have?

The simplest option would be to choose the most basic type of rule, i.e., the *'IF condition THEN action'* rule. But this would lead to some limitations in terms of expressing more complex instructions and strategies. On the other hand, if a rule becomes too complex regarding the functions it has to execute or the operations it has to perform, it could cause problems in terms of execution time or, if one of the operations or functions of the rules would fail, it may cause the entire algorithm of the rule to stop. And, because part of the operations rules perform in general imply the generation of other rules, reaching several goals of the applications could be delayed or even interrupted. Finding a correct balance between the elements that compose the structure of rules is an important aspect that the developers have to bear in mind when developing complex software systems.

2.2 Reasoning

If in the previous section the thought of providing a **structure** for capturing and expressing knowledge was described, this section is more focused on the **behaviour** of systems that are using some of the formerly mentioned structures for representing knowledge and on their actions.

The term **'reasoning'** is applying to about everything which involves taking a sophisticated decision. Reasoning is about an existing mechanism that can interpret knowledge and about determining what to do with this knowledge. Several types of reasoning can be differentiated, from *analogical*, *conditional* or *inductive* reasoning to *traditional logic*, based on the analysis of premises [41, p. 156].

In everyday use, all the software systems are reasoning systems as they incorporate, in their code, some mechanism for taking decisions. However, most of these systems are using simple, straightforward techniques of reasoning like *'if else'* statements or mathematical propositions. Even though many authors refer to more advanced reasoning processes as *intelligent* reasoning this utilisation of the word *intelligence* is not agreed by all the authors which argue that intelligence is only a term to describe a typical form of human behaviour [42]. Either way, the most common use of the term **'reasoning'**, in computer science, is related to the methods or strategies of applying the computer representation of logic.

Reasoning systems cover a wide range of applications from robotics, computer vision and game theory to natural language processing or business management. A number of significant categories of reasoning systems have been used or proposed for reasoning in AI systems, workflow systems or deductive database systems. *Knowledge Management* is a well-known field where reasoning plays an important role. It is, however, more centred on organisational knowledge and how to make the best out of it in companies, public institutions or other large organisations. More relevant to this dissertation are the categories of reasoning systems used in AI. These categories include constraint solvers, theorem provers, logic programs, rule-based systems, deductive classifiers, machine learning systems, expert systems, case-based reasoning, agents or intelligent agents, or procedural reasoning systems. Many of them share, to a certain degree, strategies, techniques and various algorithms for applying knowledge.

Rule-based systems, for example, use rules to represent knowledge and inference engines to manipulate the knowledge to arrive at a particular solution. Deductive classifiers, a concept which started to attract attention after rule-based systems were studied intensively, use frame languages, similar to O-O languages, to describe classes and their relations. Case-based reasoning systems address an issue by focusing on previous problems related to that particular issue, which already have a solution. Logic programs are another side of the coin because they rely on programming languages like Prolog to

formally solve problems.

2.2.1 Rule-based Systems

RBSs employ rules not only to represent knowledge, as seen in section 2.1.1, but to encode the behaviour of the system. And, by referring to the behaviour of a system, it is meant to include defining actions as capturing, distributing and applying knowledge. Trying to define problems and to solve them by thinking in terms of rules is one of the most common and old ways of human reasoning, including the most famous set of rules, the *Ten Commandments*. Going not so back in the past, in the eighties, traditional rule-based systems were considered to be one of the best solutions for encompassing the know-how of a human expert [43].

Even though RBSs are basically tools present everywhere in technology and science, most of their usages are done in a straightforward way, without taking into account their properties or considering any kind of optimisation. They are one form of implementing knowledge-based systems and applying them to fields like AI or Knowledge Engineering (KE). In fact, RBSs provide a paradigm for fields like decision support, system monitoring, intelligent control, operational knowledge encoding or situation classification [5, p. 7]. They support the development of applications in the previously mentioned fields. Numerous successful applications using RBSs have been developed, including controlling the power on a research reactor at the Massachusetts Institute of Technology [44] or, in the medical domain, to help the diagnosis of aphasia's subtypes and the classification of pap-smear examinations [45].

A RBS consists of at least 3 main parts: a collection of rules or a knowledge base, a collection of facts or a working memory and an interpreter, sometimes called inference engine [46, p. 126] [47]. Some authors consider that simple RBS can be composed only of a rule base and an interpreter [p. 270][48]. *Knowledge Engineering* is a wide discipline that involves the creation of RBSs [49]. KE encompasses steps like problem assessment, representing the knowledge as facts and rules, or choosing the appropriated software tool to develop the RBS and testing it. The collections of rules represent long term knowledge, while the collections of facts, which are case-specific, represent a form of short term knowledge. Facts are unconditional statements, which are true at the moment of their usage [50, p. 19]. Or they can be simple objects relevant for rules. The way facts are obtained in a RBS differ from system to system. They can be acquired from the sensors connected to the system, they can represent information obtained from the user's input, can be already stored in the memory of the system or can be derived from rules. The *inference engine* is a mechanism for processing rules and is carrying out the reasoning process for finding a solution by making the connection between the facts and the rules.

Reasoning in RBS is the way in which rules are put together to create

new knowledge. It is performed according to two main schemes that manage deductive inference: *Forward Chaining (FC)* and *Backwards Chaining (BC)* [46, p. 195].

The FC scheme is a data-driven approach, meaning that the initial facts are known but not the necessary conclusion. In a FC system, the working memory, which is in a continuous process of updating, is responsible of managing the facts. The inference engine, also referred to as the rules engine, is in charge of handling the rules and of taking the right action course needed in each situation. Rules are triggered when their specific conditions are fulfilled. These conditions need to match patterns of facts in the working memory. If the facts are matched, then an action, which normally adds or deletes something from the working memory, is performed. The steps of a basic algorithm of a FC system are to 1) find the rules whose conditions hold, 2) select one of the rules to fire and 3) to carry out the targeted actions. The second step involves an extra step for choosing a strategy for handling conflicts between the rules. The algorithm is part of a cycle which is repeated until the targeted goal is reached or there is no rule which fires any more.

Two of the most well-known examples of FC systems are XCON, written in the forward chaining rule-based language OPS5 and used for configuring VAX computers [51], and CLIPS [52], one of the most used RBS. Alternatives to FC approaches imply either that priorities should be attached to each rule or the first rule found should be executed or the rules that match should be investigated in parallel. The Rete algorithm was proposed to solve the problems that concerned the rule matching step, which was taking more than 90% of the time of a RBS [53].

If the designers of a system know what the conclusion might be, a FC system would be less efficient than a BC system. In a BC system, as its name says, the starting point is the goal of the system. In contrast with FC, which is applied at assertion time, BC is used at query time. In order to achieve this goal or to satisfy the initial hypotheses, the BC system queries the knowledge base for validating the hypotheses or to determine if the goal can be reached with the available information.

One of the main differences of these two approaches, FC and BC, is that the rules used in each approach differ from each other. Even though rules keep the same *'IF...THEN...'* structure, a FC rule would not usually contain an action after the *'THEN'* statement but, instead, it would contain a certain state. If the premises or the conditions after the *'IF'* statement are true then the state is valid. For instance, the rule *'IF (lecturing Programming 1) AND (writing article) AND (day Monday) THEN (overworked person)'* is an example of a BC rule. A BC system does not have to keep a working memory but is more concerned about the list of goals. BC is used, for example, in Prolog and for solving SQL queries.

2.2.2 Expert Systems

Two decades ago, in 1997, IBM's super computer, Deep Blue, defeated Gary Kasparov, the world chess champion at that time [54]. It was one of the first signs that AI systems are able to surpass the human intellect in a particular domain. Later on, in 2011, another super computer from IBM, Watson, consisting of ten racks of ten Power 750 servers, defeated two of the best *Jeopardy!* players in the world [55]. This was yet another example of how man was overcome by a machine, how machines are getting closer to passing the Turing test [56]. These two famous cases were actually implementations of Expert Systems (ESs), where human expert knowledge in a particular field was integrated in a computing system. These cases show how it is possible for computers to *'understand'* and to analyse complex situations with precision, including many variables and possible scenarios, and how software programs can take the best possible decision in the given circumstances. Nowadays, advances in the particular field of AI reached such a development point that chess programs are able to beat any human chess player in the world.

Some authors make no distinction between ESs, RBSs or production systems [41, p. 149] [57, p. 68]. RBSs and production systems are frameworks that help construct ESs, rather than the ES itself. Production systems are FC RBSs, while ESs can be implemented with the help of RBSs, case-based systems, neural networks or a hybrid model, a combination of the previous ones [57, p. 68-71]. A RBS, for example, becomes an ES when there exists an ontological model for representing the domain and the other stages like knowledge acquisition. Just because some knowledge is represented inside a rule does not make the system an ES. RBSs can be constructed without the assistance of a human expert in a certain domain as it can contain simple instructions.

The goal of an ES is to replace human experts in a certain domain [58]. This does not mean that an ES applies the thinking model of the specialist, it just focuses on the knowledge provided by the expert on how to address a problem in a specific domain. Several definition for ESs exist but they all differer in one or several ways and so, no common, unique and precise definition exists in the literature. Despite the lack of a single definition, ESs present a couple of characteristics they all share like the capacity of separation between knowledge and reasoning, the ability to perform reasoning over different KRs or the feature of solving a problem heuristically [59, p. 3] [60, p. 35]. More precisely, the first characteristic refers to the clear separation of the knowledge base and the inference engine, which is also a general strategy used by all traditional RBSs. The second one refers to the multiple KR schemes available as ESs are implemented using various programming languages such as Prolog or Java. And the third characteristic refers to the rule of thumb which is applied when solving a problem. These features are

```
rule "validate payment"
dialect "mvel"
dialect "java"
when
    $h1 : Payment(
        paymentType in ("Euro", "Swiss Francs") && (type in ("VISA",
            "MASTER-CARD") || bank in ("Luxembourg", "Switzerland"))
    )
then
   System.out.println( $h1.paymentType + ":" + $h1.bank);
   $h1.setValidState(true);
end
```

(a) The 'validate payment' rule in Drools.

```
public class Payment {
    private String paymentType;
    private String type;
    private String bank;

    private boolean validState = false; //the payment is initially invalid

    //setters and getters for each field
}
```

(b) The class Model of the 'validate payment' rule

Figure 2.7: A rule written in Drools language and its corresponding Java Class Model.

important because they make the differentiation of ESs from conventional programs.

ES technologies include *specific ESs*, *shells* or *ES development environments*. Specific ESs are that category of ESs which offer recommendations in a specific task domain. ES development environments include tools like the high level symbolic programming languages like LISP as well as hardware devices like workstations or minicomputers. ES shells are tools that provide both an inference engine and a user-interface for it. When shells are provided with a knowledge base then they become ESs. Shells are used intensively in developing ESs and not to solve a particular problem by becoming the ES itself. Advanced shells offer the possibility of choosing from multiple KR approaches and reasoning techniques. Among the best-known ES shells, of particular interest is the Java Expert System Shell (Jess) [61], which is a rule engine, firstly developed as a Java port for the CLIPS language, that allows experts to build Java software that has the ability to *'reason'* using

knowledge provided by the experts in the form of declarative rules. Jess can be really easy to integrate with Java-based web-enabled applications as it creates its rules in XML format. Another option is the open-source rule engine, based on Drools, called JBoss Rules [62]. A big advantage of this implementation is the adaptation of the Rete algorithm for the O-O approach [41, p. 212]. This approach can be quite handy when an expert in a field that does not have programming skills in a certain language can express his/her expertise in a more natural way.

For example, figure 2.7 contains a simple rule written in Drools and its corresponding class model in Java. The 'validate payment' rule, illustrated in figure 2.7a, is set to print out the details of a payment and to validate the payment when certain conditions are meet, e.g., the payment has to done at a bank either in Luxembourg or in Switzerland. It uses two dialects, mainly the Java and the MVEL, which is focused of pointing the variables to the setters and the getters declared in the model class, shown in figure 2.7b. This category of rules is well-suited for writing business applications but still has a lot of catching up to do for the mobile world, even though attempts have been made [63].

Rule-based Expert Systems

In general, this subtype of ES is one of the most representative subtypes for ESs. This subcategory includes famous programs in the early eighties such as MYCIN [64] and DENDRAL [65] or, more recently, programs constructed with the help of CLIPS, an ES shell. Rule-based Expert Systems (RBESs) constructed in CLIPS are widely used in sectors like industry, government or academia because it offers support for three important paradigms, i.e., O-O programming, rule-based programming and procedural programming [52, p. 912].

A RBES consists of five main parts, i.e., the knowledge base, a database, an inference engine, an explanatory subsystem and a user interface [41, p. 154] [66, p. 151]. The last two parts are extra components specific to RBESs. The explanatory subsystem is a tool which provides explanations to the end-used about the reasoning schema employed by the ES and how was the conclusion reached. This can be useful when checking the active rules that are currently running on the system or the rules which were unsuccessful. The user interface is in fact the tool which permits the end-user to work with the ES and can consist of several graphical elements like interfaces or menus.

2.2.3 Agents

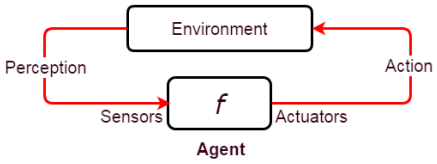
Agent is an umbrella term which covers multiple domains. Many authors tried various definitions for describing what an *agent* is [67, 68, 69, 70, 71].

The term of *agent* is associated by many other authors to the ability of a system to mimic the human intelligence. That is why these authors use the notion of **intelligent agent** to express the same meaning. A common agreement for a precise definition, accepted by the AI community, has not yet been found due to the many research sub-fields in AI, ranging from more general topics such as perception, knowledge or logical reasoning, to specific areas such as medical diagnosis, robots or prediction systems, where *agents* have different properties, specific to each sub-domain. *Agents* were proposed in the idea of being a tool for analysing software systems and not for splitting the world into agents and non-agents [67, p. 33].

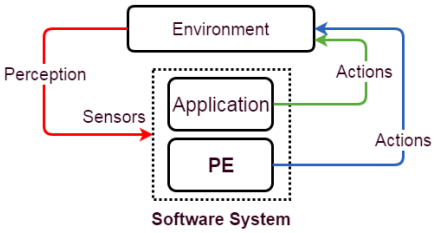
Definitions for agents can be either very general, e.g., an *agent* is an entity that perceives a particular environment through its sensors and acts upon it through effectors [67, p. 31], as depicted in figure 2.8a, or very specific, e.g., *agents* are systems that have properties such as autonomy, social ability, reactivity and pro-activeness [69]. In [71], *agents* are considered as entities capable of acting in the place of someone with explicit permission. This definition is very general and weak, and can be interpreted very easily in many different ways. Some authors make a clear distinction between *agents* and *autonomous agents* [72], while others impose some more restrictive properties for defining agents like sensing and acting autonomously or like realising, in the sense of accomplishing, a set of goals and tasks [70].

If a general definition is taken, then, an agent can be nearly anything or anybody that is able to take decisions on its own. Thus, for practical reasons, the authors in [73, p. 23] proposed that agents should be divided into several categories, i.e., **physical or tangible agents** such as robots, **software agents** or pieces of software code that can take autonomous action and **natural agents** or humans and animals. Following this line of logic, the authors of the first definition [67, p. 31] extended and explained further their initial definition and thus, agents became software agents. A *software agent* is a computer program that performs a certain task if it decides any action is necessary [74]. In order to be distinguished from simple software programs, they have other characteristics such as autonomous control, can perceive their environment, are adaptive, can persist over a longer period of time and can take over another's goals [75, p. 4]. Also, in [69], the authors distinguish between a weak notion of agency, provided in the previous paragraph, and a strong notion of agency, i.e., where an agent has more human-like properties such as knowledge, belief, intention and obligation.

As pointed out previously, the difference between these definitions is quite unclear. They try to identify what do various software agents have in common in order to distinguish them from other programs or software systems. In [71], authors indicate that if all the software agents are in the end software systems, the opposite, i.e., that all software systems are software agents, is not true. They propose two properties for verifying if a program can be considered as a *software agent*: the test of temporal conti-



(a) The representation of an agent



(b) A proactive software system composed of an application and a Proactive Engine (PE)

Figure 2.8: A side-by-side representation of two related software entities.

nuity and the perception of its environment. They argue that the majority of programs are failing one of the two properties. The third property, **autonomy**, is another requirement considered essential in other studies [73, p. 26] [76]. However, **autonomy** or an **autonomous** agent, has various meanings, varying from one author to another. In [76], autonomous means that an agent can perform an action without implicit intervention from a user or other programs, while being in control of its own actions and internal state. On the other hand, in [75, p. 37], *autonomous* refers to the ability of an agent to lean as much as it can to fill out the gaps caused by partial or incorrect prior knowledge. More precisely, *autonomous* refers to a system that can opt to perform a task free from outside influences [77]. There are other properties, like *collaboration* or collaborative behaviour and adaptiveness, which are used by some authors to define agents [78, p. 112], but it is more a strong property for a certain type of agents, i.e., **multi-agents**.

The previously described properties are not outlined for creating a tool that checks if a software system is a software agent, but it is dedicated for comparing the proposed computing model described in chapter 3 to a software agent. The computing model presented in this study, i.e., the Proactive Engine, is a rule-based system, which executes rules. It is used as a complementary software and, if it not used aside another program, it has no purpose because initially it does not contain any rules to execute. The task of creating rules for a certain application is assigned to the development team. The PE is executed, on the server-side Operating System (OS), like a normal Java application [14] and on the mobile OS as a middleware software [79]. The OS can interfere at any given time, for well-known reasons like resource limitation, and can stop the activity of the PE. The engine depends on the environment where it is executed. This will affect the *autonomy* or the capacity of acting on their own and the *long-time property* which need to be satisfied by a software agent. If it is not taken into consideration the intervention of the OS, then these 2 properties are weakly satisfied because the PE is designed to run and execute rules continuously in the background.

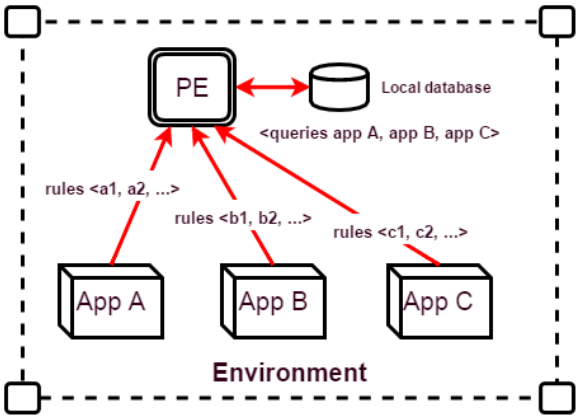


Figure 2.9: The architecture of a Proactive System.

The PE taken together with an application and a set of rules, as seen in figure 2.8b, is not similar if compared to a software agent, as the rules are predefined on the Proactive System. There is no mechanism for generating new code or new rules, and so, the learning process is limited to what is coded inside the rules. The PS does not help building intelligence, it acts how it was programmed, based on the integrated knowledge. Figure 2.8b shows that both, the PE and the application, can take different actions and interact with the environment in different ways. Another point, shown in figure 2.9, is that the PE was designed as a system where multiple applications could execute sets of rules or PaSs, without developing a new execution mechanism each time. On the other hand, software agents have exclusive access to their data [73, p. 73]. If an external process or another program can change the agent's data, it would not have control over them and this could prevent it from following its goals. An agent builds knowledge and needs a structure to store and represent facts, actions, goals and its computing environment. This helps the agent to deal with **uncertainty** or situations that may occur and which were not anticipated. The PE, on the other hand, does not handle unforeseen situations, it performs only actions that are integrated in its rules. And, to know from the beginning all the possible situations that might happen and to create a rule for each of them, is close to impossible when talking about complex computational systems.

To summarise, even though the proposed software system in this work can be seen as a ***software agent*** because they may share some properties like context-awareness, adaptiveness or collaboration, they are, nevertheless, distinct entities. A PE could be used together with a software agent for providing proactive features, addressing one of the important research questions in agent theory, i.e., about providing an agent with proactive behaviour.

Chapter 3

The Proposed Computing Model

As it has been pointed out in the previous chapter, RBSs comprises both an approach to support robust software development and a simple and well-structured method to represent knowledge and to reason about it. Among other benefits, RBSs give the developer the possibility to add rules to extend an existing set of rules, if the behaviour of the system is not complete or some additional unprocessed situations or actions were discovered at a later date. However, a pure, traditional rule-based approach has its limitations as any other particular approach when compared to others, as also seen in the previous chapter. O-O programming, opposed to traditional RBSs, uses objects as knowledge units, which describe various concepts and operations. However, a solution which will combine the advantages of procedural programming, of logic programming, of declarative programming and of O-O programming is not so easy to design. Following this line of reasoning, in this chapter, a solution incorporating both the O-O paradigm and the rule-based paradigm is proposed. This approach is preferred because it benefits from the advantages of both paradigms and because several limitations of each particular paradigm are overcome.

3.1 The Proactive Engine

The model proposed in this chapter for representing knowledge and for performing advanced computations is called the ***Proactive Engine***. It is a rule-based system designed to work not only with common data types, variables and facts, which are supported by the majority of rule languages, but also with *objects*. Objects are necessary because they correspond usually to real-world entities and because they can express quite complex structures. Instead of providing a solution that implements objects into rules, which suffers because rule-based programming is not as portable as O-O

programming [80], this chapter offers a model which uses O-O languages as a foundation for implementing rules. The level of expressiveness, precision and wordiness often depends on the programming language. For example, high-level programming languages like Ruby or Python may be better in terms of expressiveness, while lower-level programming languages like Java, C or C++ tend to go for wordiness.

The **PE** is conceived as a middleware system, performing actions between the OS of the device and the applications. Together with a software application and a corresponding set of rules it represents a PS. The **PE** is intended to behave 'silently', in the background of the OS, where it operates, the same way as *background services* do on many OSs. This approach is done with respect to the PaC paradigm, which is intended to remove the person that uses applications from the computational loop and put him/her in a supervising role. This way, the user does not have to give an explicit command in order for the system to perform an action, e.g., execute a rule.

The ***PE*** is responsible for maintaining a precise overview of the PS's goals and for executing Proactive Rules. It is also used for controlling the state of the PS. The object-oriented rule-based structure ensures an intelligent and automatic management of knowledge.

3.2 The Modular Architecture of the Proactive Engine

The **PE**, as a middleware rule-based system, is composed of multiple components, i.e., a Rule Engine (RE), a local database, a Queue Manager (QM), a Notification Manager and a Communication Engine, as seen in figure 3.1. These components work together for allowing the **PE** to compute proactively. Each of the PE's components has a different function and is in charge of a particular process. For example, the RE is taking care of executing all the rules that are in the current queue. More about each component's function in the next subsections. An additional component or a database Application Programming Interface (API) is added when the **PE** works together with a system which has its own database. This database API is composed of a set of subroutines and queries necessary to access the complementary software's database. The main advantage of the PE is that it keeps the logic separated from the code of the application. Being split in different components or modules it is much easier for developers to work or modify a single unit instead of changing the entire system.

With such a modular architecture, the PE is able to capture relevant *context information*, support *distributed reasoning*, to *act pro-actively*, to provide *adaptation mechanisms* and to perform multiple *collaborative* actions at the same time.

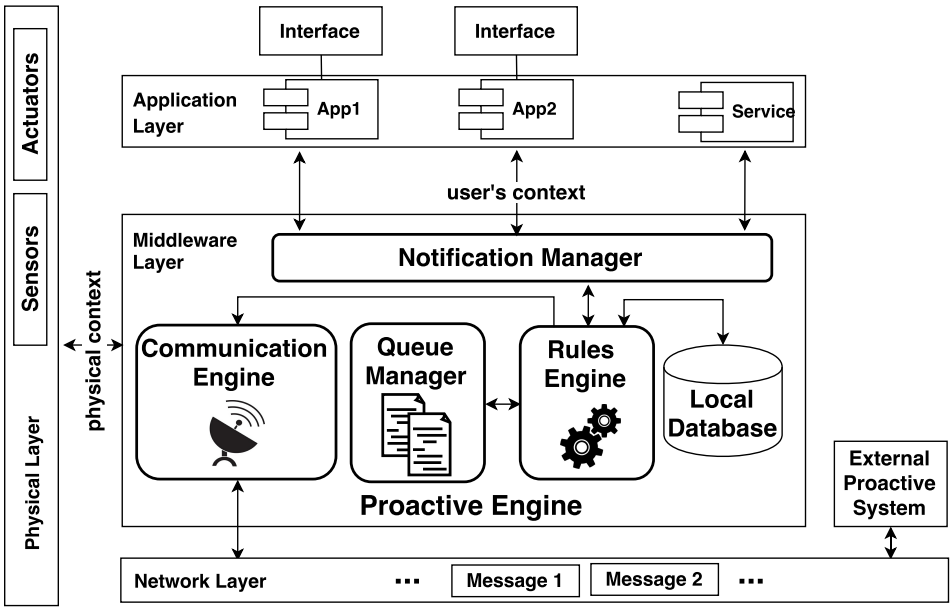


Figure 3.1: The Modular Architecture of a Proactive Engine.

3.2.1 The Rule Engine

The ***Rule Engine*** is a mechanism that is responsible for executing Proactive Rules. This process happens in **iterations**. An *iteration* is a repetitive step, consisting of a step of verification and execution of Proactive Rules (PRs). The duration of an iteration can vary, in function of the number of rules that are in the queue of rules that have to be checked for execution. The duration of an iteration depends on the computing power of the system. An example of how many PRs can be executed in one iteration on different OSs and how much time it take for the iteration to carry out all its tasks is shown in chapter 3.4. The time interval between two consecutive iterations is always constant and its value is determined by the developer. It can be adjusted in function of the performance of the device on top of which the PE operates.

Basically, the functions of the RE can be broken down as follows:

- Selecting Proactive Rules to be examined or checking the Current Queue for a list of Proactive Rules;
- Selecting the Proactive Rules to be fired in an incremental order;
- Checking the activation guards of each selected Proactive Rule for *triggering* the rest of the phases;

- Finishing the execution of the different phases of the triggered Proactive Rules.

The RE executes PRs in the order they are added in the queues of the QM. They can also have priority labels, to specify the priority order in which the PE should execute the PRs. It contains 3 important parameters: *F*, or the frequency of an iteration, *D*, or the latency that can occur between consecutive iterations and *N*, or the maximum number of PRs that can executed during an iteration.

Multiple instances of the same PR can exist on the PE. For example, the action of sending a message to the GUI of an application can be defined inside a PR. This rule can be instantiated multiple times in different situations, a process which can happen during the same iteration of the PE.

3.2.2 Proactive Rules

The rule that the RE is processing is a special type of rule, with a particular structure. To make it easier to refer to it, the name **Proactive Rule** is chosen. Its structure is more complex then the classical *IF-THEN* rule and other types of rules presented in chapter 2.1.1, and is adapted for the O-O paradigm. It is composed of 5 parts and it was first proposed by Zampunieris [14]. These 5 parts include ***data acquisition***, ***activation guards***, ***conditions***, ***actions*** and ***rules generation***. Choosing a more complex structure for rules was done for the efficiency and simplicity of parsing rules and of executing them. Also, it is clearer to the developer where to put instructions or actions. This structure was particularly selected to comply with O-O implementations of the PS. This approach allows the PS to skip the step of checking for the type of the parameters of traditional rules which are part of a knowledge base. The 5 parts of the PR can be implemented in O-O programming languages using *methods*.

During the **data acquisition** phase, relevant contextual information is collected. This information is used in all the other 4 phases of the PR and it is vital for helping the PR to take different decisions. The **activation guards** phase is made out of a series of tests to establish if the rest of the rule is going to be processed. If the *activation guards* are evaluated as true then the next phase, i.e., *conditions*, is going to be executed for sure. Otherwise, the two next phases, i.e., the *conditions* and the *actions*, will be skipped. The **conditions** part or phase is making a series of tests, based on the information acquired in the first part of the rule, and evaluates these tests. If the tests are evaluate positively, then the PE can execute the next part, i.e., the *actions*. Phase 2 and phase 3 are very similar in terms of structure, representing validation tests before taking any action or generating any rule. They can be implemented as boolean methods in an O-O programming language. The fourth part or the **actions** is the composed

```
1: procedure The parts of Proactive Rule
2:     dataAcquisition
3:     if activationGuards then
4:         activated = true
5:         if conditions then
6:             actions
7:     return boolean ruleGeneration
```

Figure 3.2: The 5 sections of a Proactive Rule and their execution.

of a list of actions that are performed in a sequence. Such actions can include sending messages to the front-end users or inserting new entries in the database. This part will only be executed if the conditions in parts 2 and 3 are validated. And, the last part of a PR or the **rule generation** phase will determine if the PR will terminate its execution during the current iteration or it will continue to be executed at the next iteration. Technically speaking, this process is called *cloning a rule*. From a theoretical point of view, this ensures the continuity of a chain of rules, rules which imply a certain logic, a procedure which is described later on, in chapter 4. During this last phase, additional PRs can be generated. A very basic version of this algorithm in pseudo-code is given in figure 3.2.

Cyclic Rules

A PR can have the property of being ***cyclic*** if it continues to clone itself and gets executed by the PE at each iteration. Cyclic rules can be generated by other rules and do not have to run from the beginning, when the PE starts; this is the job of ***meta-rules***. These types of rules are usually the starting point of a chain or of multiple chains of rules which try to achieve a certain goal. This particular rule is very useful when the developer predicts the occurrence of an event or the lack of an event in the future. The rule does not have to trigger immediately other rules, but only when special conditions are accomplished, e.g., the date and the hour of the system is the same as the date and the hour specified inside the PR for automatically sending reports about a certain happening. The cyclic rule will continue to be in the next queue of rules which will be evaluated by the PE but will not continue its actions until the special conditions will be meet.

This example is one of the many examples where rules help specifying a set of instructions or actions that should happen only in particular situations. Cyclic rules do not have to be necessary listed in the current queue of rules from the beginning, i.e., the first iteration of the system, but can be added on the fly to the current or the next queue. It depends on what precisely the developer wants to check and what the system is designed for. Certain functions require cyclic rules that check at each iteration if certain

conditions are met or if certain things happened, in order to be able to provide a quick answer. As activation conditions tend to be complex, rules have to be able to wait for longer periods of time to be activated, e.g., multiple iterations of the PE. For example, a rule that is in charge of sending messages should not be cyclical because these messages can spam the receivers, if they appear more than one or two times (in case of very important events).

Meta-Rules

Meta-rules are a special type of PRs and have the property to trigger PaSs. They are normally the first rules in the current queue of the PE, integrated before the beginning of the first iteration of the PE. They can be seen as a starting point of the system. Similar to the concept of *metarule* used in AI, activating a meta-rule implies the triggering of additional PRs, which open chains of rules that compose PaSs. However, the difference is that *metarules* in AI describe how other rules should be used. Instead of having a logical chaining procedure of rules, the behaviour of this rule takes care of implicitly triggering other rules.

3.2.3 The Queue Manager

The Queue Manager (QM) is a structure composed of 2 First-in, First-Out (FIFO) lists called the *Current Queue (CQ)* and the *Next Queue (NQ)*. The CQ contains the PRs which will be executed by the PE during the current iteration. The NQ is an additional queue created for the next iteration, i.e., it contains PRs which were generated during the rules generation phase of PRs and that will be executed during the next iteration. At the end of the current iteration, the new CQ will be the old NQ. Then, the old NQ will be emptied and will contain no PRs at the start of the iteration. One queue for managing the PRs is not enough because of a particular situation, i.e., during an iteration new PRs are generated and they cannot be added at the end of the same queue as this would result in always reaching the value of N, one of the 3 important parameters of the RE, or the maximum number of rules per iteration allowed for execution.

3.2.4 The Local Database

For saving the state of the PS and for storing relevant information, a data storage component or a database is necessary. As seen for RBSs in general, a storing mechanism is needed for keeping knowledge. This is realised with the help of a *knowledge base*. However, there are a couple of differences between knowledge bases and databases. Databases are accumulation of information handled in such a way that links between its elements can be easily done, enabling users to quickly and efficiently access the desired information. In contrast, knowledge bases are more complex mechanisms and

require more computing capabilities. A database will continue to expand by adding information while a knowledge base would continuously absorb and transform information. Knowledge bases can include, for example, other mechanisms such as data mining methods. For a PS it is sufficient to work with a database, as it uses prediction methods to anticipate the situations that the PS will have to respond to. The queries for inserting, accessing and handling the data on the PE have to be implemented.

3.2.5 The Communication Engine

The Communication Engine (CE) is a component of the PE that handles the data exchange between various PSs. It is in charge of ensuring that messages are sent correctly, that they arrived at the destination and that the PE is keeping track of what was send, when and to whom. Its technical implementation can be done with the help of sockets, a widely-spread mechanism for sending and receiving data. Each PE is equipped with a CE in order to be able to collaborate with other PEs. The communication of PSs is done via messages, which activation commands for PRs.

3.2.6 The Notification Manager

The Notification Manager (NM) is an optional component of the PE, necessary only in cases when the PE would require a GUI. This can be very useful in general for checking and supervising the activity of the PE on a device or a machine, to see which PRs are executed and what the PS is doing in the background or, even more, to send directly notifications, messages and/or alarms both to the developers and to the front-users. In some cases, like the Android-based mobile devices, it plays a role in making the PE to work silently in the background of the OS.

3.3 Properties of Proactive Systems

Due to its design and architecture, the model proposed in the previous subsection has several important properties that are necessary for addressing many of the challenges of current computing networks and systems. These properties and how they occur on Proactive Systems are better described in the second part of this dissertation where real-world applications are developed. Each case study from chapters 5 to 7 demonstrates how these properties are revealed by empirical research.

3.3.1 Anticipation

A PS, due to its approach of representing and reasoning about knowledge called PaS, is able to anticipate future situations and the state of the system.

The future state of the system is deduced from the current state of the system. The necessary information regarding the state of the system is saved in the local database, a component of the PE, and is continuously updated. At a technical point of view, PaSs describe how should a PS respond in case of foreseen situations/events and does not provide a methodology to handle unforeseen situations or events that may occur on a system. From the user's point of view that is interacting with applications the behaviour of the PS, in some cases, would look like it is anticipating his/her next move. However, the PS will behave exactly how it is programmed by its creator.

Making decisions before a possible occurrence of a situation or an event is one of the features of PSs. PaSs make it possible for PSs to take decision in advance, due to their mechanisms of anticipation, of performing real-time observation and of being context-aware. The possibility of acting ahead of time determines the proactive behaviour of a system. PaSs are anticipating modules which include prediction approaches. They do not work with probabilities but the conditions of the PRs that compose them allow PSs to express a certain degree of anticipation. These series of conditions are patterns which can determine what action to be taken and precisely when.

3.3.2 Context-Awareness

Context, in computer science, refers to the information which is used for characterising a situation and its circumstances [81]. **Context-awareness**, or the ability of a system to acquire and utilise contextual information, is an important property of PSs because it allows them to monitor their computing environment.

Context-aware behaviours of PSs include showing refined information to the user based on the context, the automatic execution of services and adding contextual information to knowledge to transform it into wisdom. For understanding the current situation a PS is able to combine several contextual values [82] including *time context*, e.g., the day, week, month or year, *user context*, e.g., location, profile and social characteristics, *computing context*, e.g., network parameters and connectivity, and *physical context*, e.g., heart rate, noise level or temperature. The role of context-awareness fits the goals of PSs, i.e., to reduce the explicit interaction of the user as much as possible [83].

By using specific PRs, a PS is able to keep an eye on the slight or major changes that are occurring and is ready to act accordingly. Sensing information from one environment can sometimes lead to some inconsistencies if, for example, sensors are broken and they transmit the wrong information. But, together with the *collaboration* property, a PS can get extra information to check the consistency of the local contextual information.

3.3.3 Adaptiveness

This property is important as PSs can activate in different computing environments. This property refers to the capacity of the PE of adapting its behaviour depending on the computing power of the system on which it operates or depending on the different policies of the OS. It does not refer to *architectural adaptation* [83] nor to the **self-adaptive** property of autonomous systems, which can change the structure of their components, modify their behaviour while they are performing their duties and decide how to react, to what and when to reach to it. **Self-adaptiveness** includes all the **self*** properties of a system and, in this work, this does not concern PSs. All the adaptive measure of a PS are implemented in advance by the developer/designer of the system. The PS is not able to change its behaviour if a policy or a rule has not been added to the system. Moreover, the functions of each component of the PE will not change during execution. The PS is not capable of generating PRs by itself, while it is running.

From the point of view of the end-user, which interacts with smart applications or PSs, it will seem that the system is able to adapt its behaviour, in function of certain parameters. For example, if a user would be driving, the PS integrated in the mobile device would sense the high speed due to its context-aware property and would make changes at the level of the screen of the mobile device. This changes could include restricting the normal interface of the application, blocking the user's access to the interface or changing the size of the buttons and other elements to make them bigger and more noticeable.

3.3.4 Collaboration

Due to its design and the ability of communicating between them, PSs offer the possibility of active collaboration. But what is active collaboration and how does it work between PSs? This property refers to a shared effort by a group of PSs to individually contribute for solving a task. An efficient rule-based strategy, named *Global Proactive Scenarios*, offer PSs the possibility to collaborate and to take advantage of the abundance of information. Collaboration, in this context, involves a particular form of distributed computing. More precisely, it offers a data exchange mechanism through which PSs can have access to a more global knowledge and to a larger set of actions. It is not breaking down the computing tasks of a PS and distribute or assign them to other PSs.

Collaboration involves more than just communication or interaction between PSs that are running on different devices. It involves the cooperation of these systems for working together for finding solutions to various tasks or for checking certain information. Figure 3.3 shows a point of view where the authors [3] put collaboration as the layer on top of communication, coordi-

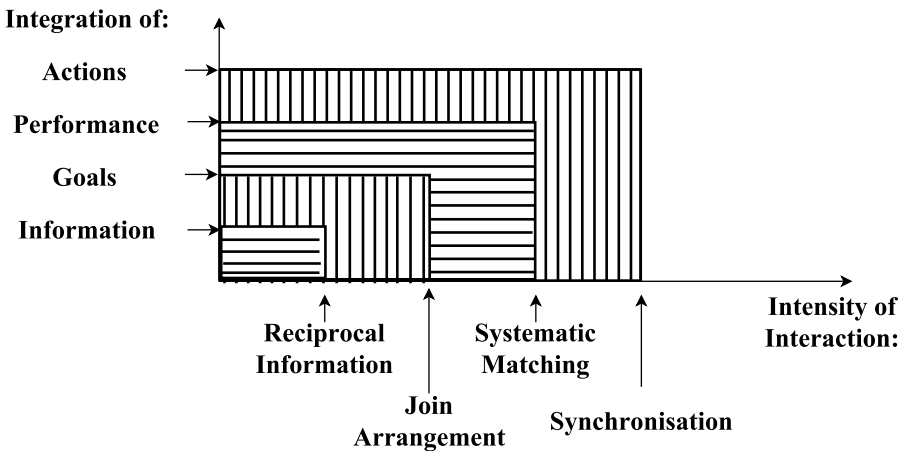


Figure 3.3: Different layers of interaction [3].

nation and cooperation. Communication handles only information which is sent from one system to another, coordination refers to joint arrangement, cooperation goes above and offers systematic matching, while cooperation add synchronisation to complete the degree of interaction between PSs.

3.4 Implementing the Proposed Model

Two main implementations of the PE were created for validating the model and for using it to create real-world applications: one for servers and one for mobile devices. Java is selected to be the programming language for developing the PE because it has O-O concepts like abstraction, encapsulation, polymorphism and inheritance, because it is *platform independent*, being able to run on any OS with a Java Virtual Machine, because there is a big number of devices running Java, e.g., mobile devices, desktop Personal Computers (PCs) and servers, and because Java works with a big collection of additional open source libraries.

3.4.1 Middleware Model for Server Platforms

A server application is a must have in the case of working with multiple distributed systems. Exceptions to this rule include *peer-to-peer* systems where tasks are distributed between peers. This type of architecture excludes the need for a central coordination point or a server. Multiple sorts of tools are available for building server-side applications in Java.

Implementing the PE on a server as a Java application was chosen because of the compatibility with other platforms, i.e., with Android, the Java-based OS for mobile devices. On the server side, because of its hardware structure and because of the powerful OSs that operate on top of it, it is

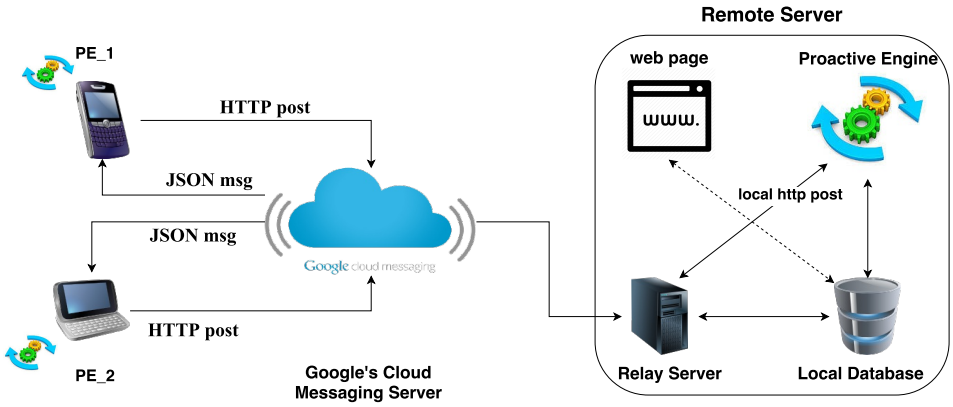


Figure 3.4: The server-side architecture.

much easier to develop applications which can perform tasks in the background of the OS.

All the components of the PE, i.e., the RE, the QM, the CE and the NM, are included in the application. The local database is developed using MySQL [84] and the data persistence mechanism with Hibernate ORM [85]. The CE, working with another application, called the *Relay Server*, ensures the sending of messages to the other PSs. As the Relay Server is not an internal part of the PE, its description and functionality are provided in the subsection 3.4.3. Several external libraries such as Java API for JSON processing from Oracle [86] or the MySQL connector [87] were used to make the connection between the different parts of the PS. A more general overview of the server's structure is offered in figure 3.4.

3.4.2 Middleware Model for Mobile Platforms

Developing a PE for mobile devices came in response to the increasing popularity of smartphones, which are connected to the Internet most of the time. This allows people to stay online everywhere they go and to access online services remotely at any time they want.

The PE was designed to run constantly in the background in order to allow the user to interact with different applications. On Android this can be achieved with the help of background services. However, these services cannot run constantly as they might be 'killed' or stopped by the OS if they use too many resources over an extended period of time. Therefore, the existing solution for desktop computers needed to be adapted in order to still allow the PE to execute PaSs. This was done using an *"Alarm Manager"* that activated the background service every F seconds and that executed one iteration of the RE. The Alarm Manager was triggered by the *onBoot*

event.

Similar to the middleware model for servers, saving the rules at every iteration of the PE was achieved through the Object Relational Mapping Lite (ORMLite) [88] package framework for Android. Additionally, the same procedure applies for saving the sent and received JSON messages. The ORMLite package is a lightweight package for persisting Java objects to SQL databases. The procedure for adding notifications was also simplified, e.g., if the developer needs to add a new type of notification, he/she only has to create the corresponding file with the correct annotations and the PE will take care of creating the equivalent tables and of the saving the related entries. Notifications use the internal notification system provided by Android and can be further customised and modified by the applications developers if needed.

Related work

This section presents the state-of-the-art related to RBSs and to collaborative middlewares for mobile devices, published in [79]. Multiple collaborative middleware systems for mobile devices are available on the market, e.g., [89], [90] and [91]. They differ by their type: event-based architectures, e.g., for supporting location aware-mobile applications [92], or publisher-subscriber architectures, e.g., for [93] systems. These frameworks offer communication services for the mobile devices, which use them for performing collaborative tasks. Our framework does not only achieve message passing from one device to another but also permits more complex actions like remote rule activations, parallel commands or complex reasoning algorithms.

Existing RBSs for mobile devices [94] [95] [96] currently solve only simple tasks and do not provide methods for achieving more complex tasks like distributed reasoning, task distribution, data sharing, acquiring global context information or/and collaborative filtering. And most of all, these systems do not have an efficient algorithm for handling global information, expect, in some cases, the logic that is set on the server side.

IF THIS THEN THAT [96] is a mobile application that realises automation for small tasks between Internet-connected services. The user can write simple rules, also called recipes, in order to achieve different goals like adding the photo of a user to the cloud-based archive if the user has been tagged in that particular photo on Facebook [97]. These rules, however, are just simple conditional statements. HeaRT [94], a lightweight rule-based inference engine designed for mobile devices, was used in [98] for providing simple tasks like online reasoning, part of a bigger plan to develop context-aware mobile applications. The rules that are written for this engine can achieve local reasoning only based on the internal sensors of a mobile device and do not explore the possibility of having multiple engines performing global reasoning. Minimal Rule Engine (MiRE) [23], a context-aware processing

engine, was implemented in order to obtain an engine capable of processing rules on mobile devices. However, the rules are written in XML and, due to their structure, are not capable of integrating more complex logics. These approaches, i.e., [94], [95] and [96], try to address the growing demand of using rule-based tools on mobile platforms and manage to do it but for a very narrow type of applications.

3.4.3 Communication architecture for Proactive Engines

Several technologies are available for making possible communication between mobile devices and/or servers, each with their own advantages and disadvantages. After analysing the alternatives, the chosen solution, which suits mobile-based and server-based PSs is the GCM server together with a **relay server**. There are another few alternatives on the market like Parse [99], PubNub [100] or UrbanAirship [101]. Nevertheless, while these services make it easier to develop push notifications for iOS and Android, they are still GCM-based solutions. One alternative, which does not use GCM at all, is Pushy [102]. However, Pushy's architecture is very similar to GCM, maintaining its own background socket connection, to receive push notifications [103].

Google Cloud Messaging

GCM for Android is a service provided by Google, which allows the sending and reception of data between a server and Android-based smartphones. The GCM service handles the delivery process of the messages, meaning that it takes care that the messages are delivered to the correct device. In the case where a message is sent to an offline device, GCM temporarily stores the message until the receiving device comes back online.

The Relay Server

The Relay Server is an entity necessary for handling the messages between different RBPSs. The GCM server is not enough to provide all the functionality for passing messages between devices. The device-to-device communication implementation is illustrated in figures 3.5 and 3.6. The registration process of one device is shown in figure 3.5, while figure 3.6 is showing the communication between already registered devices. The devices first have to register to the GCM server and get a registration identifier. Then, they communicate this identifier along with a *username* through a simple HTTP request to the **relay server**, which stores them in a database. If an already registered device now wants to communicate with another registered device, it sends its message along with the receiving device's username to the *relay server*, which retrieves the correct identifier and pushes the message along

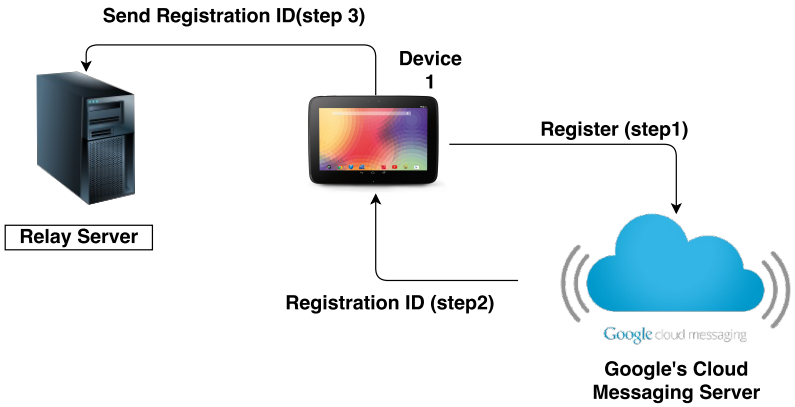


Figure 3.5: The registration process for one device, passing through GCM Server.

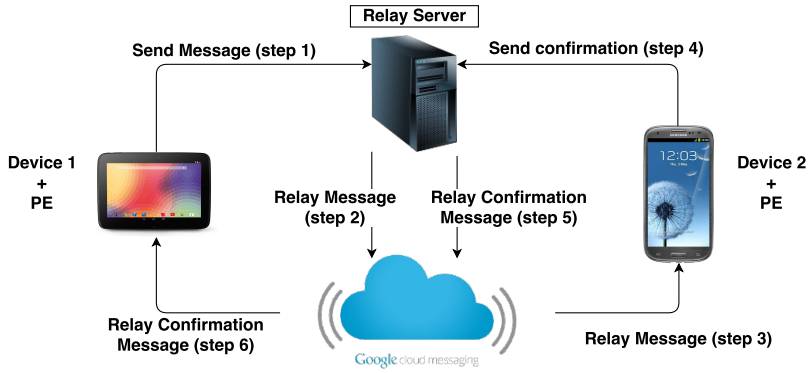


Figure 3.6: The communication process between 2 registered devices.

with the identifier to the GCM, which then takes care of the delivery of the message.

Message Structure

The messages exchanged between different RBPSs need to follow the same standards. The messages are encoded in the JSON protocol and have different attributes depending on the type of message. The only attribute that is common to all message types is the instruction attribute that allows the receiving RBPS to parse incoming messages correctly. There are currently two types of messages, the *activate rule message* type and the *confirmation message* type. The *activate rule message* type contains the following attributes:*instruction*,*msgID*, *senderID*, *receiver ID*, *ruleName*, *parameterList*.

The *msgID* along with the *senderID* allows the receiving RBPS to keep track of already received messages. This will be explained in detail in the next subsection. The *ruleName* and *parameterList* attributes are the core

```
{
    "instruction":"activate rule",
    "msgId":"messageID",
    "senderID":"ID",
    "receiverID":["registrationID"],
    "ruleName":"R004",
    "PARAMETER_TYPES":["param1", ... ,"paramx"],
    "PARAMETER_VALUES":["val1", ... ,"valx"]
}
```

Figure 3.7: Example of a JSON message exchanged between 2 Proactive Engines, which contains a command to activate a Proactive Rule.

attributes of the communication process. They allow the creation of a PR on another RBPS. The *ruleName* attribute contains the name of the PR that will be created on the receiving device and the *parameterList* contains the parameters necessary to create this PR dynamically. After the PR is created, it will be added to the next Queue of the PE. An example of a message exchanged between RBPSs, which contains a command to activate a PR, can be seen in figure 3.7. The confirmation message only contains three attributes: *instruction*, *msgID* and *receiverID*, where *msgID* is the ID of the message whose delivery is confirmed.

Error Handling

In a distributed environment, messages are not guaranteed to arrive, as they can be lost along the communication process. In order to prevent the loss of messages, RBPSs keep track of sent and received messages, including their senders and receivers, by saving them in a table in the local database. After sending a message, the PE saves it to the database. If no confirmation message is received within a given time period, the message is sent again. The time period varies depending on the priority level of the message, which can be set when sending the message. It is also possible to set no priority level at all so that the message does not have to be confirmed. This is quite useful for doing message broadcasts in order to find other devices with the same preferences, where it is not important if every device receives the broadcast. Upon receiving the confirmation for a specific message, this message is deleted from the database.

The procedure after receiving a message is very similar to the sending procedure. The message is saved in a designated table in the database and a confirmation message is sent over to the sender of the message. In the case

the confirmation message does not arrive, the sending device resends the message and as the receiving device has saved the message in its database, it ignores the second message. The stored messages are deleted after a fixed period of time. In order to take care of resending and ignoring messages, there are two PRs constantly running on the engine: one rule that takes care of the messages that were sent and one rule that takes care of the received messages.

Limitations of GCM and Workarounds

GCM has two big disadvantages, a messages size limit and a limit on the number of devices a message can be simultaneously sent to. The size limit on the messages is of 4 Kilobytes. The messages designed for RBPSs can be delivered using the initial method proposed by the framework as they are smaller than 4 Kilobytes. If the size limit is exceeded, the relay server will store the complete message in the local database and just send a small message to the receiving device to notify it that a message is available. The device will then download the message directly from the server and add the PR to the Queue. The second restriction of the GCM is that a message can only be sent to 1000 devices simultaneously. This limit can be reached by the proposed model only if a broadcast is sent to all devices and the number of devices is bigger than 1000. In this case, the server just splits the list of all devices into groups of 1000 and pushes the same message to the GCM for every group.

3.4.4 Performance analysis of Proactive Engines

Mobile devices include a multitude of smartphones and tablets produced by different companies, with different specifications. It is thus necessary to investigate how PEs perform on different devices and what are the factors that influence the overall performance of PaSs.

Methodology

The method for analysing the performance of the PEs involved measuring the time between two consecutive *iterations*. An *iteration*, as described in section 3.2.1, is an executing instance composed of a set of PRs and is measured in terms of duration. Ten different sets of PRs were considered for the tests: starting from small sets of rules, containing 100 rules, until larger sets of rules, with 1000 rules. For these tests, an iteration was composed of two main operations: the *execution phase* and the *saving phase*. During the *execution phase*, the RE runs each instruction, part of each PR, that contains actions. The actions can include acquiring data from the sensors or from the local database, sending notification to the users or even the generation of other rules for the next iteration. The *saving phase* was mainly used for

Table 3.1: Average time of an Iteration of the Proactive Engine on 3 different devices.

#rules/iteration		100		300		700		1000	
Smart phone	Iteration Time (ms)	436.7		672.5		1099.8		1294.2	
	Saving Time (ms)	329 (75%)	Execution Time (ms) 107.7 (25%)	475 (71%)	197.5 (29%)	671.4 (61%)	428.4 (39%)	772.2 (60%)	522 (40%)
Tablet	Iteration Time (ms)	130.5		233.2		383.9		495.5	
	Saving Time (ms)	108.6 (83%)	Execution Time (ms) 21.9 (17%)	175.4 (75%)	57.8 (25%)	275.4 (71%)	108.5 (29%)	352.4 (71%)	143.1 (29%)
Desktop Computer	Iteration Time (ms)	42.5		68.5		127.5		220	
	Saving Time (ms)	6.5 (15%)	Execution Time (ms) 36 (85%)	2 (3%)	66.5 (97%)	24.5 (19%)	103 (81%)	35.4 (16%)	184.6 (84%)

saving the set of rules that will be executed during the next iteration. A list of rules, together with their parameters are saved into the local database. This phase was created as a safety measure in case a crash would occur or the RE is stopped by internal factors. After a crash, when the RE restarts, it reads the last list of rules that was saved in the database and it will start executing them.

Multiple rounds of tests were performed, each round of tests containing 10 evaluations for each device. The execution time averages for all the tests were computed for obtaining more accurate values. Other applications and services running on the devices involved in the tests were not explicitly closed because the intention was to analyse the performance of the PEs in the same circumstances that a common user would be using his/her device.

Table 3.1 contains the averages of the total amount of time, in milliseconds, that one iteration required for running sets of 100, 300, 700 and 1000 PRs. The total time is divided furthermore into *saving time* and *execution time*, which are average values in milliseconds that represent the amount of time needed by the RE for the *saving phase* and for the *execution phase*. All the sets of rules contained clones of the same basic PR. This rule was designed specially to see how much time does the RE needs to save an instance of all the rules that will run at the next iteration. This explains the relatively low values for the *execution time*.

For example, running 100 PRs on the smartphone took, in average, 436 milliseconds for one iteration. The time necessary to save the rules for the next iteration took 329 milliseconds, representing 75% of the total amount of time of the iteration. This case is confirmed by the values obtained from running the same 100 PRs on the tablet, were the time for saving the rules for the next iteration took 83% of the total time of the iteration. The same results were obtained for *saving the rules in the* was also confirmed by the

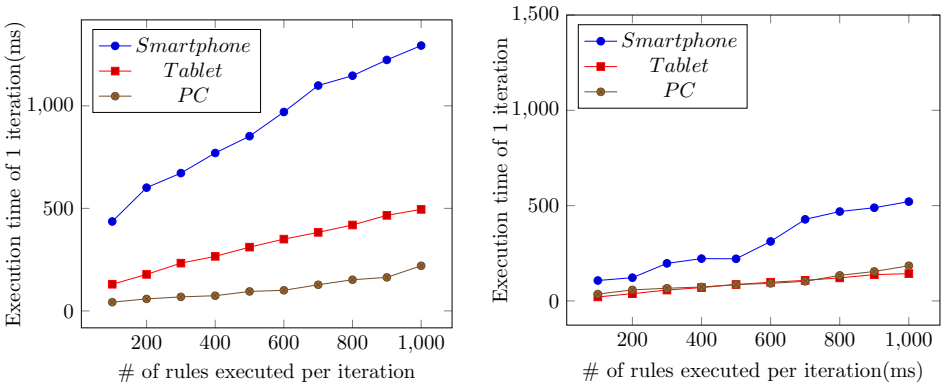
Device	Samsung Galaxy S3	Nexus	PC
Model	GT-I9300	10	
OS	Android 4.3	Android 4.4.4	Window7 64-bit
Baseband	I9300 XXUGNA8	KTU84P	
Compilation			
Kernel	3.0.31-2429075		
RAM	1 GB	2 GB	8 GB
Chipset	Exynos 4412 Quad	Exynos 5250	Intel 6 C200
CPU	Quad-core 1.4 GHz Cortex-A9	Dual-core 1.7 GHz Cortex A15	Intel Core i7-2600 3.4 GHz
GPU	Mali-400MP4	Mali-T604	
Sensors	Accelerometer, gyro, proximity, compass, barometer	Accelerometer, gyro, proximity, compass, barometer	None
WLAN	Wi-Fi 802.11, a/b/g/n, dual-band, Wi-Fi Direct, DLNA, DLNA, Wi-Fi hotspot	Wi-Fi 802.11, a/b/g/n, dual-band, Wi-Fi Direct, DLNA, DLNA, Wi-Fi hotspot	None

Figure 3.8: Hardware and software specifications of the devices used in the experiments.

values obtained from running 300, 700 and 1000 rules. If for 100 rules it took 75% of the total iteration time for saving the rules in the database, for 1000 rules the percentage decreased to 60% of the total iteration time and stabilised around that value.

Significant differences for running an instance with 1000 rules, between the smartphone, the tablet and the desktop computer, appear in the last column of table 3.1. If, in the desktop computer's case, the total time for finishing the execution of one iteration took 220 milliseconds, which is quite fast, the same operation took almost twice more on the tablet and approximately 5 times more on the smartphone.

The results in table 3.1 also meet the general expectations in terms of computing capabilities of the involved systems. For instance, executing one iteration with 100 rules required approximately 4 times more time on the tablet than on the PC and more than 8 times more on the smartphone that on the desktop computer. The difference did not change much when executing 300, 700 and 1000 rules. This is mainly due to the particular hardware



(a) Total time of one iteration on all the devices when saving the rules at the end of the iteration

(b) Iteration time on different devices without saving the rules at the end of the iteration

Figure 3.9: Different iterations times on all 3 different devices.

configuration of each system as illustrated in figure 3.8. The smartphone was equipped with 1 Gigabyte of RAM and a quad-core 1.4 Gigahertz processor, while the tablet, which achieved better performance results, was equipped with 2 Gigabyte of RAM and a dual-core 1.7 Gigahertz processor. The desktop computer had the best configuration, i.e., 8 GB of RAM and an i7 processor with 4 physical cores capable of operating at frequencies up to 3.4 GHz, and so, it had the best results.

Saving rules on the database took a lot of time in comparison with the operation of executing the rules. A possible cause is the ORMLite library used for data storage on Android. On the other hand, the data storage on the desktop computer was done with the help of MySQL [84] and Hibernate ORM [85] frameworks and needed far less time than for executing rules. A solution, which can be applied in production for avoiding time losses, is to remove the feature of saving rules at each iteration and to set up a *saving phase* for the RE only once, e.g., when the shut-down event appears on the mobile devices. Another important conclusion is that running more than 1000 PRs on the smartphone or tablet is a time-consuming process and, at this moment, these devices are able to run only limited sets of PRs, while still being able to provide real-time services to their users. This amount of rules is directly related to the computing capabilities of each device and the libraries used to ensure different functionalities of the PE. However, for the mobile applications that use PaSs on top of the PE, sets of up to 100 rules should be enough to achieve the desired goals.

Figure 3.9a and figure 3.9b illustrate better the differences of running one iteration between the different devices. They include the average results of all the 10 sets of PRs that were performed on all 3 devices. The

scale for the execution time of one iteration was kept on purpose to show the major difference between amount of time needed for one iteration with the saving phase and without the saving phase. In figure 3.9a it can be noticed relatively big fluctuations in the execution time of one iteration on the smartphone. It is not linear like in the tablet's case and the desktop computer's case. If, until running iteration of 800 rules, the time increased in an expected manner, afterwards, it started to rise up quickly. It means that for iterations with more than 800 rules the smartphone is starting to consider the PE quite heavy in terms of the processing resources needed and this can slow down other applications that need to access the same resources. In figure 3.9b, however, the distribution of execution time without saving the rules in the database is increasing constantly, meaning that the duration time can be anticipated for various number of rules.

Communication Tests

Other series of tests were conducted between the smartphone and the tablet to measure the duration of their communication. It was measured how much time it took for sending and receiving a message, meaning the first 3 steps of the communication process, between the two already registered devices. After 10 tests, it took, in average, 751.3 milliseconds for performing the following operations: the creation of the message on device 1, sending the messages to the relay server, sending the message to the GCM server and then, receiving the message on device 2. The message used in the experiments was the same one as the one presented in figure 3.7. It contained specific instructions for activating a rule on device 2. The Relay Server was running on the desktop computer with the software and hardware configurations shown in figure 3.8. The entire communication process also depends on external factors like the network bandwidth and latency, and on the response time of the GCM server, which is also not part of our system and cannot be controlled.

Battery Consumption on the Mobile Devices

Mobile devices obtain the necessary energy for performing complex operations from their batteries, which implies that analysing power consumption on these devices is very important. Our approach to calculate energy consumption was to measure the battery level on the smartphone using Android's internal system functions calls [104].

Benchmarks for Battery Consumption

Three types of benchmarks were carried out. The first one was designed for testing only the PE alone, which was executing sets of 100 PRs each 30 seconds. The PE used in these tests was designed to simulated rules that

Table 3.2: Average Battery Consumption.

Applications	Average
`Proactive Engine`	1.5%
`Wakelock application`	4.3%
`Wakelock application + Proactive Engine`	5.4%

would be used in different real-world applications. In the second benchmark, the interaction of a user with the screen of his/her smartphone by using a wake-lock application that woke up the screen of the device, at 100% brightness, for a total of 18 minutes per hour was simulated. And, in the last benchmark, both the PE and the wake-lock application were running simultaneously on the smartphone.

For all three benchmarks, 10 executions of 1 hour each were performed in order to compute their averages. During the tests the smartphone's Global Positioning System (GPS), wireless connection, Bluetooth and the other mobile data connections were turned off.

Results of the Benchmarks

The results, shown in table 3.2, indicate first that executing Proactive Rules on a running PE for mobile devices, during 1 hour, takes only 1.5% of the total amount of the battery of the smartphone and second that a standard application consumes significantly more energy than a PE. The results of the third benchmark confirm the difference obtained between the first two benchmarks.

3.4.5 Conclusion and Discussions

These tests were necessary for estimating the optimal number of PRs, which can be executed in one iteration by the PE on a smartphone, on a tablet and on a desktop computer, without having big delays. This aspect, or the *response time*, is very important when we want to design applications that can execute PaSs on each device. The energy consumption analysis on the smartphone indicates that PE does not consume much of the battery, which is a key aspect when developing mobile applications. The experiments indicate that PEs can be successfully integrated into mobile devices, that the model is able to run on different computing devices and that the processes of the proposed model are very efficient from a computational point of view and do not affect the overall performance of a device. Moreover, different performance evaluations of PEs on smartphones, tablets and desktop computers were conducted and compared. A solution was provided for improving significantly the execution time of PRs and implicitly of PaSs by

saving them when *shut-down events* occur instead of saving them during each iteration of the PE.

Chapter 4

Proactive Scenarios

4.1 Linking and Grouping Rules

A ***Proactive Scenario*** (PaS) is a set of predefined rules with several particular properties. The rules that compose the scenario describe a particular case and take according actions, each rule making an incremental progress towards a situation-level goal. It can be seen like a collection of rules that implies a sequence of events and actions. Proactive Scenarios come as a response to one of the major challenges in rule-based systems, i.e., how to organise and link rules in order to obtain the desired output, presented in chapter 2.1.2. PaSs are not just a particular collection of rules but they are situation-solving modules, where rules are linked in a particular way to reach a certain goal. The traditional way of creating rules through dialog, with the help of an expert, can block the development process as it can get very slow [105]. Many rule classification systems were developed in the last thirty years like decision tree-based systems, which include C4.5 rules [106], or covering-based systems, including the AQ15 algorithm [107], or association rule-based systems, with methods like CMAR [108] or CBA [109]. However, association rule-based systems are used intensively in the field of data mining, decision tree-based systems in the field of decision support systems and covering-based systems in fields such as expert systems or information systems. For these systems, a particular collection of such rules doesn't necessarily imply a sequence. Also, as the rules used in these systems are expressions following the basic structure of the IF-THEN rule, it is not difficult to chain these rules based on their input and output conditions.

PaSs are good knowledge representation and reasoning mechanisms because they give the user the possibility of specifying exactly which PR should be executed in which order and under which kind of special conditions. From the point of view of their design, PaSs are backward chaining mechanisms, as they start from the initial goals of the system and they end by validating or not that goals. From an execution point of view, or the way the PE is

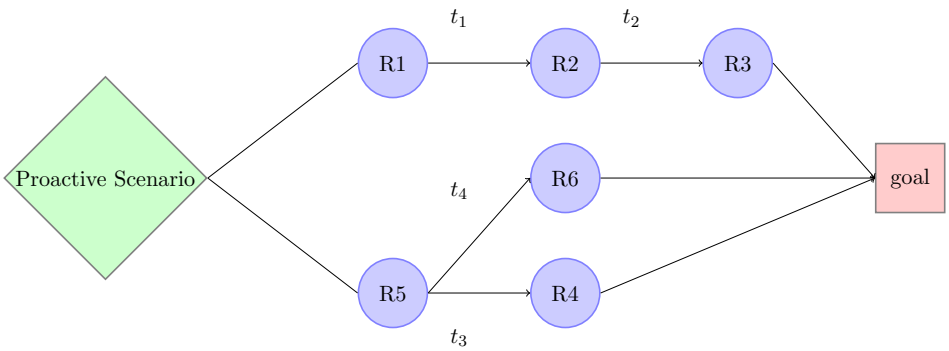


Figure 4.1: The *decision tree* structure of a Proactive Scenario.

firing PRs, the PaSs are a forward chaining mechanism as they work their way towards reaching a goal.

Several methods exist for structural representation of rules such as *decision tables*, *decision trees*, *binary decision diagrams* or *tabular systems* [5, p. 97]. These methods are used specially for propositional logic. Table 4.1 contains an example of a classical form of decision table. This is an easy way of representing a simple rule, where condition c_{ii} indicates which condition of $rule_i$ to be checked and action a_{ii} indicates which action of $rule_i$ to be executed. But decision tables cannot be used because of the structure of the PaSs, which are composed of more complex rules, i.e., PRs. Therefore, the simplest method is to use a **decision tree** structure for expressing and creating PaSs, as shown in figure 4.1. The PaS is decomposed in function of the different actions that need to be executed. The PRs are linked based on the set of defined actions. The PaS in figure 4.1 is composed of 6 PRs. PRs 1, 2 and 3 form a chain because each rule contains different conditions that have to be fulfilled before triggering the next rule or achieving the final goal. These conditions do not necessary become true after an iteration of the PE. The time between the activation of two rules is normally one iteration, excepting the case when something happens to the PE and it is stopped. But the time to activate a PR, i.e., the time to satisfy all the conditions in the *activation guards* phase, differs from PaS to PaS.

4.1.1 Local Proactive Scenarios

Local Proactive Scenarios (LPaSs) are PaSs that run on a single system and perform only tasks or actions that happen on the system where they are executed. LPaSs keep the *decision tree* structure of PaS and the possibility of having various distribution of PRs, depending on the number of actions and what functionalities they have to address.

LPaSs do not have constraints in terms of PRs or actions that compose them. They can be activated by GPaSs or by other LPaSs. The difference

Table 4.1: A classical decision table with vertical rules [5, p. 110].

	rule1	**rule2**	**...**	**ruleX**
condition1	c_{11}	c_{12}	...	c_{1x}
...	...	...		...
conditionZ	c_{z1}	c_{z2}	...	c_{zx}
action1	a_{11}	a_{12}	...	a_{1x}
...	...	...		...
actionZ	a_{z1}	a_{z2}	...	a_{zx}

is that the activation is done locally, when it is requested by a LPaS, or remotely, when it is requested by a GPaS. The downside of LPaSs is that they are limited from getting information from multiple external sources, from performing collaborative actions and from adapting to other computing environments. They are not involved in a data exchange process with software systems performing on other devices. They can collaborate, at most, with applications running locally on the same device.

4.1.2 Global Proactive Scenarios

In the current settings, not all the devices can natively communicate with other devices. This offers no support for collaborative tasks. This subsection proposes a modal that will allow PSs to collaborate by providing a communication protocol based on PRs. Then, later on, in the second part of this thesis, real-world examples are given with how this collaboration between PSs works in practice.

Global Proactive Scenarios (GPaSs) are information sharing mechanisms with the following characteristics: they can detect unexpected events, they dynamically collect information from devices with an integrated PE, they provide strategies for cooperative reasoning and they support collective decision making processes. In contrast with traditional distributed systems, GPaSs provide the possibility of achieving distributed reasoning between PEs by the concurrent execution of various PaSs.

The main idea of GPaSs is to look for the minimal set of relevant information to perform a local or collective task. And this operation has to be done in an efficient matter as there is abundant information coming from all the sensors of a ubiquitous environment. A balance needs to exist between the proportion of retrieved data and the precision of processed data. In practice, it is hard to achieve this balance. To address this problem, GPaSs activate local PRs on devices that have an integrated PEs for retrieving and processing data coming from their embedded sensors. After it is processed, only the relevant data is exchanged between the devices.

GPaSs imply several PEs working together for solving a bigger puzzle. As the communication between PEs functions by exchanging JSON messages

which activate PRs, several PRs have to be loaded into the memory on the receiving device. And, as any device with a PE is assumed to be able to start a GPaS, the PRs have to be present on both, the sending and the receiving devices.

The design of GPaSs is aiming at numerous objectives. With the help of GPaSs, the local information available on a smart device is distributed to another PS, context is captured at various levels, expected situations are detected and treated accordingly, resources shared and general knowledge is gathered for achieving a ubiquitous intelligence capable of taking smart decisions. From the front user's point of view, only the effects of GPaSs are noticeable and not how they are activated or how they manage to exchange information. Thus, it will not be necessary for the users that interact with the smart applications to have advanced technical skills for using smart devices, instead they will be able to focus more on their tasks and achieving their goals.

GPaSs enable devices to perform a wide range of group tasks and to benefit from a collective intelligence, which is far more effective than the knowledge of each single device. They can change the state of the PSs as well as being affected by the system's state changes. The effects of GPaSs can be measured by developing case studies, where the performance of the PEs and frequency of triggering PRs can be evaluated.

Two main ways of triggering GPaSs exist, i.e., either the occurrence of a foreseen event or the interactions of users with the screens of their devices. At a technical level, foreseen events are detected by PRs, which are acting as background services with a clear purpose of monitoring the environment of each device. For example, running low on battery on a smartphone is a foreseen event. If the battery level drops under a certain percentage, specific action are taken, e.g. the luminosity of the screen of the smartphone is reduced to a minimum level. The user's input can be divided into explicit or implicit instructions. Explicit instructions refer to precise and clear input from the user, while implicit instructions point out commands which are not directly expressed but they are implied. For instance, on mobile devices, an explicit command can be when a user wants to share his/her location with a group of friends by pressing a certain button on the screen of his/her device.

One of the biggest advantage of GPaSs is that they can work without having to implement any server-side strategy, which would contain all the logic for exchanging the information between different devices. However, an additional server-side strategy could be added, on top of the existing GPaSs, for ensuring a better management in case the GPaSs fails. This method requires an additional development step and is recommended only as a backup feature in case the communication between PEs does not end successfully.

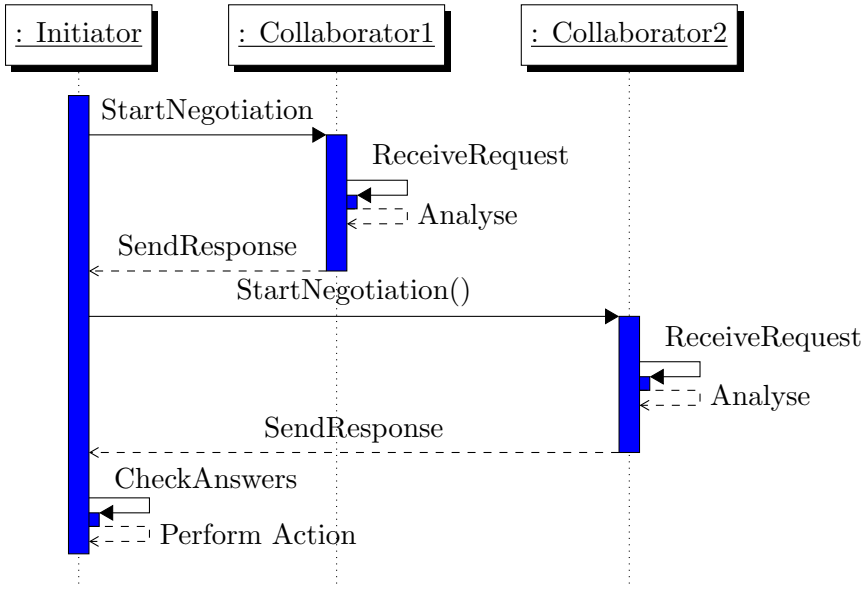


Figure 4.2: A sequence diagram with the collaboration mechanism of a Global Proactive Scenario.

4.1.3 Negotiation protocol of Global Proactive Scenarios

A simple GPaSs with only one round of negotiation between 3 devices equipped with a PS is presented in figure 4.2. It is simple because the PS on the initiating devices is asking for information from the other devices, i.e., collaborator1 and collaborator2, in order to solve a task. After this information is received, the initiating device has enough information to do the required actions. A more complex GPaS involves multiple rounds of collaboration and is taking into account the unsuccessful sent or received messages. An unsuccessful action of sending or receiving a JSON message between PSs can be cause by multiple factors like the network latency where they operate, an unconnected devices, etc. These factors are well-known in the networking field.

The device that starts the GPaS is a called an *initiator*. Any PS can be the initiator of a GPaS. Additionally, any device can have more than one GPaS that is being executed. The PSs that are involved in the collaboration are called *collaborators*. As PSs communicate through messages which activate PRs, each collaborator has to have the according PR, which is sent as a parameter in the JSON message, available in the rule base.

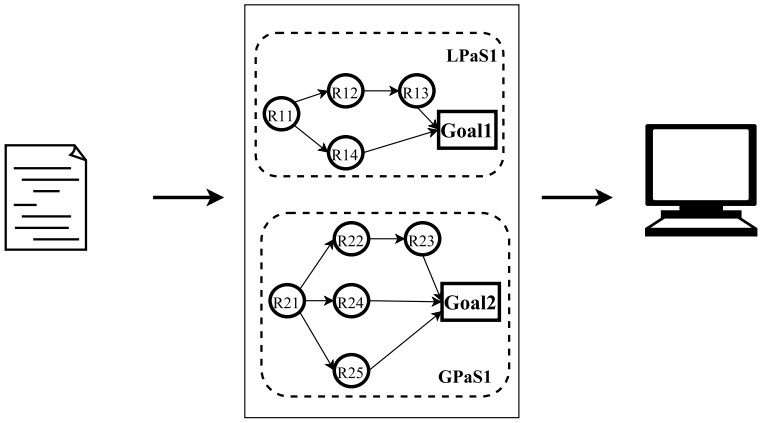


Figure 4.3: Designing Rule-based Proactive Systems.

4.2 Rule-based Proactive Systems Design

As seen in figure 4.3, RBPSs should follow three main development phases: *the conceptual design*, *the logical design* and *the physical design*. This top-down structured method represents a standardised and organised approach for analysing and modelling a set of requirements. The advantage of this method is that the developer/engineer mainly needs to discover the appropriated goals and functions and to translate them to PaSs. Additional work can include the development of the GUI, if the system is using one, or the development of database tables and structures, if the system needs to store, for example, its state. RBPS design phases have the advantage of being modularised. Another advantage is that, if an additional goal is needed, it can easily be added to set of goals of the PSs due to its modular design. The same applies for creating additional PaSs and for adding additional PRs.

4.2.1 The Conceptual Design Phase

This stage is made for breaking-down the application/software system into multiple goals. It is up to the developer to determine primary goals and secondary goals of the application. These goals are constructed with a strong reference to the requirements that where created. Requirements specification is tough another particular topic that is not included in this work. Nevertheless, requirements are important as they represent the source for specifying the goals.

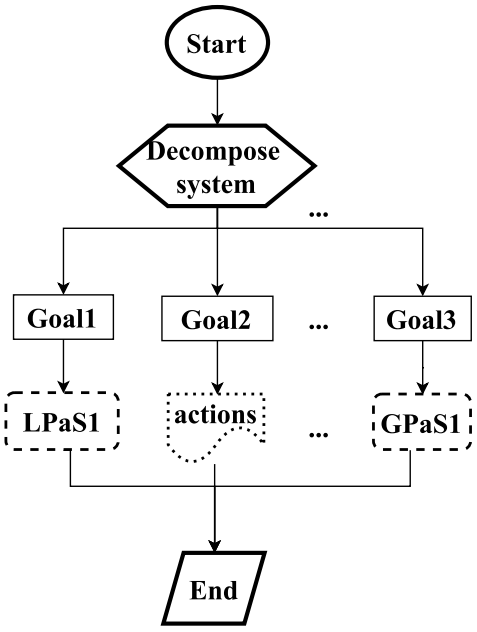


Figure 4.4: The Conceptual Design Phase of the design process of Rule-based Proactive Systems.

The *conceptual design* stage does not only focus on obtaining a list of goals but is also concentrating on identifying important entities, their characteristics and the relations between them, including database tables and elements that need to be stored, and functions that the application is able to perform. Also important, the goals of the PS are related to the properties that the system has to have. For example, if the general purpose of the system is to detect various situation changes then the PS needs to be *context-aware*. This cannot be a particular goal, but a more general goal. Instead, this property should be described in a particular situation, when it is needed in a much narrower sense.

Figure 4.4 is an illustration of the method proposed to decompose applications/software systems according to their goals and, then, based on these goals, into PaSs. The case where PaSs are present for a particular situation signifies the option to automate a procedure or a set of procedures. Not all the goals of a systems require PR.

4.2.2 The Logical Design Phase

The *logical design* stage is the stage where the developer/engineer, after identifying the main goals and functions of the systems, is developing PaS and how they achieve the proposed goals of the PS. The developer has to choose the correct type of PaS, i.e., LPaSs for taking care of local actions

and GPaSs for handling situations where collaboration is needed between various PSs. Then, it is important to establish how should the PRs be linked and grouped in order to build valid PaSs.

An efficient logical design phase is achieved when the number of iterations needed to achieve the goals determined in the *conceptual design* stage is reduced to a minimum. This is also a stage that depends on the experience and the set of skills of the designer. Linking and grouping PRs into PaSs is a process that relies on the judgement of the developer. He/she is in charge of managing the interdependencies of PaSs and to see if, for example, they do not produce conflicts in the actions that they perform.

4.2.3 The Physical Design Phase

During this last phase, all the previous PaSs should be implemented and tested by using a PE. This phase is one of the most important as it verifies if the decisions of grouping PRs into certain PaSs was done in an efficient way. The last design phase consists of managing the different functions of the PS. If, for example, the PS is not behaving according to its initial goals because maybe too many instructions were inserted into a PR or one of the actions of the PR is taking more time than an iteration of the PE, then corresponding measures can be taken, e.g., redistributing the actions of a rule or creating additional rules as part of the PaSs or increasing the duration of an iteration of the PE.

4.3 Distributed knowledge

Collaboration has many forms. However, in the area of RBSs, collaboration is process which involved the exchange of rules. It does not refer to collaboration in the sense used in the field of computer-supported cooperative work (CSCW), where the collaborative activities focus more on interaction including synchronous document editing, video-conferencing or real-time groupware. This subsection contains an overview of the role played by rules and rule-based systems in distributed systems, a task already which was already started by some authors [110].

4.3.1 Parallel Rule-based Systems

The initial success of RBSs in the eighties and nineties pushed forward this technology to traditional research fields like distributed and parallel computing. Firing rule in parallel was already taken into account for both FC [111] and BC production systems [112]. If this approach is used it will increase the execution time of the RBS but it will also create consistency problems in the conditions and actions of the rules.

```
ACLMessage request = new ACLMessage(ACLMessage.REQUEST);
request.setProtocol(FIPANames.InteractionProtocols.FIPA_REQUEST);
request.addReceiver(new AID(receiver, AID.ISLOCALNAME));
myAgent.addBehaviour(
    new AchieveREInitiator(myAgent, request) {
      protected void handleInform(ACLMessage inform) {
        System.out.println(Protocol finished. Rational Effect achieved.
            Received the following message: +inform);
      }
    }
);
```

(a) A FIPA-Request initiator behaviour

```
MessageTemplate mt = AchieveREResponder. createMessageTemplate(
    FIPANames.InteractionProtocols. FIPA_REQUEST);
myAgent.addBehaviour( new AchieveREResponder(myAgent, mt) {
  protected ACLMessage prepareResultNotification(ACLMessage request,
      ACLMessage
response) {
    System.out.println(Responder has received this message:+request);
    ACLMessage informDone = request.createReply();
    informDone.setPerformative(ACLMessage.INFORM);
    informDone.setContent(inform done);
    return informDone;
  }
});
```

(b) A FIPA-Request responder behaviour

Figure 4.5: FIPA-Request responder and initiator behaviour [4].

The model proposed in chapter 3 is designed to fire PaSs in parallel and is capable of also firing PRs in parallel, depending on the implementation. As PRs are written in Java and each instance of the rule is an object, the possibility of firing multiple PRs in parallel can be solved with the help of threads. However, in cases where the PS has tens or hundreds of rules in the CQ, this could represent a major problem. A solution would be to limit the number of threads created and assign these rule to the NQ. But, this method would not guarantee a correct flow in executing PaSs as many iteration would be execute by the PE until certain PaSs would achieve their goals.

4.3.2 Distributed Rule-based Systems

Distributed computing for rule-based systems is mainly used in the field of *multi-agents*, where agents use a rule-based implementation. Multi-agents in production systems are able to communicate asynchronously, like MAGSY agents for example [113], as well as being able to provide services to other agents. In this case, distributed computing, refers more to the interaction of agents, which can exchange messages. The content of this messages is used for changing/updating the facts of another agent or, if a service is invoked by a remote agent, it can change both facts and rules of the targeted agent. Another widely-used type of agents, i.e., JADE agents [114], can send messages to other agents using Jess, a shell written in Java, for interacting with them. An example of FIPA ACL message used in the interaction by an initiator and its responder are illustrated in figures 4.5a and 4.5b.

Peer-to-Peer Systems

Rule-based distributed computing is applied also to Peer-to-peer (P2P) systems which have multiple applications in domains like ubiquitous computing, distributed content management and distributed processing. The P2P systems do not only communicate but also can collaborate by exchanging resources and providing services to each other. Examples of P2P rule-based systems include distributed database management systems [115] that maintains and manages data across multiple systems spread on different networks or sensor-based platforms for ubiquitous environments [116] that abstracts the passage to sensor data using an inference mechanism based on rules. Even though if P2P rule-based systems provide the infrastructure to avoid all the configuration needed for using a client-server approach, the negotiation protocol can become very slow and complicated.

Part II

Application Development and Case Studies

In order to illustrate the potential of the model presented in chapters 3 and 4, three different case studies are proposed. The main purpose of these case studies is to focus at first on providing relevant answers to **RQ5**. The upcoming case studies, each representing a chapter of its own, are also provided to better understand how would the proposed computing model be applied in real-world situations.

The following chapters provide validation examples of PaSs, addressing **RQ4** and examine the properties of RBPSs, which is the core subject of **RQ2**. These properties are not studied only from a theoretical point of view but they are analysed in a real-case scenario according to the particular situation of each case study. The ability of capturing changes in context is shown in each example, while the capacity of anticipating various situations is underlined in each PaS.

The case studies are proposed in an incremental way of complexity, starting with the most basic example of them, presented in chapter 5, until the most complete and advanced example, presented in chapter 7. The first case study uses the first type of PaSs, i.e., LPaSs, to address the needs of a complex e-Learning system and does not involve the cooperation with additional systems.

Starting from chapter 6, the proposed examples indicate the necessity of having collaboration and communication mechanism to address issues in multiple research fields. This necessity is taken care of with the help of GPaSs. This solution does not replace LPaSs but completes them. Chapter 7 offers an application which includes both LPaSs and GPaSs at different levels, and shows how they can work together to provide an efficient solution in a complex situation. The complexity of each case study refers to functions carried out by the involved PSs and to the type and number of devices involved in the experiments. If, in the first case study, the PS is developed next to a single system running on a single platform, the other case studies include both mobile and server platforms and devices.

The case studies are validated by creating several prototype applications, which are implemented and tested. Different application domains like e-Health, e-Learning and e-Business were explored for underlying the broad range of areas of expertise where RBPSs can be successfully applied. However, the point is not to demonstrate that all the problems in these domains can be solved by using RBPSs. RBPSs address only a specific category of systems which can benefit from PaC.

Each case study is introduced with additional background information related to the tackled domain and to discussion on related work with respect to that particular research field. This additional information completes the background information and theoretical information presented in chapters 3 and 4.

Chapter 5

Application 1 - Online Social Communities

The first case study [117] involved developing multiple **LPaSs** to automatically create, organise and develop social groups inside a LMS, i.e., **Moodle** version 2.2.4, and to guide, inform and assist students though the whole process [118]. The **LPaSs** used in this study were an enhanced version of the *PaS* presented initially in [119].

This study did not only show how the behaviour of a LMS can be transformed, by employing PaC, but it tackled some other important aspects in the e-Learning field like the online participation, the level of engagement and the collaboration of students. Other interesting aspects were taking into account such as analysing OCoPs, a particular form of Communities of Practice (CoPs) developed and maintained using the Internet, from the point of view of their life cycle and of their activities. To achieve these goals, new ways of organising students into virtual communities with a clear purpose and tools for building and sharing knowledge were proposed.

LMSs had a major influence on how teachers taught and how learners learned over the past 20 years. They continue to represent a very hot topic in Information Technology (IT) as many universities and colleges across the world use LMSs for delivering educational content to their users. Recently, many people consider that learning to be more of a social process and many educators are using social learning platforms such as *Edmodo* or social networking software like Elgg for addressing the growing needs of their learners. With the increasing popularity of networking platforms like *Facebook*, *Twitter*, *Google Plus* or *LinkedIn*, there have been numerous attempts to use similar tools in education. The main arguments are that these tools would enhance the online collaborative learning process and that it will allow their users to build and maintain stronger relationships with one another. It is essential that LMSs either change their approach on how to deliver educational content to their users or they adapt to the newest trends by including

more social features such as internal ratings and reviews for courses, events and resources or a collaborative system which would allow students to build powerful learning networks.

But why integrate social features inside a LMS when the majority of students have a Facebook account? One answer could be that social platforms like Facebook have many problems with fake accounts. These accounts represented more than 5% of the total number of registered accounts [120]. This decreases the level of trust among users and discourages users to branch out from their circle of friends. Students used a unique account with their real names for logging into the local Moodle platform, for accessing their online course, for enrolling in their exams and for uploading their online assignments. This is one of the main reasons for choosing Moodle for this study. Another reason was that students do not possess the rights for adding resources and tools like wikis, forums and chats or other content into their courses as they have limited permissions. This only allowed students to be trained by their teachers, and limits their knowledge to the given materials. The final reason for using this platform was that it could easily be extended with the help of blocks and plug-ins, without modifying its source code. Platforms like Facebook or Twitter are an ideal place for social learning, which can be defined as learning by conversation, observation or by questioning. But social learning is more concentrated on the individual's needs and that is why we consider OCoPs to be more suited for collaborative work. They are more focused on enlarging and distributing the knowledge of a group, as opposed to a single person [121]. Inside a Online Community of Practice (OCoP), people are learning by collaborating, exchanging and sharing information. *Moodle* offers the possibility of grouping students inside a course, but this procedure has to be done manually by the teacher or by an administrator. This can easily be done if the number of participants is small, but it rapidly becomes time consuming when there are many students. Moreover, students cannot build their own communities or knowledge networks inside a course, which is important, because they have limited capabilities on the LMS. Students cannot discover other students from other semesters or from other study programs with the same particular interests in various fields because they are limited to knowing only the participants of their own courses. This is where PaC came as a solution for the LMS.

5.1 Related Background Information

5.1.1 Social Awareness Systems

Context or situation awareness is a major research topic in areas like transportation, aviation, navigation and air traffic control. Another type of context-aware systems is represented by social aware systems that use their sensors and their capacity of analysing data to take smart actions. Several

attempts were made do define and understand what a social aware system is and how does it behaves [122] [123]. In [122], the authors divided awareness into 2 types: superficial and non-superficial. The non-superficial awareness was considered more important as it was described as the process of understanding the individuals and their actions. Finally, they arrive at the conclusion that social awareness requires trust and the absence of fear, meaning that users that are using social awareness systems should have a certain level of trust otherwise their behaviour would affect the other members. Coates [123] describes social awareness software as *"software which supports, extends, or derives added value from, human social behaviour"*. Social awareness systems are systems that are capable of helping connected individuals or groups of people to be aware of different activities, interests and situations [124]. Studies about social aware systems include the creation of a framework for the development of social awareness systems [125]. This framework was designed to ease the authors' investigations on how to include information of close friends into a person's work environment.

5.1.2 SNSs and LMSs

Previous discussions argue that courses inside a LMS are focusing too much on simply delivering content to its users [126] [127]. The authors argued that social networking, as an emerging technology, could be applied in education because it represents an important factor on how we know and understand things. Recently, in the e-learning research community, many efforts have been made to integrate SNSs with e-Learning systems [128] [129] [130]. In the first study [128], the successful integration of a SNS, i.e. Facebook, with a multimedia based LMS, i.e., Coome, was done as a possible solution for increasing the level of interaction and engagement of the students. The second study [129] investigated, during a period of 4 months how students and teachers would collaborate inside a hybrid e-learning environment using specific social networking tools. The importance of this study was to observe and analyse the different types of interactions that would support and maintain multiple styles of learning. Du et al. [130] propose an e-learning collaborative and interactive platform that integrates social software inside the LMS. One of the benefits of having such a platform is the ability to provide a personalized space for each user where users could read information coming from their groups and circles of friends. The key feature of the whole model is a filtering mechanism that would process the information coming from the user's knowledge network and social network. Creating groups, as part of a course, on the LMS, would encourage the process of active learning, and would better support the interaction between the participants, and increase online participation in the platform.

5.2 Research Hypotheses and Objectives

Additional to the main research questions presented in section 1.2, several research questions related to the e-Learning domain were also addressed. In order to validate these research questions several LPaSs were developed and implemented on top of a PS, which included the LMS. The aim was to establish to which extent the users of a LMS would benefit from having social features inside their e-learning platform, which properties of learning software should be improved in order to have proactive characteristics and if organizing students into various OCoPs would have a positive impact on how students interact, learn and collaborate inside a LMS. For addressing the questions above, three hypotheses **(RH)** were developed:

RHa. *Social features support OCoP and increase the student engagement inside a LMS.*

RHb. *The e-learning platform is aware of different situations and can adapt to the user's need through PaC.*

RHc. *Community building based on multiple themes allows a better interaction, creates well-defined knowledge networks and stimulates the learning process of the students.*

5.3 The Proactive System

For making the Moodle platform a proactive system, a middleware architecture was used [119]. This architecture followed the initial rule-based proactive model proposed by Zampunieris [14]. The middleware architecture was composed from several components both on the server-side and on the client-side, as shown in figure 1 in [131]. The server side consisted, apart the Moodle application written in PHP, a *PE*, a *local database* and a wrapper to access Moodle's database on the server-side. The *PE* was composed of a *rules engine* for processing rules and several queues for storing the rules. On the client-side, which was the web interface of Moodle, multiple elements were implemented such as a pop-up question box and two plug-ins, i.e., a side-block for displaying the OCoPs and the activity inside them and another side-block for coaching messages, illustrated in figure 5.1 and figure 5.2. The structure of pop-up question box was designed to show different questions, with different content, but with the same possibility of answers for all of them, i.e., "Ok" and "Cancel". The answer of the user was then immediately sent to the server via web sockets and then registered in a specific table of the local database, created specially to hold extra information that cannot could not be stored on LMS's own database.

Figure 5.1: Pop-up question box and *Social Groups* side-block on Moodle.

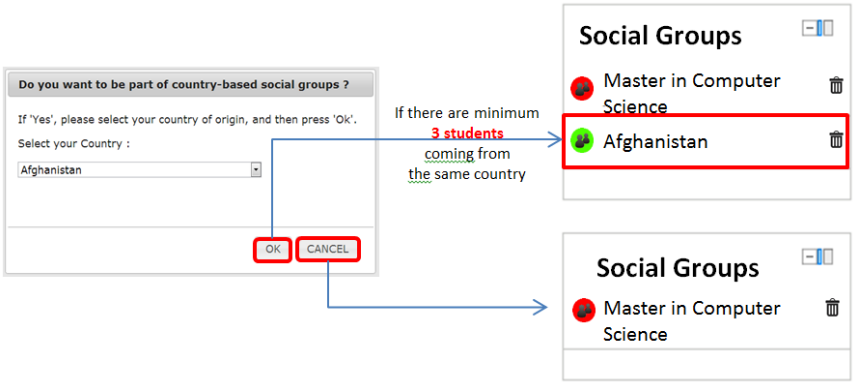


Figure 5.2: *Coaching Messages* side-block and the detailed list of messages on Moodle.

The two side-blocks were used to make users more acquainted with the events happening in their social groups and to show to everyone a list of their groups together with their level of activity. These blocks were displayed on the right side of any web page inside the LMS during valid login sessions, after a user was able to successfully log-in to his/her account. The *"Coaching Messages"* side-block, shown in figure 5.1, was created to increase the level of awareness of each user by providing fresh information about the groups, by alerting the users in case of an important event and by showing them that the PS was dynamic and active all the time. The first version of this side-block was used for conducting experiments to obtain an automatic and enhanced management of the online assignments on Moodle [132]. The side-block displayed only the subject of the last five messages and their importance, which were calculated with the help of LPaSs, on the left side of the subject. A detailed list with all the messages was opened when a user wanted to read all his/her messages and clicked on the *"Read All"* button situated at the bottom of the *"Coaching Messages"* side-block. Another button was provided for deleting any message both from the block and from

the page with detailed messages. The *"Social Groups"* side-block, shown in figure 5.2, showed to each user a list of 5 groups, according to the time and date they were joined by the user. To the left of each OCoP an icon was added to show if the OCoP was active or not, i.e., red for inactive groups and green for active groups, and, to the right of each OCoP, a button was also added to remove the current user from that specific group.

5.3.1 Local Proactive Scenarios

Working with only one system, i.e., the Moodle platform, involved developing *PaSs* that analysed only local information and took only local actions. Also called LPaSs, these scenarios were based on the first type of scenarios presented in chapter 4. The LPaSs that were developed for this study case were included into a structure called *the Proactive Cycle*. This structure was composed of 3 main phases: *"setting-up the groups"*, *"enhancing social interaction inside the groups"* and *"adjusting the groups"*. All the rules were part of one LPaS of the 3 main stages of the proactive cycle, except the first LPaS **(LPaS01)**, which included 2 rules, i.e., **S001** and **S002**. These rules were included at the first run of the PS.

Initially, for this study, each LPaS had only one corresponding rule, except for the first LPaS which would trigger the other LPaSs. This approach would was not so efficient for larger groups of LPaSs, where situations would have multiple corresponding actions. This common challenge of rule-based systems, as seen in figure 2.6c in chapter 2.1.2, was addressed in chapters 4 and 4.2, by dividing scenarios into multiple corresponding rules which would target a common goal, as seen in figures 2.6a or 2.6b or 2.6d. Also, this first set of LPaSs did not contain the naming convention established in chapter 4. More precisely, a *proactive meta-scenario* was considered a LPaS that was composed of a **cyclic rule**. However, the *meta-scenario* is in fact a *meta-rule*, presented in chapter 3.2.2, which is in charge of creating other *PaSs*. So, the only *proactive meta-rule* was **(R02)**, part of **LPaS01**, which created all the other LPaSs.

Designing LPaSs

LPaSs were designed according to the goals and target actions of the system using the methodology described in chapter 4.2. The first main goal of the system was to create a course on Moodle and to inscribe all the participants into this course. **LPaS01** was designed to achieve this goal. It is composed of 2 PRs, i.e., *R001* and *R002*. The content of the first rule, i.e., *R001*, is shown in pseudo-code in figure 5.3. The other goals of the PS concerned the social groups, i.e., to create the groups and their materials, to check and adjust the social interaction inside these groups and to end of adjust the life cycle of the groups. The goals also represented the 3 different stages

```
1: procedure LPaS01(R001)          ▷ First rule part of the first LPaS01
2: dataAcquisition:
3:     currentTime ← systemGetCurrentTime
4: activationGuards:
5:     if currentTime > startTime then return true
6:     else
7:         return false
8: conditions:
9:     if courseExist(courseShortName, courseName) then return true
10:    else
11:        return false
12: actions:
13:    MySQLOperations.startTransaction()
14:    Begin
15:        createCourseCategory(categoryName, categorySortorder)
16:        updateCourseCategory(categoryID)
17:        createCourse(courseName, categoryID, courseStartDate;
18:        createCourseSection(courseID, sectionCityDescription)
19:        insertEnrolMethods(courseID)
20:    End
21:    MySQLOperations.commit()
22: rulesGeneration:
23:    if getActivated() then createRule(new S002(courseID))
24:    else
25:        createRule(this);
```

Figure 5.3: Proactive Rule R001 in pseudo-code.

of the *proactive cycle*. The first stage was composed of LPaS101, LPaS102, LPaS103, LPaS104 and LPaS105, the second stage of LPaS201, LPaS202 and LPaS203, while the third stage of LPaS301 and LPaS302.

5.4 The Experiment

For testing the LPaSs and assessing the three hypotheses, a study was conducted during the second half of a summer semester at the University of Luxembourg, as an online experiment, using the local Moodle platform [117]. With the help of the *LPaS*, the eligible users were enrolled inside a specialised course and inside the first type of OCoP, based on the study programs of the users. Two resources were added at level of the course, containing a user manual and a detailed description of the whole project. A set of three communication tools, including a forum, a chat and a folder, were generated when each OCoP was created. After the first successful login of

each participant, two new types of OCoPs were proposed through a question box specifically built for the GUI of Moodle. New OCoPs and their resources were only generated if there were minimum three participants willing to join these groups. The OCoPs were based on ***Groups*** and ***Groupings***, features already integrated in the latest version of Moodle. **Groupings** can make activities within a course available only to a set of users, which are firstly arranged into groups and groupings. Group modes were set to ***Separate Groups*** for the whole course, i.e., only members of the same group could see the group activities while to all the others they were invisible, which means that the visibility of a resource was determined by its group membership. By default, the users inscribed in the course containing social groups only had basic permissions for using the resources of the group, meaning that they could use them, they could add documents to the folder but they could not change the settings of the existing forum, chat and folder.

5.4.1 Participants

A total of 2404 individual participants were included in the tests, but only 1088 were selected for the enrolment in the course which contained the first set of communities due to the fact that they were only assigned in the ***student*** role in their other courses. A particular case was given by the PhD students that represented a large percentage of the total number of unregistered users. They have multiple roles on the Moodle platform like ***teacher*** or ***manager***. The main reason for only selecting students for eligibility to participate in the experiments on Moodle was the assumption that students would feel more comfortable in sharing experiences, course materials and projects between one another without having teachers or assistants monitoring their activity and actions. The participants were divided into two main categories: ***engaged students*** and ***uncommitted students***. Engaged students refers to the students whom accepted being a part of at least one other community, and voluntarily joined the newly formed groups, while uncommitted students refers to the students that were automatically placed in their study-based OCoPs but did not want to take part of any other community.

5.4.2 Data Collection and Analysis

Data was continuously collected during the experiments by both the LMS and the proactive scenarios and meta-scenarios. Additional data was also gathered because some important data could not be registered in the LMS's database. For example, actions like delete a coaching message, view a coaching message and leave a community, which were specific to the two Moodle blocks used in the experiment, were recorded in another database specifically created for information about the communities of practice and their

members.

Analysing the data was done in two main phases. The first phase consisted of a step-by-step process of examining the data for making local decisions such as creating or ending the life cycle of a community. This continuous process of evaluating the data was an outcome of using PaC. It was necessary for sustaining the second hypothesis, **RHb**, and for answering one of the research objective of this case study. The second phase, which happened at the end of the experiment, included gathering relevant data about the communities, the interactions that took place inside the communities and the resources that were used by the participants.

5.4.3 Measurements

Quantitative measures were taken for supporting the first hypothesis, **RHa**. For evaluating the activity of the participants it was taken into account their actions related to the three main resources that were provided, i.e. the forum, the chat and the folder, from the moment they were enrolled in their communities. Operations such as *Add Discussion*, *Add Post*, *Update Discussion*, *Update Post*, *View Discussion*, *View Post* were considered for the **forum** resource, *Report*, *Talk*, *Update* and *View* for the **chat** resource and *Edit*, *Update* and *View* for the folder resource. The activity inside the groups was measured in two stages, i.e., three weeks before the beginning of the experiments and three weeks after the start of the experiments. These periods were considered very relevant because they were situated in the middle of the semester and did not include weeks where the activity would be influenced by other factors like the beginning or end of the semester, when students' online activities are quite numerous on the e-learning platforms.

The second hypothesis, **RHb**, was mainly evaluated with respect to the proactive procedures that determined which participants did not leave and actively took part in their OCoPs until the end of the semester. These procedures included *notifications*, *reminders* and *messages* sent by the proactive LMS to the participants in order to encourage them to participate actively in their OCoPs. These procedures also informed students about new posts and new resources, and offered guidance within their social environment.

The last hypothesis, **RHc**, was verified by investigating quantitatively and qualitatively communities in terms of collaborative work regarding learning practices, knowledge networks and particular interests. It was also considered how many OCoPs of each type were initialized and how many were waiting to be created.

5.4.4 Results and Discussions

A total number of 3618 actions was recorded for the three main tools provided inside each OCoP. *Forum actions* were account with 46.6% of the

Table 5.1: Results of Forum actions inside all the different categories of OCoPs

Types of OCoPs	**Forum**					
	Add discussion	Add post	Update	Update post	View discussion	View post
City-based	7	8	2	6	98	196
Country-based	9	8	2	6	135	263
Study-based	7	7	1	4	254	675
Total	23	23	5	16	487	1134
	1688					

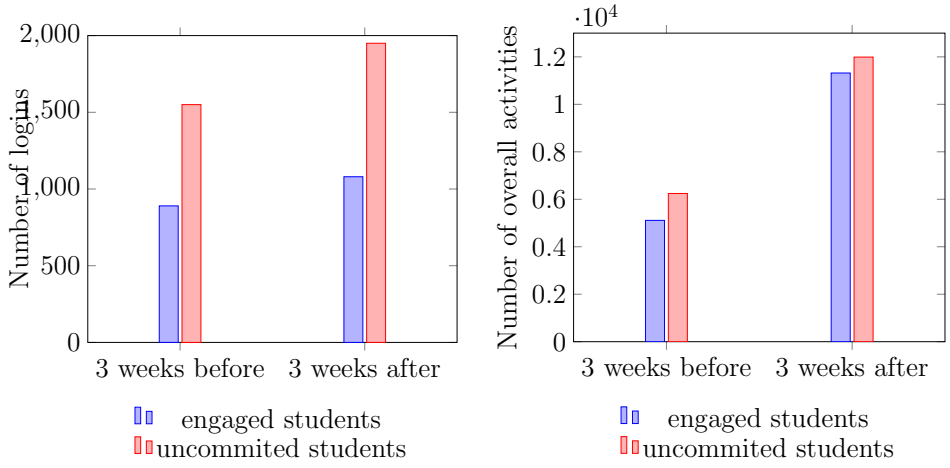
Table 5.2: Results of Chat and Folder actions inside all the different categories of OCoPs

Types of OCoPs	**Chat**				**Folder**		
	Report	Talk	Update	View	Edit	Update	View
City-based	35	9	14	127	7	2	86
Country-based	46	27	14	163	9	2	107
Study-based	166	51	9	614	8	2	432
Total	247	87	37	904	24	6	625
	1275				655		

total number of actions, the *chat actions* with 35.2% and the *folder actions* with 18.2%. The forum was by far the most used tool in all three categories of social groups, also shown in tables 5.1 and 5.2. Actions like *viewing forum posts* and *viewing discussion* represented more than 96% of the total activity in the forums. These results show that students were more reluctant to add or update a post or a discussion, and would rather participate as observers. The *folder* and *chat* tools were rarely used by the participants considering the number of participants and the short period for the experiments.

One explanation for the lack of activity of the students is the structure and the design of these tools. For example, the chat was designed more like a chat room, i.e., a place where students can participate in discussions at any time. To use the chat, students had to be online at the same time, which is quite improbable for communities with few members. Also, discussions between students were not saved by the LMS, so each time the chat had no remaining participants, the LMS would remove all the conversations. This explains the difference between *conversing* (87 recorded actions) and *viewing the chat rooms* (904 recorded actions). Adding and editing content of the folder in each community, with a total of 30 actions, was significantly less used opposed to other operations like viewing the resources of a certain folder which recorded more than 600 actions.

Out of the three main categories of OCoPs the **study-based** communities had the highest number of activity. In the study-based OCoPs, the



(a) Statistics containing number of logins (b) Statistics containing overall activities inside the LMS without logins

Figure 5.4: Comparing activities inside the LMS before and after the start of the study

LMS recorded a total of 948 actions concerning the *Forum*, 442 concerning the *Folder* and 840 concerning the *Chat*. This came as a consequence of having five times more participants in the **study-based OCoPs** than in the **city-based OCoPs** or **country-based OCoPs**. The registered number of activities showed a small difference in favour of country-based OCoPs compared to city-based OCoPs. One argument is that people participating in the country-based OCoPs found it much easier to express themselves in their native language. A relevant example is given by the community of students coming from Portugal, which represented one of the most active social communities in the study. All the forum posts and replies were written in Portuguese, which showed a clear desire of students to communicate in their native tongue.

Measurements, presented in figure 5.4a, indicate that the average login actions per week, of both engaged and uncommitted students, increased in the first three weeks after the launch of the experiments. The rise in the case of the engaged students, i.e., 21.1%, was similar to the one in case of the uncommitted students, i.e., 23.3%. If an engaged student logged into his/her Moodle account 3.5 times in average per week before the start of the experiment, then, after the start of the experiment, this average reached 4.3 logins per week. For the uncommitted users, the average went from 1.67 to 2.03 login actions per week.

In figure 5.4b, a similar increase as the one in figure 5.4a can be noticed for all the activities performed by the students within the LMS except their successful login actions. These activities included actions completed

Table 5.3: Students who left their communities compared to those that stayed

Types of OCoPs	Number of students that left his/her community	Number of students that stayed in his/her community
City-based	9	98
Country-based	6	159
Study-based	222	866

by the students inside their courses regarding assignments, grades and other resources and actions done at the level of the LMS including checking the calendar for important dates, upcoming course events and updates for discussions. The recorded activities showed, in the 3-weeks period after the beginning of the study, an increase of 32.0% for the students that took an active part inside the OCoPs and an increase of 10.7% for the uncommitted students.

The results indicate that inserting more opportunities to socialize as part of a course into a LMS have a positive effect on the online engagement of the students, which supports the first hypothesis **RHa** and which confirms expectations about the first research objective.

Table 5.3 shows that the number of students that preferred to remain in the groups is quite high compared to the number of students who decided to leave their social groups, especially for the engaged students. The number of students who decided to leave their groups in the time interval of 4 weeks constitutes 9.1% of the total number of members in study program-based OCoPs, 0.37% for the city-based OCoPs and respectively 25.6% in the country-based OCoPs. This demonstrates that students were quite satisfied with their communities and preferred to remain inside these communities. One reason for this outcome could be that students were alerted when the system detected relevant activities inside their OCoPs such as new forum posts or discussions, new resources that were added to folders or when new chats or discussion took place. Students were also informed when their OCoPs were inactive for more than 3 days, meaning that there was no action detected by the system inside the OCoPs. During the experiments a total of 5913 reminders, hints and notifications were generated for all the 1088 students, meaning an average of 9.2 messages per user per week. Under the proactive analysis lens, LPaSs were used for constantly analysing the situation inside the OCoPs and for deciding whether to send either a notification, reminder or hint.

5.4.5 Conclusion

These empirical investigations clearly demonstrated that the limitations of a LMS can be overcome and the LMS can be transformed into an intelligent and aware system. The key contribution demonstrated that including

social features inside the LMS and creating OCoPs as part of a course, with the help of LPaSs, enabled different collaborative learning techniques. LPaSs permitted the LMS to actively seek out and analyse information about OCoPs and their members, and make decisions based on this information without an explicit command from any user or administrator.

The expectations about the first hypothesis were confirmed by the results, which show that employing social features as part of a student's courses increased the level of online student engagement. It is also likely that this could positively influence a student's performance when completing online assignments and tasks. The second hypothesis was not fully measured and more specific actions need to be implemented for measuring, with the help of PaC, the positive benefits of OCoPs and by applying them inside the LMS. Future efforts will focus on how to make activities more engaging inside OCoPs, on how to add valuable content to expand the general knowledge inside OCoPs, and on how to make OCoPs more attractive in order to increase user participation. Creating a variety of communities offers students the possibility of creating well-defined, knowledge-based networks and social networks inside the OCoPs. Arranging students into OCoPs to exchange information, express new ideas and to observe one anothers work led to collective solutions for various problems and questions; it also stimulated the learning process of these students, thus confirming the third hypothesis, and answering the third research question. OCoPs require space to expand and need time to evolve in order to reach a mature phase. Social barriers need to be broken by their members whom do not know one another, and without physical presence, but reaching this level of comfort within an academic LMS will take some time. The evolution of OCoPs inside e-learning platforms should continue to be observed and investigated. Future work should also include a proactive mechanism capable of motivating the users inside the OCoPs. The integration of OCoPs should not be limited to a single specialized course, but rather as part of all courses inside a LMS. Finally, more focus should be placed upon the correlation between the engaged students that get socially involved in the LMS and the student's academic performance in that course.

Chapter 6

Application 2 - SilentMeet

The second case study introduces a mobile application, i.e., SilentMeet, which uses group-driven collaboration and location-based collaboration for automatically switching smartphones into silent mode during meetings or important events. It is the first application that employs GPaSs in order to achieve its target goals. This application was firstly published in [133] and then an extended version in a journal [134].

Apart from the main function of the prototype application, which is to silence mobile phones during meetings, there are three main objectives: a) to provide a collaborative application capable of acquiring contextual information from various devices, b) to check if it is possible to achieve collective reasoning using a rule-based middleware architecture for mobile devices, and c) to validate GPaSs in a real-case example. Objective (a) is addressing **RQ2**, objective (b) focuses on **RQ3** and **RQ4**, while objective (c) targets to provide one of the answers to **RQ5**.

More precisely, for the first step of the collaboration, a partial agreement algorithm is used for establishing if a meeting is confirmed by its participants and, during a second round, for confirming if the meeting will take place, based on the location of the participants. The application avoids those cases when a meeting is accepted but the participants are not coming to the meeting or when participants do not reply to the meeting invitations but they are still attending the meeting. SilentMeet uses a new technique for exchanging information, for coordinating and for taking distributed decisions, called GPaSs. For executing GPaSs, a rule-based middleware architecture for mobile devices, presented in chapter refmobiledev, is utilised. GPaSs and the middleware architecture allow developers of collaborative applications to define the actions of their applications in a structured way without having to take care of the communication and coordination of the mobile devices. Also, there is no need for developing a server-side application where the global decision would be taken; all the logic is integrated into GPaSs.

6.1 Related Background Information

Related work is divided into several categories which are considered the most relevant for this case study. The first one examines context-aware mobile collaborative systems, where the focus is on the context of groups of users, the second one discusses relevant collaborative middleware architectures for mobile devices, the third one has examples of collaborative mobile applications developed for other fields than the ones that turn the mobile phone into silent mode and the last one contains several examples of mobile applications developed for silencing smartphones in various situations.

6.1.1 Context-Aware Mobile Collaborative Systems

A key characteristic of mobile collaborative systems, where groups of users perform common activities and have the same interests, is the ability to acquire different contextual information from multiple sources, not only from local, individual sources. The idea is that multiple devices can observe and reason about the same event from different angles. Multiple frameworks were developed to ease the creation of context-aware mobile applications [135] [136] [137], but the aspect of reasoning about the shared contextual information, coming from multiple applications, was not explored. Wang et.al [138] propose a context-aware strategy for collaborative mobile applications based on location. However, the collaboration process is limited as the context information depends only on the near proximity of the participants. Despite a collaborative strategy for sharing context between devices, the authors only provide in [139] a simple integration of the context, which is just added to the knowledge base.

6.1.2 Collaborative Mobile Middleware Architectures

Numerous studies [140][141][142] have been conducted that provide middleware architectures as tools for developing collaborative applications. One important difference is that these studies look at collaboration from a different angle. More precisely, they concentrate on user-centred collaboration, where the focus is to get the users to interact more and more with their applications on the mobile devices. The issue is that these applications would depend too much on the actions of their users and, if the users do not engage properly in each step of their interaction with their devices, the applications may remain at the same step. Opposite of this, PaC tries to reduce the users' involvement by automatizing large amounts of processes. By doing so, the users can concentrate more on the essential tasks they are required to do. MobiSoC [143], a middleware enabling mobile social applications, showed on initial tests indicated that this framework provided good response times for 1000 users for location-based matching and place-based matching.

6.1.3 Collaborative Mobile Applications

Using the WatchMyPhone (WMP) tool kit, a shared text editing collaborative application was developed in [144] with the help of Mobilis Framework [145]. The proposed tool kit is compressed into a library and can be used by other collaborative applications by including this library into their project. However, this type of applications imply synchronous collaboration, which is another area of research. Another field where collaboration is crucial is represented by mobile-based games. In [146], the authors created a mobile game based on collaborative game play. The game was developed on top of a middleware architecture. However, the whole framework consisted not only from a client side middleware but also from a server side middleware, which handles the biggest part of the complexity.

6.1.4 Applications for silencing the Smartphone

Many commercial mobile applications exist on the market, like *Silence* [147], *Go Silent* [148] or *Advanced Silent Mode* [149], which automatically switch off the sounds of mobile devices based on the user's preferences. These simple applications are focused on one user and perform only local tasks like checking the user's predefined preferences or detecting calendar events. They do not use any kind of collaboration with other devices to make the application smarter. For example, *SilentTime* [150] searches for weekly events in the local schedule and automatically silences the user's phone if a future event is detected. It offers the user the possibility to add exceptions, in case he/she is waiting for an important phone call. However, the application has a couple of downsides. First, it is exclusively based on the user's input, i.e., a calendar event or exceptions of a special situation will only be detected if the user creates them before, and second, it does not use any kind of communication with other devices to check if the events will take place or not. Another example is *AutoSilent* [151], which is slightly different from *SilentTime* because it adds an extra step of verification before muting the user's phone, i.e., it will verify if the user's location corresponds with the event's location at a certain time. This extra feature is again just a simple check because it does not use any kind of collaboration, like, for example, checking also the location of the other participants.

6.2 Domain-Specific Problem Statement

There are quite a few mobile users who went through embarrassing situations when their phones rang during important meetings, lectures, exams, presentations, concerts, interviews or key talks offered at international conferences. Imagine, for example, that during a viola recital of a famous musician, the mobile phone of a person starts ringing, like it did during a recital in Slo-

vakia [152]. The musician is not only interrupted but he/she could also lose focus and find it difficult to continue. Another case happened during a performance of the New York Philharmonic in 2012, when the conductor had to stop the orchestra because of a person sitting in the front row whose smartphone produced a disturbing sound because of an alarm going off [153]. This happened despite the fact that an announcement was made to silence all the mobile phones in that room. There are many more other examples when muting the phone is a mandatory requirement. The main problem is that each user has to manually configure his/her phone to be silent during important events. And often, they forget. The detrimental and disruptive effects of interruptions are a problem which got a lot of attention in many different areas [154] [155]. A general common strategy or approach which performs collaborative actions is missing.

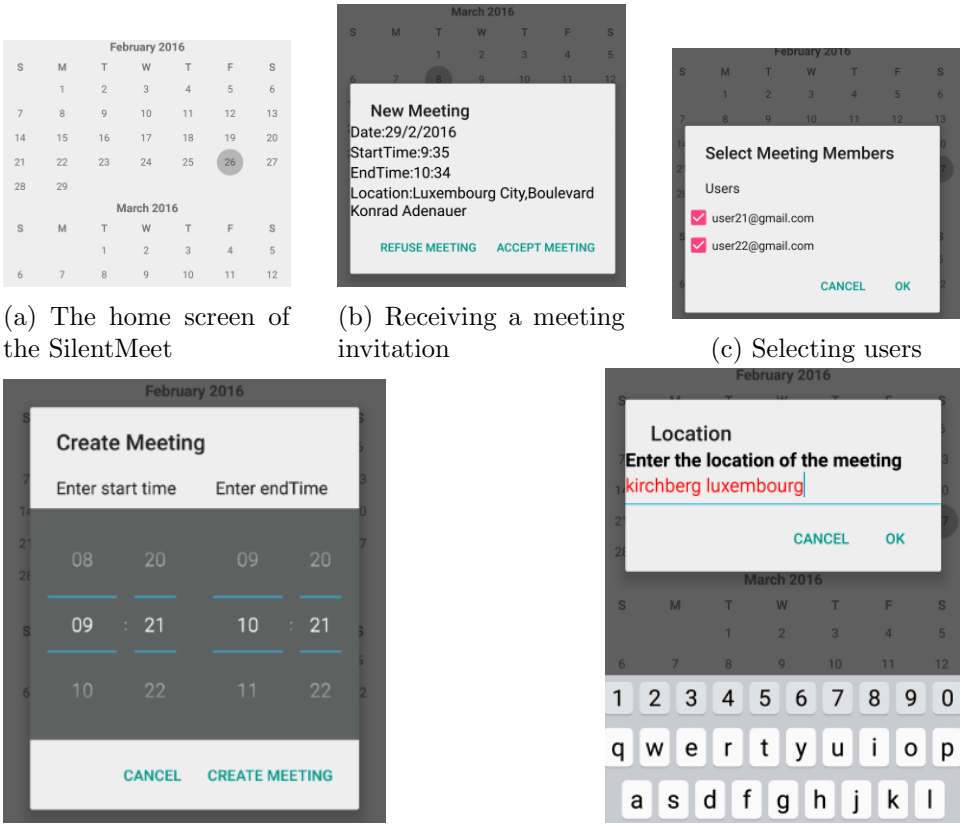
Let us imagine the following real-world situation: an important event is about to begin. The mobile devices of the participants, located in their pockets, go automatically into silent mode. The participants do not have to worry they forgot to silence their mobile phones, they can focus more on their important tasks. The meeting can continue without any interruptions or embarrassing situations.

6.3 A Rule-Based Solution - *SilentMeet*

SilentMeet is a mobile collaborative application that was developed in order to minimize the risk of interruptions and their distracting effects during an important event such as meetings, interviews or public events. Moreover, in order to have an efficient collaboration algorithm, part of the user's actions are automated by using PaC. The main difference between SilentMeet and other applications is that SilentMeet does additional checks, based on GPaSs or the collaboration with the other mobile devices, to establish if a meeting is taking place or not. More precisely, it checks, among the possible participants of the meeting, if there are at least 2 users that have accepted to attend the meeting and have the meeting in their calendar, and, finally, on the date of the meeting, it will check the location of the users 15 minutes before the start of the meeting. The additional checks are necessary for trying to avoid cases such as silencing a smartphone even if the meeting is not taking place.

6.3.1 The GUI of SilentMeet

The application was developed for the Android OS and consists of a main activity with a calendar view working as a date picker, laid out in figure 6.1a. The basic idea is to provide the user with a simple interface for creating the meeting, i.e., selecting the participants for this meeting, the place where the meeting will take place and the starting and ending hour of the meeting. In



(a) The home screen of the SilentMeet

(b) Receiving a meeting invitation

(c) Selecting users

(d) Selecting the start and end time of a meeting

(e) Entering the location of the meeting

Figure 6.1: SilentMeet's GUI.

figure 6.1, an example of creating a meeting using SilentMeet is provided. At the start, the user selects a date from the calendar when the meeting should take place. The date of the meeting should be higher or equal to the current date otherwise the meeting will not be created. Afterwards, a new dialogue opens asking the user for the start time and end time of the meeting, as shown in figure 6.1d. Again, the start time has to be bigger than the current time if the meeting is on the same day or, if the meeting is on a further day, the start time has to be smaller than the end time. Then, the user has to select the participants for the meeting. He/she will have to choose from a list of predefined users, as presented in figure 6.1c, i.e., the users that have agreed to be part of the same group for creating future meetings. Additional information about how these groups were created are given in section 6.3.2. And finally, before sending the invitation to the other members, the user has to input the location of the meeting, depicted in figure 6.1e. The location is given by the user as text, which is then converted into

GPS coordinates. These coordinates are stored locally on the phone, and, just before the meeting takes place, they are compared with the current GPS coordinates. The members selected for the meeting receive an invitation with the date, location, start time and end time of the meeting, depicted in figure 6.1b. Then, a user can accept, reject or not respond to the invitation by selecting another area on the application's screen. If the invitation is accepted, the response is sent back to the initiator of the meeting. For a meeting to be confirmed there needs to be at least 2 participants that accepted to participate. The initiator of the meeting can cancel at any time the meeting if he/she decides that the meeting should take place only if all the invited members accept the invitation or for other reasons.

6.3.2 Grouping the participants for a meeting

Creating groups of people that knew each other and were more probable to meet was predefined. Although, this step can also be partially automated with the help of another GPaS, for SilentMeet the groups were predefined before an event was created by each user. Then, when a calendar event is created, the user also adds the participants from the list of users belonging to a certain group. Users can perform collaborative actions only if they are part of the same group of the same event. So, users first have to build their own groups or agree to be part of already created groups. For example, in a company, the secretary of a department creates a group for the employees of that department that have meetings regularly. By joining this group, the members agree that their mobile phones can be silenced by the application of the other members, after multiple rounds of negotiation. Moreover, extra conditions and checks are taken into account like the location of the event and the participants, the date and the hour of the event and the local preferences of each user. In figure 6.1c, a user is about to start a new meeting and decides to invite both members of his/her group, i.e., users with emails *user21@gmail.com* and *user22@gmail.com*, to this meeting. In the given example, the group is composed of precisely 3 users.

6.3.3 Global Proactive Scenarios for *SilentMeet*

The idea of SilentMeet is that the devices participating in a collaboration process can take decisions based on global information, coming from other smartphones equipped with PEs, therefore enhancing the local information. Each device is able to make use of the global knowledge that is created by all the devices involved in the collaboration. For example, a basic application would only be able to detect an event based on the local information provided by the calendar of a device. SilentMeet is able to query all the relevant devices to obtain more precise information about that event by using a particular GPaS.

```
1: procedure GPaS1(R011)          ▷ Looking for new pending meetings
2: dataAcquisition:
3:     boolean newMeeting ← engine.isNewMeeting()
4: activationGuards:
5:     return true
6: conditions:
7:     return newMeeting
8: actions:
9:     List meetings ← engine.getListOfPendingMeetings()
10:    for for each m in meetings do
11:        Begin
12:            ArrayList params.add(m.getMembers())
13:            params.add(m.getDay(), m.getMonth(), m.getYear())
14:            params.add(m.getStartHour(), m.getStartMinute()
15:            params.add(m.getEndHour(), m.getEndMinute())
16:            params.add(m.getLatitude(), m.getLongitude())
17:            params.add(m.getMail())
18:        End
19:        if engine.meetingRequestWasNotSent(meeting.getId()) then
20:            engine.sendMessage("R021", params, deviceIDs, 100)
21:            SentMeeting sentMeeting ← new SentMeeting(m.getId())
22:            engine.getLpeDBWrapper().save(sentMeeting)
23: rulesGeneration:
24:    createRule(this)
```

Figure 6.2: Proactive Rule R011 in pseudo-code.

SilentMeet uses two GPaSs: the first one for creating and establishing if a meeting will take place and the second one to check, just before the meeting, the location of the users and to decide if they are close to the meeting's location in order to put the mobile phones into silent mode. A meeting is confirmed in two steps: the first step checks if the participants of the meeting have accepted the invitation to the meeting and have that particular meeting in their calendars, and the second step checks the location of the participants to see if it corresponds with the meeting's location, on the exact date, 15 minutes before the meeting is about to start. This algorithm with all the extra checking steps is useful because false positives are avoided, i.e., those cases where the meeting is not taking place but the phones are still put into silent mode.

GPaS1

The purpose of the first GPaS is to create a meeting and establish if a meeting is confirmed by checking with the mobile devices of the other par-

ticipants. This is only the first step of verifying if the meeting is going to take place. It is necessary for starting the second verification step, i.e., the second GPaS. Each device needs additional information from the other devices before taking a decision. The idea is that if multiple devices, part of a collaboration group, have an event in their local calendar, with the same date, time and location, it is very probable that the event will take place. It is presumed that the same information about an event coming from 2 different devices part of the same group is enough for the application to decide what to do next, e.g., in this case, it will activate GPaS2. The minimum number of 2 devices is motivated by the fact that a device should not be able to mute, by itself, other devices without any kind of agreement. GPaS1 allows a decision to be taken without the confirmation of the meeting coming from all the participants, as this is very difficult to achieve in real-life situations, with large numbers of users, where each user is expected to manually add the event into the calendar.

GPaS1 is composed of 4 PRs, i.e., **R011**, **R021**, **R012** and **R022**. **R011** is one of the rules that is running from the beginning, when the application is installed, and is checking for new meetings in the local database. The code of the rule **R011** is shown in figure 6.2. For SilentMeet, the PE was set to execute an iteration each 5 seconds, so, R011 will be checking each 5 seconds for a new meeting. When creating a meeting, as seen in figure 6.1, SilentMeet registers the meeting's location, date, start time, end time and the persons invited to that meeting. The status of the meeting, after it was created locally on the smartphone, is set to *pending* and *unsent*. **R011** checks for all the pending unsent meetings, this step being part of the data acquisition method of this rule, and, only if such meetings are detected, the actions method will be activated. Inside this method, an invitation will be sent to the users selected to attend the meeting. The invitation will contain all the meeting's details like its *location*, *start time*, *end time*, *date* and *members*. The name of rule to be activated on the receiving PEs is included among the parameters when the message is sent to the receiving PEs. And so, rule **R021** will be activated on the devices of the receivers. For example, if *user1* decides to create a meeting and invite *user2* and *user3* to that meeting, on the devices of user2 and user3 the PE will activate rule **R021**. When rule **R021** gets activated, it means that the receiver of the message is invited to a new meeting. In its *data acquisition* phase it looks in its own calendar if there is no other meeting on that specific date and time and if this invitation has not already been accepted. If these conditions are satisfied, then this rule will trigger a pop-up dialogue on the mobile phone of this user to ask him/her if he/she accepts to attend this meeting, as seen in figure 6.1b. The operations are part of the actions phase of the rule. This rule does not generate other rules and does not clone itself. When receiving the invitation inside the pop-up, the user has 3 options: accepts the invitation, rejects the invitation or does not respond to the invitation

by changing the application or by clicking on another part of the screen. In case he/she accepts the meeting, rule **R012** gets activated. Even though the cyclic rule **R012** is one of the rules which is executed by the PE at each iteration, it only gets activated when the conditions are true, i.e., the user accepts or rejects a meeting. The immediate effect, if the conditions are true, is to send the response to all the users invited to attend the meeting. If the answer of the user is positive and he/she accepts to join the meeting, then, at this particular moment, there are at least 2 persons that accepted to attend the meeting. In case *the answer is negative*, the device of the same user will register the meeting as **refused** and even if the meeting will still take place with the other participants, the devices of this user will not be switched into silent mode. The receiving devices activate rule **R022** that gets as parameters the answer of a user with regard to a specific meeting invitation. If the answer is positive and the meeting is confirmed by at least 2 persons, **GPaS2** will be activated. If the answer is negative, the device of the initiator of the meeting still waits until all the answers from the invited members are received. Until then, the meeting will be in *pending mode*. If all the answers are negative or part negative and part unanswered before the meeting starts, the meeting will be considered as *cancelled*.

GPaS2

GPaS2 is in charge of the second verification step by exchanging the location of participants, if they are close to the meeting's location. So, it is not enough for accepting the invitation when the meeting is created by a user, for example, 1 week before the actual meeting takes place, but there are 2 extra steps to be completed. The first one is that the users that accepted the invitation have to be near the location of the meeting 15 minutes before the meeting will begin and the second one is that they have to exchange their location with at least one other participant that is also near the meeting's location. Only when these steps are fulfilled, the silent mode will be activated. These extra steps of verification are useful for cases when even if persons confirm their attendance at a meeting, they are stuck in traffic, or they had an emergency and cannot attend the meeting, and so, activating the silent mode on their smartphones is not necessary.

GPaS2 is composed of 3 PRs, i.e., **R013**, **R023** and **R024**. **R013** is only activated when a meeting has been created and accepted by at least 2 participants. It will check the current time on the device, and, if it is equal or less but not more than the meeting's start time minus 15 minutes, it will start to check for the location of the device. If the location also corresponds to the location of the meeting, then the condition for executing the rule's actions are met. These actions include sending a message to the other participants to confirm the device's presence at the meeting's location. After sending the message, the device of this user that activated **R013** waits for receiving at

least one message from another PE of a participant in order to activate the last rule, i.e., **R024**, which turns the smartphone into silent mode. Checking for the location of the user every 5 seconds consumes a lot of battery, so, this action is performed only when the current timestamp approximatively corresponds to the meeting's timestamp. Upon receiving the message from one user that is close to the meeting's location, the activation condition of rule **R023** will be fulfilled on the other participants' smartphones. This rule tells SilentMeet that there is at least 1 person attending the meeting and so, has permission to switch the smartphone into silent mode if the local PE is close to the meeting's location. If this last condition is carried out then the last rule is activated, i.e., rule **R024**. The last rule of **GPaS2** is in charge of finally silencing the mobile phone during that meeting. The only way this rule is executed by the PE is to get through all the previous collaboration steps of both GPaSs and to fulfil all the necessary conditions of each rule. The command for silencing the device is given in the actions phase of the rule **R024**. So, a user that did not reply with yes or no for attending the meeting, can have his/her mobile phone switch into silent mode if his/her device are at the meeting's location, on the same date and same hour as the meeting. SilentMeet considers that by fulfilling these conditions the device is very likely to participate at that meeting, even though it did not provide a precise answer. This case includes a hybrid algorithm for establishing if a meeting will take place or not. The existing algorithms either check for an entry in the calendar or, more advanced applications, just check for the location of the current user but not the other users' location.

Additional Rules

One of the rules that is executed at the beginning by the PE is called **RegisterToServerRule** that registers the user on the GCM server, if not already registered. This will provide to each user a unique ID, which is then used in the communication with the other smartphones equipped with PEs.

6.3.4 Collaboration Process

For muting the mobile devices of the participants of a group, after a calendar event is detected, SilentMeet passes through a couple of rounds of collaboration. These rounds of collaborations are depicted in Figure 6.3 with the help of a sequence diagram. Moreover, it is shown how rules are activated by other rules. This example shows what will happen on the PE from the beginning of GPaS1, when a meeting is created on one smartphone, until the end of GPaS2, when the meeting is confirmed and the devices are silenced. The first GPaS can be activated, for example, 1 week before the meeting actually takes places but GPaS2 needs to wait until the same date and approximatively the same hour of the meeting to get activated. A user

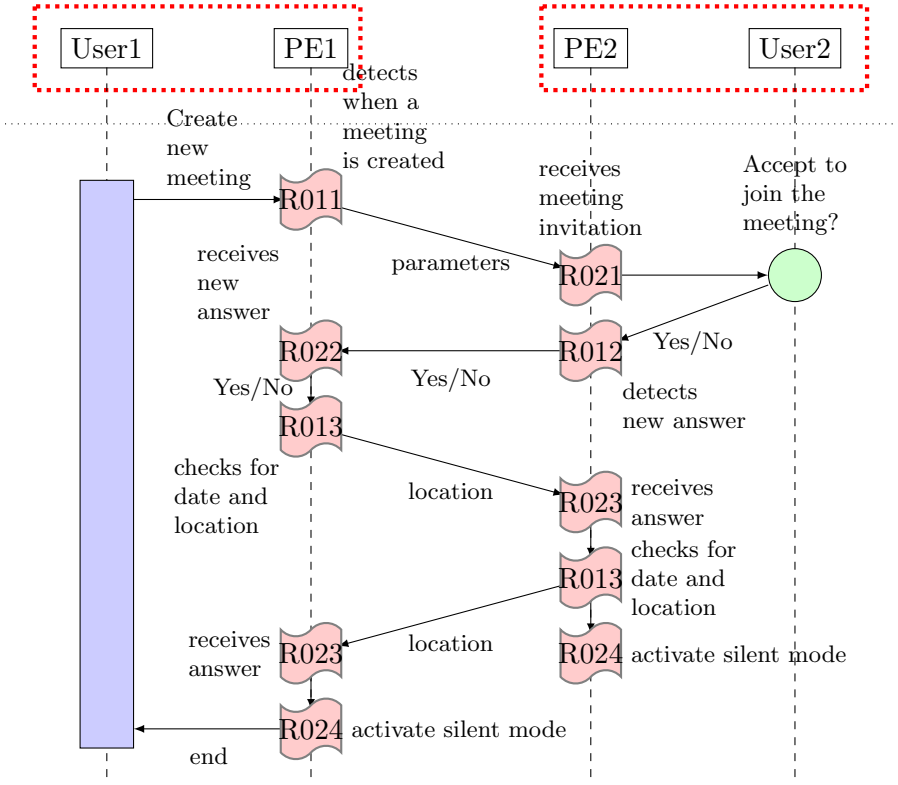


Figure 6.3: A sequence diagram with the collaboration of 2 smartphones of the members of the same group, during a meeting negotiation process.

can receive multiple invitations at the same time and does not have to worry about how they will be handled. This is done automatically by the PE. The collaboration process depends on the communication of PEs, which depends as well on the connectivity setting of each mobile device. The PE performs also error checking and handling in case a message is lost somewhere in the network and no answer is received from other PEs.

6.3.5 Message Exchange between Proactive Engines

As seen in chapter 3.4, PEs exchange information between each other with the help of JSON messages. The messages can contain commands to activate certain PaSs or they can contain just simple context information. Figure 6.4 shows the content of a message written in JSON that is exchanged between PEs in the case of SilentMeet. More exactly, when a meeting is created on the device of the user with the email address *user21@gmail.com*, rule **R011** gets activated and sends a message to *user20@gmail.com* and to user22@gmail.com. The devices that get the invitation to the meeting

```
{
    "instruction":"activate rule",
    "msgId":  /*set automatically*/,
    "senderID":  /*set automatically*/,
    "receiverID": ["user20@gmail.com", "user22@gmail.com"],
    "ruleName":"R021",
    "PARAMETER_TYPES":[
        "String[]", "Integer", "Integer", "Integer",
            "Integer", "Integer", "Integer", "Integer",
            "Double", "Double", "String"
    ],
    "PARAMETER_VALUES":[
        ["user20@gmail.com", "user22@gmail.com"], 29,3,2016,19,
            6,20,6,49.6278694, 6.153422, "user21@gmail.com"
    ]
}
```

Figure 6.4: An example of a JSON message that is passed between R011 and R021, when a meeting is created.

activate rule **R021**, which receives precise data about the date, start time, end time, location, sender and list of invited members of the meeting.

6.4 Tests

A series of tests were conducted locally at our university on 3 different devices: a Samsung Galaxy Note 3 and two Samsung Galaxy S6. All 3 devices used an Android operating system and had installed SilentMeet, working together with the mobile middleware, in order to be able to execute GPaSs and collaborate with each other. The devices were part of a predefined group of 3 participants with the following email addresses used as unique identifiers: *user20@gmail.com*, *user21@gmail.com* and *user22@gmail.com*. During the tests, all 3 devices were connected via WiFi to the same network. Initially, all the devices had their sound turned on.

In the first series of tests, the user with the email address *user20 @gmail.com* and using the Samsung Galaxy Note 3 was the initiator of a meeting and created it on SilentMeet's local calendar. The meeting was set to happen after 10 minutes of its creation time, on the same date, in the same location as all the devices, i.e., the campus at our university. The invitation was displayed on the screen of the 2 other mobile phones and, after the meeting

was accepted by both guests, it was marked in the calendar of SilentMeet. The devices started immediately to check their locations, compared it to the meeting's location and shared it with the other guests, as the start time of the meeting was very close to the current time, i.e., less than 15 minutes difference. All 3 devices were silenced when the meeting started.

The second series of tests happened in the same conditions as the first tests except with one minor detail: one of the invited users did not accept or reject the meeting proposal. However, the minimum of 2 persons that accepted the invitation was reached and so, all the devices had activated GPaS2, which checked their locations before the start of the meeting. Because all the other conditions were accomplished, the 3 devices were silenced again when the meeting started.

For the third series of tests, the 3 users that were part of the meeting proposal accepted the invitation but only one user was at the same location as the location of the meeting, i.e., the user with the email *user20@gmail.com*. The device of this user did not get a location confirmation from the other 2 users involved in the group meeting, so, it did not switch into silent mode. Neither the 2 devices of the two other users.

6.4.1 Measurements

The first goal was to check if the application behaved as expected in the most common cases, e.g., when all the users confirmed their presence at a meeting and their location is the same as the meeting's location when it started, as well as the unusual situations. These unusual situations included not providing an answer of participating to a meeting but still attending that meeting, accepting the meeting's invitation but not coming to the meeting or being the only one present at a meeting confirmed by all the other members of the group and where nobody else is present.

6.4.2 Results and discussions

The tests showed that the application behaves as expected and that all three devices were muted after the negotiation process. In the given settings, it took around 10 seconds to reach a common agreement that the meeting will take place and to mute all three devices. However, this time is highly dependent on the frequency parameter of the Rule Engine, meaning that setting a lower time interval between two iterations will also lead to a faster execution of the GPaSs.

6.5 Conclusion

In this case-study, it was demonstrated that it is possible to easily design and implement a context-aware collaborative application on top of a rule-based

middleware engine with the help of PaC, more precisely, by using GPaSs. *SilentMeet* is able to detect and acquire relevant context-information about calendar events, to use a collective reasoning algorithm to establish if a meeting will take place or not and to take decisions of silencing the smartphone, based on the shared locations of the users. Furthermore, the location sharing process is handled very efficiently in order to reduce the unnecessary battery consumption. At the same time, several parts of the collaboration process were automated and the user's involvement reduced only to the most important operations, i.e., the creation of a meeting, the selection of the participants and the sending of the invitations for participation. *SilentMeet* reduces the possibility of having meetings in the calendar that do not take place any more or which are cancelled by the other participants. The smartphone turns into silent mode only when multiple conditions are met, reducing thus the risk of having the smartphone on mute when not attending any event. With only two GPaSs, composed of 7 PR, it is enough to achieve *SilentMeet's* goals.

Chapter 7

Application 3 - e-Health System

The third case study, which is the most complex of the case studies presented in this dissertation, focuses on providing a proactive e-Health system that will allow patients, following a Cardiac Rehabilitation (CR) program outside the hospital, to exercise safely, according to their recommended training zones [156]. The e-Health system includes a smartwatch application, a smartphone application and several server-side applications, working on predefined personalised LPaSs and GPaSs, which are alerting, guiding and supporting the patients. PaSs that are integrated into the mobile phones and into the server contain knowledge of the medical experts like how reacting to different events, e.g., the sudden increase of the Heart Rate (HR) of a cardiac patient while training at home.

The PS is capable of performing automated patient monitoring, of providing patients with real-time expert feedback, of alerting the patients in case they perform exercises too fast or too intensive, of alerting the medical experts in case an emergency is detected, of integrating patients into communities of cardiac patients that are also training at home and of tracking and comparing all the sessions of all the patients. The HR of patients is continuously measured, recorded and analysed during Exercise Training (ET), which is is one of the decisive and crucial factors for reducing and preventing unexpected cardiac events, with the help of wearable devices. Communities of patients are created for stimulating and motivating patients to perform according to their CR program. Opposed to traditional home-based CR e-Health applications, profiles and training zones are created and handled dynamically for each patient. The model presented in this case-study does not aim at replacing the well-known, usual CR programs, but is attempting to provide an alternative for patients who do not have the necessary means or time of doing CR programs in clinical environments or hospitals. This alternative should enable patients to adapt their lifestyle with little effort and

in safe and monitored conditions. Apart from answering to **RQ5**, this study follows 2 other objectives, i.e., the possibility of integrating multiple levels of medical expertise into smart devices for serving cardiac patients and the circumstances that would make smartwatches and smartphones alternative tools that will meet the needs of cardiac patients engaged in the second and third CR phases.

7.1 Related Background Information

Even though the death rate caused by heart diseases has decreased in the past years, the risks of heart failure are still very high, heart diseases being accountable for almost one third of the worldwide deceases [157]. People that suffered from a heart attack or have a heart disease/condition need to reduce the probability of a potential future fatal heart event and one of the main ways of doing this is CR.

7.1.1 Cardiac Rehabilitation

CR has many definitions [158][159], being seen as a set of key activities, including physical, social and mental factors, which slow, or even reverse, the evolution of the patient's disease or condition [160]. The benefits of CR include reducing the rate of mortality and unplanned hospital visits [161], lowering blood pressure and stress [162], improving the patient's exercise capacity and the overall quality of life [163]. Despite the major benefits of CR, the rate of participation in CR programs remains quite low [164]. To address this problem, new strategies for keeping the patients motivated and engaged in the CR programs need to be created. The reasons for the low attendance in CR programs differ from one phase to another. The process of CR is traditionally divided into four phases. Phase-1 starts at the hospital, phase-2 and phase-3 continue at the patient's home with periodical visits to the hospital, and phase-4 is the long term maintenance process for staying healthy.

This study focuses on phase-2 and phase-3 of the CR program, where the patients have to perform multiple sets of exercises in an environment that does not have medical equipment nor specialists to guide them, which is possibly less safe. Studies indicate that patients in phase-2 of CR are most vulnerable because they tend to do intensive exercises, if not guided correctly, and this can have devastating effects on their health [165] [166].

7.1.2 Wearable devices

Recently, major advances in wearable technology, such as sensor technology, communication technology and data analysis methods [167], give the possibility to assess the physiological health of patients, by collecting data in a

non-invasive and non-obtrusive way [168]. Wearable devices were successfully used as monitoring and alerting systems in case of high-risk cardiac patients [169], of analysing Parkinson patient's movement for detecting and preventing freezing of gait [170] and for estimations of heart and breathing rates from wrist motions [171]. The newest generation of smartwatches is now capable of measuring the HR very accurately with the help of Photoplethysmogram (PPG) technology [172]. The HR is a crucial health parameter measured in most of the CR programs.

7.1.3 Risk factors and challenges for home-based exercise training

When doing home-based exercises in a non-Electrocardiogram (ECG) environment, i.e., at home, during phase-2 and phase-3 of the CR program, cardiac patients are exposed to a series of risks, which may affect their safety. These risks include training at high intensity, outside the recommended limits, over-training or sudden increases or/and decreases of HR, known as arrhythmias or abnormal heart rhythms. Challenges for CR programs include establishing an adequate training program and training limits, personalised for each patient, defining complex profiles for each patient and handling emergencies in an appropriated way.

High-interval training versus moderate training

Prescribing safe exercises is not an easy task. There are many guidelines and recommendations for exercises for cardiac patients like the European Society of Cardiology's guidelines [173], American College of Sports Medicine's guide [174] or the recommendations of the European Association for Cardiovascular Prevention and Rehabilitation [175]. For example, for aerobic training, in [176], the authors recommend as intensity of the exercise 40-70% heart rate reserve or 50-80% maximal HR, as frequency 3-7 days per week and as duration 20-60 minutes per session or multiple 10-min sessions. However, it is hard to determine the *optimal* exercise program in terms of intensity and volume [177]. There is an ongoing discussion if training in higher intensity zones substantially improves the patient's peak oxygen consumption VO_{2peak} [178] and the exercise capacity for coronary artery disease patients [179]. Some studies have indicated, in the case of vigorous physical activity, that the risk of sudden cardiac death [180] or myocardial infarction [181] increases.

Over-training

Over-training appears when a patient exceeds the recommended training frequency, duration, intensity or all of them. Training more often has been shown to improve not only the (VO_{2peak}), but also the ventilatory anaerobic

threshold and quality of life [182]. But exceeding a certain limit of training sessions per week has minor effects on a person's physical condition [183] and can even become dangerous. The same applies to the duration of a training session. Some experts consider over-training more risky than not training at all.

Sudden changes of the Heart Rate

Patients with cardiac arrhythmias or with high risk of cardiac arrhythmias need to be closely monitored when training. Arrhythmias, or irregular heartbeats, may be completely harmless or life-threatening. A home-based CR system should be able to detect when a patient has an abnormal increase or decrease in the HR while exercising, as indicated in figure 7.3. Normally, irregular heartbeats are detected by ECG but evidence of detecting arrhythmias with a finger pulse sensor, which is also an optical sensor, was shown in [184].

Low engagement level

Inadequate training stimulation, the lack of immediate results and the long duration of training programs are only a couple of reasons for decreasing the patients' motivation and engagement in CR programs. Major beneficial effects can be seen only after longer periods of time [185] and this requires complex solutions, not only a monitoring device which shows some training limits. One solution, proposed in this study, is to organise patients into communities, based on similar characteristics and profiles, and to inform each patient about how the other patients in his/her community are performing.

Remote Monitoring Systems for Cardiac Rehabilitation

The increasing need and the high significance of remote monitoring systems for post cardiac patients has been emphasized in numerous studies [186][187][188][189]. A shift can be noticed in the CR technology, from self-monitoring, self-awareness and self-determination [188] towards automated monitoring [187], automated feedback systems [186] and real-time data processing [189].

In [186], the authors proposed a mobile solution for a home-based CR program, including a server-side application and a web portal for educational materials, for providing statistics or health reports and for discussion messaging. This solution was capable of automatically turning on the application on the mobile phone, of transmitting data to the server without the user intervention and provided an automatic data synchronization of new parameters, without changing the application. However, the patients still had to provide health parameters manually to the application, including the HR, exercising hours, sleeping hours, etc. The system was able to

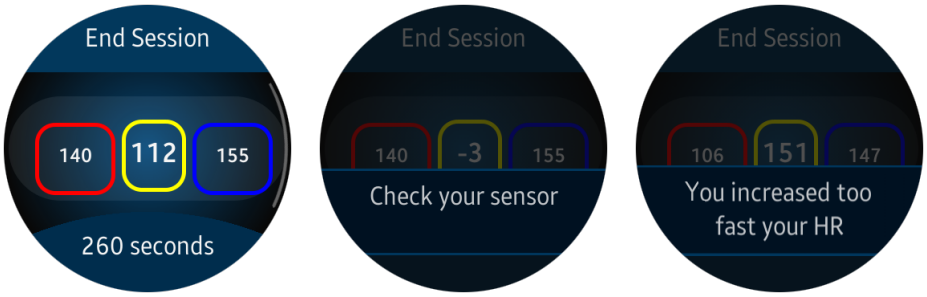
provide motivational Short Message Service (SMS), video and relaxation audio based on the received data. During the exercise sessions, patients were guided through traffic light indications. The patient's health condition was evaluated by transforming different activities like walking or cycling into metabolic equivalent of task hours and not by continuously measuring the HR. A more advanced system was proposed by Kyriacou et al. (2011), where the system is supporting the recording and transmission of health parameters such as ECG, HR, blood pressure, oxygen saturation and others, directly from the measuring devices [187]. Despite of the advanced technological support, the data aggregation is used only for telemonitoring, and, other functionalities such as sending personalized alarms or messages according to each patient's condition or establishing the risks zones of each patient in function of his/her medical parameters and/or past training sessions, are not included.

7.1.4 Wrist-worn devices for Cardiac Rehabilitation

Monitoring continuously health parameters like the HR is known to have an important impact in CR programs. In clinical environment and hospitals HR measurements are done using an ECG. But, outside hospitals, the use of an ECG apparatus is quite limited because of the high costs and remote houses of the patients. With the current technology on the market it is now possible to record the HR using smartwatches that are equipped with an optical HR sensor. In [190], the results of a performance evaluation on the optical HR sensors of an LG smartwatch showed that it is reasonably accurate to measure the HR via a wrist-worn device. Another study on remote monitoring, using a wrist-worn device, indicated that wrist-worn devices can be used in home telecare, telemedicine and for detecting emergencies [191].

7.2 The Architecture of the e-Health System

The architecture of the proposed e-health system consists of several components: (i) the smartwatch application, (ii) the smartphone application and (iii) the server-side applications. Each component is designed to add more functionalities according to its computing capacity. Each application can work on its own and is not dependent on the other ones. More precisely, if the application on the smartphone cannot connect to the server it will continue to work as a standalone application and will analyse the data coming from the application on the smartwatch. The same case applies if the smartwatch application loses its connection to the smartphone application and cannot transmit the data coming from the sensors. It will have limited processing capacity but it will still be able to measure and display different parameters like the duration of the training, the recommended training limits and the current HR.



(a) The main screen of the application during ET.

(b) A notification when the sensor does not detect a valid HR value.

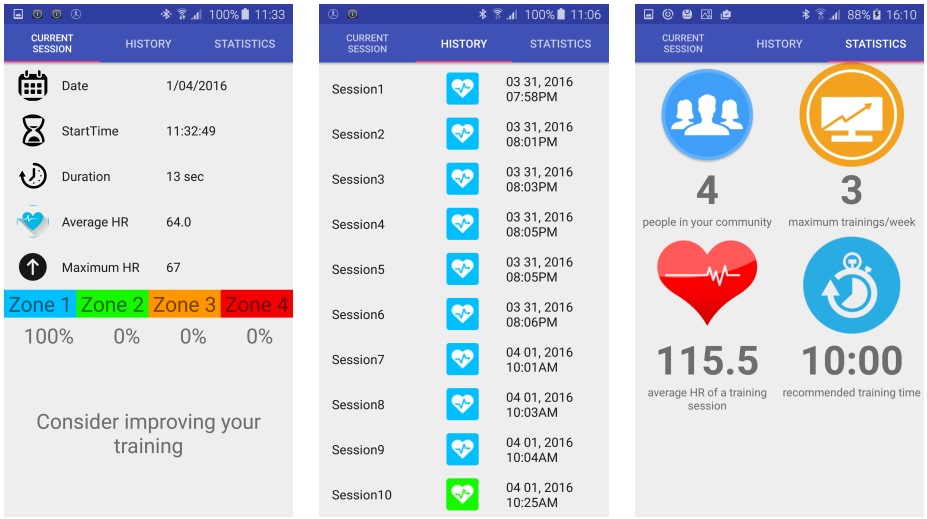
(c) Alerting the user.

Figure 7.1: The GUI of the application on the smartwatch Gear S2 from Samsung.

7.2.1 The Prototype Application on the Smartwatch

Commercial smartwatches like the Gear S2 from Samsung offer programmers the possibility of creating applications which have access to a variety of sensors on the smartwatch such as the accelerometer, gyroscope, HR monitor, barometer or ambient light. In order to track the activity and measure the HR of the patients involved in the CR program, a new prototype application was created for this study. The application is able to monitor and display continuously the HR using the PPG sensor of the watch, also known as an optical sensor, to display messages and notifications on the screen, to vibrate in case of alerts and notifications, to send raw data to the smartphone and to receive information from the smartphone via a Bluetooth connection.

When the user starts the application, by pressing on the screen of the smartwatch on the application's icon, a new session starts to be recorded. The values of the HR are written to 2 files, i.e., *slice.txt* and *session.txt*, which are stored locally on the internal memory of the smartwatch. New data is written into *slice.txt* each 10 seconds and is sent to the smartphone for processing. The *session.txt* file is recording the HR during the entire training session and, at the end of the session, is sent to the smartphone for further processing. The need of storing data into 2 different files is motivated by the case when the smartwatch loses its connection with the smartphone and cannot send the *slice.txt*. In this case the corresponding data will be stored in the *session.txt* file and will be sent at the end of the session. The application is designed to continue running in the background during the training session, even if the user does not interact with it for a relatively long period of time. If the user does not close the application, it will close automatically if the HR is equal to (-3) for more than 30 seconds. The value (-3) is obtained from the optical sensor in case the UV light does not encounter any object in front of the sensor, meaning that the smartwatch



(a) Screen1 - The summary of the last training session.

(b) Screen2 - A list with previous training sessions.

(c) Screen3 - Statistics about the patient and his/her community.

Figure 7.2: The GUI of the smartphone application

may have been taken off the hand of the patient. Otherwise it displays (0) in case it detects an object but it cannot retrieve a proper HR value.

The application consists of one main screen, as presented in figure 7.1a, where the current HR is shown in the centre of the screen, together with the time elapsed for the current training session and with the recommended training limits. On top of the main screen several notifications can be displayed, as seen in figure 7.1b and figure 7.1c. For example, in case a patient starts his/her training session and the smartwatch cannot detect the HR, a notification with the text message **"Check your sensor"** will be displayed on the screen, together with a vibrating alarm. Another example, in case the patient increases the intensity of his/her training too fast, a vibrating alarm will be triggered and a notification displayed with the message **"You increased too fast your HR"**. These alarms or notifications are programmed to stay on the display of the smartwatch for 5 seconds, afterwards they disappear from the screen. The application is meant to show only one notification at the time.

The interaction of the patient with the screen of the smartwatch was reduced to a minimum because the patient should be more focused on exercising correctly than on interacting with the application. Nevertheless, when a training session is completed, patients can check on the application on their smartphones its details together with other relevant information.

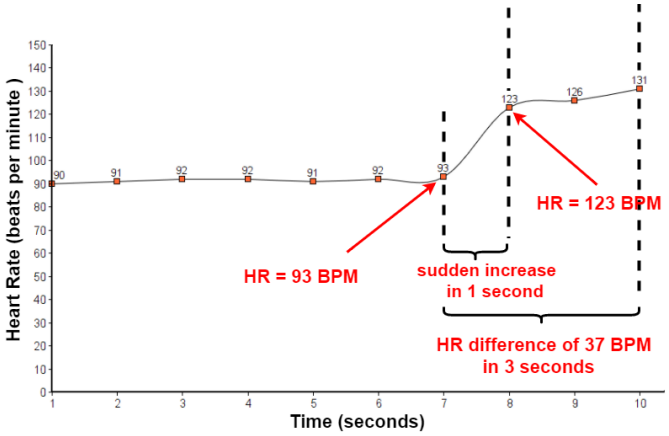


Figure 7.3: Sudden increase of the HR during a training session

7.2.2 The Prototype Application on the Smartphone

A prototype application for Android-based mobile phones is created in order to acquire raw data from the watch and to process it. The application is receiving raw data each 10 seconds, via Bluetooth, from the smartwatch application when an exercise session is started via a text file called *slice.txt* and, at the end of the exercise session, via a text file called *session.txt*. If a slice is missed, the training session can be reconstructed based on the session file. The other way around works as well, meaning that if the session text file is not received, the session's data can be reconstructed from the slice text files. The specifications of the smartwatch and its operating system do not make it a candidate for supporting middleware architectures, like a PE, and for computing complex rules, while the smartphone qualifies perfectly for this job.

The application has multiple screens where the patient can consult the details of his/her last training session, see a list with previous sessions and see statistics about themselves and their communities. In figure 7.2a, the main screen of the application, in the upper side of the screen, the patient can check if he/she respected the recommendations established before, at the hospital, in terms of duration, frequency and intensity of a training session. In the lower part of the screen, coaching messages are displayed to the user for guiding or advising purposes. It will either tell the patients to improve their training sessions or to continue, in case the exercise session was according to the medical guidelines. The second screen of the application contains a list of previous sessions, presented in figure 7.2b, and, when clicked on one of them, a new screen which contains the summary of the clicked session will open. And, finally, figure 7.2c shows the last screen which contains multiple facts and statistics, such as the number of people in the patient's community which are also participating in the same CR program, the average HR of

```
1: procedure LPaS1(R31)                    ▷ Analyses data at each iteration
2: dataAcquisition:
3:     boolean smartwatchConnected ← engine.isWatchConnected()
4: activationGuards:
5:     return true
6: conditions:
7:     return smartwatchConnected
8: actions:
9:     hrValues ← engine.getLatestHRValues()
10:    for for i = 0 to hrValues.size() do
11:        for for j = i to hrValues.size() do
12:            if (hrValues[j]-hrValues[i]>30) and (i+j<9)
13: and hrValues[j]>0 and hrValues[i]>0 then
14:                sendMessageToGear("Take a break, you increased your
    HR too fast!")
15:            else if (hrValues[j]-hrValues[i]<30) and (i+j<9)
16: and hrValues[j]>0 and hrValues[i]>0 then
17:                sendMessageToGear("Take a break, you decreased your
    HR too fast!")
18: rulesGeneration:
19:    createRule(this)
20: sendMessageToGear(msg):                    ▷ Additional method
21:    payLoad ← sender.buildMessage("title", "shortDescription", msg,
    "1", "break", GearMessageSender.GEAR_ ACTION _ CANCEL)
```

Figure 7.4: Proactive Rule R31 of Local Scenario LPaS1 in pseudo-code.

all the previous training sessions, the maximum training times of a single patient from the community and the hour recommended to begin training. Communities are created on the server side for tackling the problem of low patient motivation and engagement discussed in section 7.1.3. This can create extra motivation factors which will increase the participation of patients in their CR programs. Note that this last screen currently contains only a basic set of statistics and facts and will be easily extended in the future.

The most important part is how the application works and how it is designed to analyse the data. The application utilises the platform presented in chapter refimplementation, which processed and executed PaSs periodically, e.g., each 3 seconds. Several GPaSs and LPaSs were developed for achieving the various goals of this application.

Proactive Scenarios for the Mobile Application

The smartphone application includes 4 LPaSs and 2 GPaSs. The convention for naming the PaSs on the mobile application applies a number at the end

of the name of the PaS, followed by the letter *"m"*, coming from the word "mobile". This avoids possible confusions with the PaSs on the server side.

LPaS1m was developed for detecting sudden HR changes during exercise sessions, *LPaS2m* for checking the intensity of the training sessions, *LPaS3m* for establishing training zones and how much does the patient train in each zone and *LPaS4m* for creating coaching messages for the patient. LPaS1m is in charge of detecting if a patient suddenly increases or decreases his/her HR during a training session. If the patient is at 110 Beats Per Minute (BPM) and is arriving at 150 PPG in less than 5 seconds and it stays high or continues to increase, as shown in figure 7.3, then, the cardiac patient can have a serious problem. Or, it can be caused by the sensor, which can misread the correct value of the HR, or it can be caused by the communication channel between the smartphone and the smartwatch, which can lose data. The complexity of this problem is further described in [192] or [193]. The code of the PR which represents *LPaS1m* in shown in figure 7.4.

The two global scenarios, i.e., *GPaS1m* and *GPaS2m*, contain rules on both sides, on the smartphone's side and on the server's side. *GPaS1* was developed for dynamically adjusting the training limits of a patient, while *GPaS2m* was created to handle the communities of patients. A concrete example of the implementation of *Global Scenario 2* is given in Section 7.2.5. The PaSs used for the study were discussed together with doctors and medical specialists working at the 2 local hospitals.

7.2.3 The Server-side Layer

As depicted in figure 3.4 shown in chapter 3.4, the architecture of the server-side layer is divided into several parts: the PE, the Relay Server, the local database and a website. While the mobile application is in charge of handling local data related to each patient, the server-side applications are responsible for collecting, keeping track and analysing all the processed data coming from all the mobile phones. More specifically, the Relay Server is used for receiving and sending messages from and to mobile phones. The website contains a personalised profile for each patient and is in charge of displaying statistics related to his/her previous exercise sessions. A doctor can access the results of the exercise session of any patient he/she is in charge of, he/she can compare it with the other patients' sessions and can update the profile of each patient, thus increasing or decreasing the safety limits of a patient's HR during training. And, finally, the PE, a powerful Java rule-based engine presented in chapter 3.4, is in charge of taking actions based on the information it receives.

Proactive Scenarios for the Server Application

To take advantage of the global knowledge available on the server, several local and global PaSs were developed. *LPaS1s* is in charge of periodically generating medical reports containing the evolution of a patient in the CR program from the beginning until the time of checking. Information is available regarding the number of trainings per week, the intensity and the duration of trainings, the time spent in each training zone, for each session, the date and the starting hours of the exercise sessions. The reports generate a PDF file which is available to the specialist or doctor via the website. This scenario does not wait for the doctor's explicit command but is building and sending the report automatically. The other scenarios are global, one for the dynamic profile of the patients, presented in details section 7.2.5, one for handling the communities, one for providing extended medical coaching and another one for continuing the work of LPaS1s.

GPaS2s checks which patients are not exercising according to their CR program and sends messages to encourage them. These messages can contain statistics about their communities or about certain patients in their communities. *GPaS3s* performs one analysis on how much each patient stayed in each training zone, e.g., how much time they were training at 55% of their maximum HR, compares it with previous sessions of the patient and decides if the patient needs to increase the training time spent in one training zone or decrease this time, in case the training zone was too high, e.g., 5 minutes at 70% of the maximum HR. And *GPaS4s* is in charge of sending by email periodical reports, containing patient relevant information regarding the training frequency, intensity and duration, to the specialists in charge of the patients.

Communities of patients

Communities of patients with different characteristics are created and managed on the server side via the web interface by the medical experts. The possibility of automatically generating the communities, based on the profile of the patients, exists and can be done with the help of extra PaS as it was done in the first case study. However, as it was not the main focus of this study, it was left to be manually handled and supervised by the medical experts. So, in this example, with the current settings, an expert could decide to create a community of male patients with the same cardiac affection, with ages between 50 and 55, if there were at least 3 patients with these characteristics. The supervision of the communities is done by the medical experts via the web interface and via the periodical reports, which contain more specific data about each patient compared to his/her community.

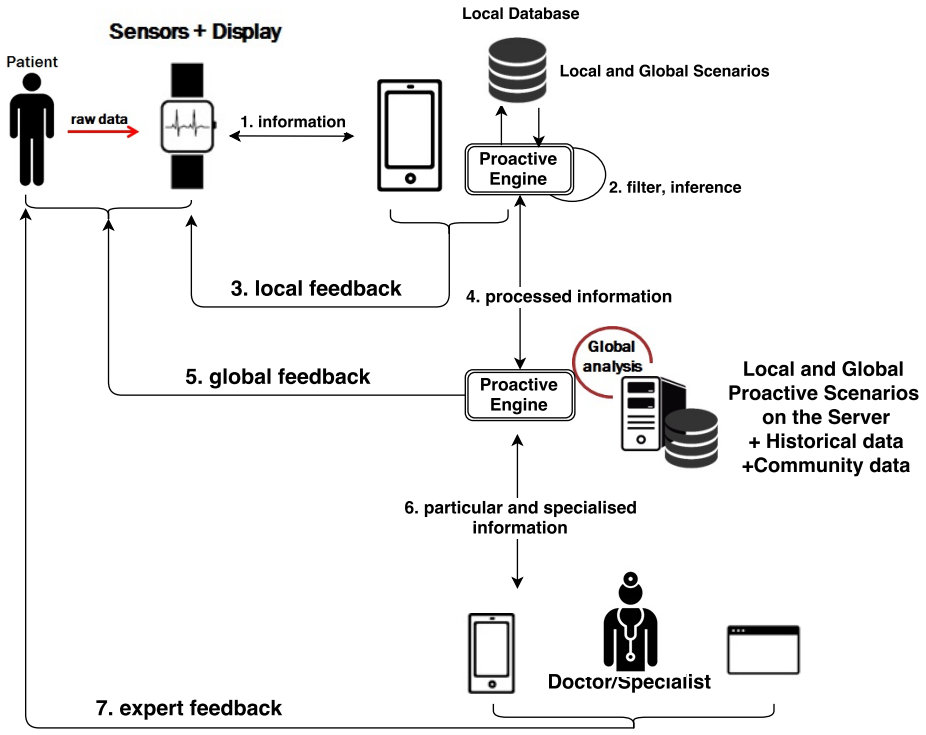


Figure 7.5: The architecture of the Proactive E-Health System with multiple levels of monitoring and expertise

7.2.4 Multiple Levels of Feedback and Monitoring

Real-time feedback is a very important aspect when working with patients with heart conditions. A simple alert can warn a cardiac patient if he/she is doing something wrong and this can avoid accidents, even saving his/her life in the given circumstances of CR. Different types of expertise are available to patients when doing the CR exercises, as shown in figure 7.5. First, a local level of expertise is embedded into the smartwatch and the mobile phone thus providing local feedback, the second level, more extensive and based on a broader context, comes from the server and the third level comes directly from the medical experts and their personal devices.

Local level of expertise

The applications on the smartwatch and on the mobile phone are able to acquire data, analyse it locally and provide patients with immediate feedback. On the mobile phone, the prototype application is performing more complex analyses with the help of PRs, which contain a set of minimum medical knowledge. This first loop between the patient, the sensors of the smartwatch, its screen and the mobile phone offers the necessary local medical

expertise.

Global level of expertise

At a more global level, on the server side, data is gathered from all the smartphones of all the patients. At this stage, a more powerful PE in terms of computing capacity is available for analysing and aggregating information from all the smartphones. As described in section 7.2.3, the computing capabilities of a similar PE are described in [79]. The full data set of the patients is stored on the server, thus offering the possibility of analysing historical data and of data mining techniques. Progressive monitoring is also available on the server's side, meaning that monitoring parameters can change after more information about each patient is gathered.

The specialist's direct advice

This level of feedback was designed to keep the doctors in the loop even when they are not physically present at the hospital. They are informed on their personal devices, e.g., mobile phones, if patients were up to date with their CR program, they receive notifications about certain patients with more severe forms of heart conditions and they are alerted in case a patient had problems during his/her training sessions and did not stop according to the indications of the local PS. In this last case, one solution is to immediately call the patient to discuss with him/her, to see what happened exactly and to take measures to prevent any unwanted heart event.

7.2.5 Dynamic Patient Profiles

As discussed in section 7.1.3, establishing training limits for each patient is a big challenge, as training outside the correct limits can be dangerous, even fatal. A post event symptom-limited graded exercise test at the hospital, before the patient's discharge, has to be done for determining the exercise capacity of each patient. More precisely, it measures the peak HR ($measuredHR_{max}$) of a patient and determines the *Anaerobic Threshold (AT)*, which is the point where energy production is supplemented by anaerobic mechanisms [194]. Another way of defining it is the exertion level between anaerobic and aerobic training. AT is very important because of the differences in physiological responses of patients during training below and above AT. Cardiac patients have lower values for AT because of their conditions.

Available applications do not offer personalised training boundaries, i.e., target HR zones, for cardiac patient. For example, Polar gives the users the possibility of manually setting the training limits or of performing a test session with the *OwnZone* program set to determine the limits of each user [195]. Training zones are determined by the sensors on the chest strap

connected to the watch or, if the HR rose too fast during the measurements, on age-based algorithms. But, if health changes appear for certain patients, e.g., they start to take medication, the limits and training zones are not updated accordingly.

The solution is to provide a GPaS to dynamically calculate the profile of each patient. This task was assigned to GPaS1s. When the mobile application is installed it comes with training limits calculated by default, as if the patient would not have an effort test. Then, the server pushes to the patient's smartphone necessary information for creating his/her training zones and HR limits such as age, anaerobic threshold if existing, maximum measured HR or if the patient is under treatment or not. Afterwards, these limits are automatically pushed to the smartwatch of the patient. The medical specialist, who controls the profile of each patient remotely via a website, can change certain parameters and so, the profile is then calculated again according to the new parameters. For example, initially, a profile does not have an anaerobic threshold value included, neither a maximum measured HR and so, his/her training limits will be calculated based on his/her age. As soon as the specialist introduces the AT's value, the limits are adjusted automatically, so, when the patient will start his/her next exercise session, the new limits will be displayed on the smartwatch.

Several cases can be distinguished when establishing training limits for cardiac patients, as the one for endurance training, where the training interval is equal to 70%-85% of the $measuredHR_{max}$ [196]. For this study, target heart rate (THR) zones were established for performing aerobic exercises. In case the $measuredHR_{max}$ of a patient is known and the anaerobic threshold is not, and the patient is under treatment, then the recommended THR interval is between 65% - 90% of the $measuredHR_{max}$. Moreover, if the patient is not under treatment but values are known for his/her $measuredHR_{max}$ and AT, the THR interval becomes $[AT - 10bpm]$ for the lower limit and $[AT + 5bpm]$ for the upper limit. On the other hand, if the patient does not have an AT neither a $measuredHR_{max}$ and is under medication, then the THR interval is between 55% and 77% of the $theoreticalHR_{max}$, which is determined by the well-known formula (220 - age) [197]. Otherwise, if he/she is not under medication, the lower limit and upper limits are set to 65%, respectively 90% of the $theoreticalHR_{max}$.

7.3 System Testing and Evaluation

For testing and evaluating the proposed system in a real world scenario, an experimental study was performed. The study involved collecting data from 5 men from our team, at the local university, for a duration of 8 weeks. The low number of people and the fact of not using real cardiac patients resulted from the difficulties of obtaining an authorization for working with

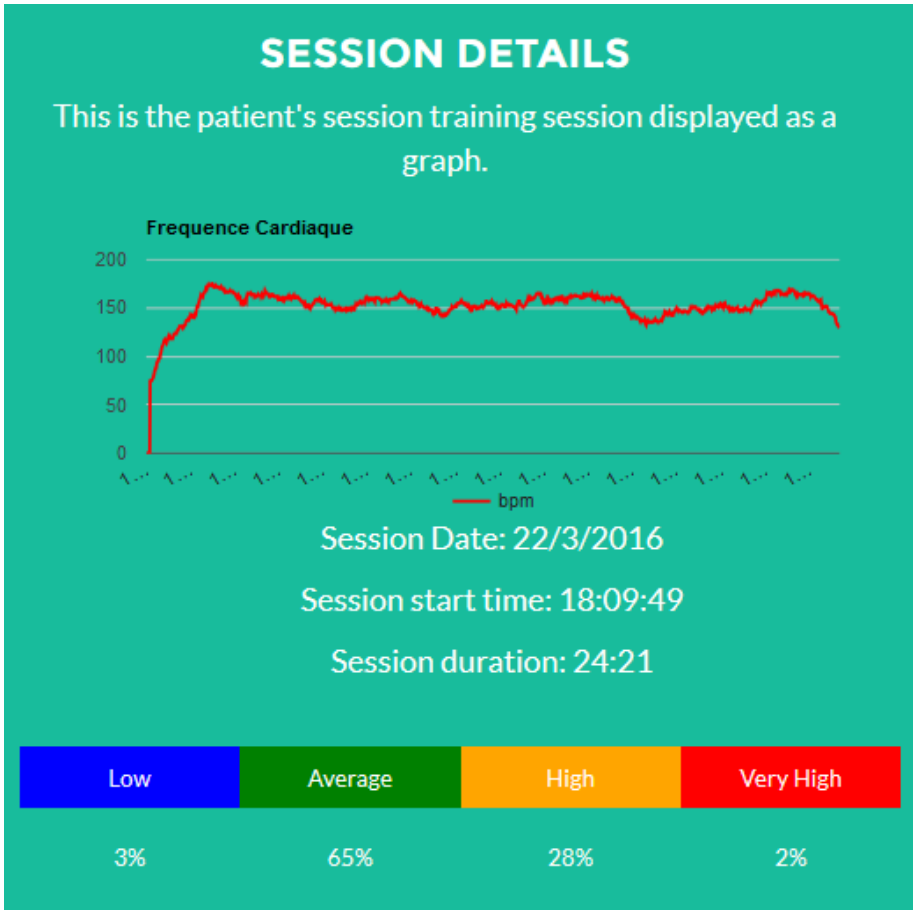


Figure 7.6: A 25 minutes training session which was registered on the server-side without any errors.

real patients involved in a CR program, which takes minimum 1 year in Luxembourg.

The participants were instructed, at the beginning of the study, to register training sessions when performing physical activities during the day. Their age ranged from 21 to 48 years old, being part of the same community. They did not take any medication during the period of the experiments. So, the recommended exercise training limits for each participant was calculated using the formula discussed in the Section 7.2.5, i.e., 65-90% of the $theoreticalHR_{max}$, (220 - age). Each person participating in the experiments was given a Samsung Galaxy S6 smartphone and a Gear S2 martwatch from Samsung. A unique email address was assigned to each of the 5 participants. Each smartphone was equipped with a PE and with the prototype application running on top of it. A special Bluetooth channel for exchanging data was defined between the devices for not interfering with

other Bluetooth enabled devices.

A total of 90 valid training sessions were registered in the server's database. A training session was considered valid if it contained values bigger than 10 PPG. The smartwatch was able to register continuously the HR during entire training sessions, as the one shown in figure 7.6, without any loss of data. Figure 7.6 shows one example of a usual training session, starting from (0 bpm), when opening the application, moving up to around (50 bpm), when a correct HR value was initially detected, and then continuing to increase up to a value in between (150 bpm) and (165 bpm). This intensity interval was then kept for the rest of the training session.

Changes on the profiles were performed several times and limits on the smartwatches were adjusted either during training sessions or later on, when starting a new training session, thus checking the PaSs regarding the patient profiles on the server and on the smartphone. The participants reported multiple times the appearance of alarms and notifications when they were overpassing the recommended training limits, when they were not training at the recommended frequency and when the sensor could not detect a HR value. Coaching messages were displayed after each session and were kept visible until the start of new session, thus validating the local scenarios on the smartphones. A few cases were detected of smaller sessions containing non-valid HR values, followed by longer training sessions with valid HR values. This fact occurred when the sensor of the smartwatch was not initially placed correctly and when the smartwatch application was restarted.

7.4 Conclusion

The main goal of this study was achieved by providing a monitoring and coaching RBPS that allows patients, participating in a home-based CR program, to train in safe limits, to improve their health condition and to develop a long term habit of exercising, while keeping the medical experts in the supervising role. Also, the RBPS makes it possible to take CR medical knowledge and to integrate it, with the help of PaC, on wearable devices, on mobile phones and on remote servers, thus make another contribution towards validating the **RQ5**. The PaSs used to build the applications show the ability of the e-Health system to anticipate various situations, to detect contextual information and to adapt its behaviour accordingly, thus addressing **RQ2**. Furthermore, by using GPaSs in this complex e-Health situation, another practical answer is provided to **RQ3**.

The tests showed promising results in favour of using wearable devices as suitable tools for registering entire training sessions, mobiles for analysing single patient-based data and servers for aggregating data from entire communities of patients, hence, verifying the second research question. This study also addresses a topic which is becoming increasingly popular, i.e.,

the utilisation of wearable sensors in e-Health rehabilitation applications. The model introduced in this case study complements the traditional CR programs that take place in specialized clinics, while offering patients an alternative for their rehabilitation training, while maintaining the connection patient-doctor. At the same time, by providing patients with a proactive automated wearable and mobile application, the patient's interaction with the system is reduced to a minimum, thus, allowing him/her to focus more on his/her training sessions. The objective of this model to expand, support and increase the utilization of CR programs was also achieved.

Chapter 8

Conclusions

The rule-based model presented in this thesis is not only able to provide a robust and simple way of computing, thus benefiting the developer, but is also able to compute proactively, for and on behalf of the user, to perform complex computations, to provide a communication layer for achieving collaboration, to detect and acquire relevant context-information and to actively seek for the information it needs. It is a development environment or a framework with which developer can create smart applications capable of running on multiple platforms.

In addition, the Proactive System can express and anticipate complex situations with the help of Proactive Scenarios(PaSs). Not only Proactive Scenarios can accurately describe these situations but they provide adequate measures or actions to handle them.

All the research questions, presented in chapter 1.2, were addressed and each one of them was answered. More precisely, **RQ1** was answered by analysing current computing choices and by providing arguments for choosing rule-based systems to represent and reason about knowledge and for being the base for implementing Proactive Systems.

The core components composing the modular architecture of a Rule-Based Proactive System were identified and the functionality of each component described. It was specified how these components work with each other in order to build a functional Proactive System on platforms running different operating systems.

Furthermore, the main properties of a Proactive Systems were portrayed and explained. These properties are essential when addressing an important category of current modern software applications. This helps answering **RQ2** and provides the necessary settings for having multiple environments where Proactive Systems are functioning. Moreover, the possibility of having various Proactive Systems communicating and exchanging information was considered. A communication protocol between them is presented and the content of their communication is specified and described, thus responding

to **RQ3**.

Because having a communication protocol and relevant content to send is not enough to build complex collaborative systems, a strategy for achieving collaboration between Proactive Systems is proposed and used. A strategy called Global Proactive Scenarios (GPaSs) was proposed for automating the communication and coordination between Proactive Systems, for providing a technique to achieve collaboration and to make multiple Proactive Systems to work together.

Following the design of local Proactive Scenarios, which are in charge of sequences of events that occur to/on a single system and a set of corresponding actions, Global Proactive Scenarios are taking advantage of the large amounts of information that is available from multiple devices or computers equipped with Proactive Engines. This technique opens new possibilities for Proactive Systems to explore. Besides from offering the solution to **RQ4**, Global Proactive Scenarios also deal with important challenges in RBSs.

For supporting the proposed theoretical concepts and to tackle the last research question, i.e., **RQ5**, which was more an exploratory research question, three case studies were described. These case studies included various applications for addressing issues in domains like e-Learning, business and e-Health. Different domains were selected to emphasise the potential of the model and techniques presented in this dissertation. Moreover, the software systems and applications developed in the case studies were specially selected in an ascending order of their complexity to point out the broad range of situations where they can be successfully applied.

The results of each case study are a proof that Proactive System accomplished to achieve *correctness*, *termination* and *response time* of Proactive Scenarios, which are important standards in rule-based systems.

In the first case study, the use of Local Proactive Scenarios were enough for creating, managing and evaluating social groups, and for transforming the local e-Learning system into a Proactive System. Experiments showed that the given set of Local Proactive Scenarios were enough for handling social communities for more than 1000 students. This included the generation of resources related to each community, of coaching messages and alerts, and of monthly reports for each group. It was considered the most basic example, not because of the the complexity of the software system which was enhanced with proactive capabilities, but because there were no other PSs involved.

The second case study uses a mobile application, i.e., SilentMeet, which is based on two Global Proactive Scenarios to solve an issue which is encountered quite often in many situations, i.e., how to mute the smartphone of people that are participating in a private event, where it is required to silence all devices that may interrupt the event. These scenarios involve several rounds of negotiation and use the location of the participants to establish if a meeting will take place or not. The users of *SilentMeet* benefit

from the fact that several steps of establishing a meeting and silencing the devices are automated and they do not have to take part in each step of the whole process.

And, the third case study, where all the theoretical concepts proposed in this dissertation are involved, demonstrates even more the great potential of RBPSs. Not only this last example shows how the Proactive Engine and several Local Proactive Scenarios are implemented successfully on top of the operating system of mobile devices and desktop computers but is shows how, by working together and collaborating, they achieve the desired output which cannot be achieved only by mobile devices or only by desktop computers. This example also shows how important aspects in *cardiac rehabilitation* can be addressed with the help of the proposed technology in this thesis.

These case studies also show that GPaSs can use different types of collaboration algorithms. For example, a GPaS can use a partial collaboration algorithm, where a minimum number of PEs have to agree or respond but not all the PEs involved in the collaboration process, or a complete collaboration algorithm, where all the PEs have to give their consent in order to proceed. A partial collaboration algorithm is proposed in the second case study, when using *SilentMeet*, where, in order to have an actual meeting, it is enough to have two people that confirmed the meeting at the same location of the meeting. A full collaboration algorithm is shown in the third case study, where all the Proactive Systems collaborate together for achieving the desired output.

8.1 Future Perspectives

The work presented in this dissertation opens two main research directions with multiple objectives. One concerns the framework for developing and implementing Proactive Systems, while the other one refers to its fields of application.

The first research objective regarding the proposed model or the rule-based Proactive System is to look for a version of the Proactive Engine which would take into account asynchronous events or which would benefits from runtime optimisation methods such as multi-threading and/or different scheduling approaches for Proactive Rules. A second possible objective would be integrating *Machine Learning* algorithms into the Proactive System, which would allow Proactive Scenarios to adapt and to improve their actions and functionalities by conducting various analyses on past experiences and events. And, a third objective for the first research direction, would be to propose other means for supporting cooperation and collaboration between multiple Proactive Systems.

For the second research direction, the first objective could be the pro-

posal of several methods for designing patterns for Proactive Scenarios regardless of their type. Having a design pattern would further help developers to arrange and link Proactive Rules. The second objective would allow Proactive Scenarios to develop properties for the Proactive System like *self-optimisation* or *self-healing*, concepts related to more advanced computing systems. Collaborative Proactive Systems open new perspectives for computing systems that spread around in all kinds of environments. The number of devices supporting these computing systems are increasing very fast. Thus, it would be useful to see how large networks of Proactive Systems would behave and how would their collaboration work in such dynamic and unpredictable conditions. One way to achieve it would be to create a framework or a tool that would simulate the behaviour and the functions of multiple Proactive Systems. This approach and possible third objective would be an alternative to using real-world scenarios with hundreds or thousands of involved devices. Another direction for collaborative Proactive Systems would be to explore multiple application domains as the real of smart applications that need to benefit from a certain degree of automation and of proactive behaviour is immense and extends beyond the one presented in this dissertation.

List of My Publications

R. A. Dobrican and D. Zampunieris, "Supporting collaborative learning inside communities of practice through proactive computing," in *Proceedings of the 5th annual International Conference on Education and New Learning Technologies, Barcelona, Spain 1-3 July, 2013*, pp. 5824–5833, IATED, 2013.

R. A. Dobrican, S. Reis, and D. Zampunieris, "Empirical investigations on community building and collaborative work inside a lms using proactive computing," in *Proc. E-learn*, vol. 1, pp. 1840–1852, 2013.

R.-A. Dobrican and D. Zampunieris, "Moving towards a distributed network of proactive, self-adaptive and context-aware systems," in *ADAPTIVE 2014-The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications*, pp. 22–26, 2014.

R. A. Dobrican and D. Zampunieris, "A proactive approach for information sharing strategies in an environment of multiple connected ubiquitous devices," in *Proceedings of the International Symposium on Ubiquitous Systems and Data Engineering (USDE 2014) in conjunction with 11th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC 2014)*, pp. 763–771, IEEE, 2014.

R. A. Dobrican, G. Neyens, and D. Zampunieris, "Silentmeet-a prototype mobile application for real-time automated group-based collaboration," in *Proceedings of the 5th International Conference on Advanced Collaborative Networks, Systems and Applications (COLLA 2015)*, pp. 52–56, IARIA, 2015.

G. Neyens, R. A. Dobrican, and D. Zampunieris, "Enhancing mobile devices with cooperative proactive computing," in *Proceedings of the 5th International Conference on Advanced Collaborative Networks, Systems and Applications (COLLA 2015)*, pp. 1–9, IARIA, 2015.

R. A. Dobrican and D. Zampunieris, "A proactive solution, using wearable and mobile applications, for closing the gap between the rehabilitation team

and cardiac patients," in *Proceedings of the IEEE International Conference on Healthcare Informatics 2016 (ICHI 2016)*, pp. 146–155, IEEE, 2016.

R.-A. Dobrican and D. Zampunieris, "Moving towards distributed networks of proactive, self-adaptive and context-aware systems: a new research direction?," *The International Journal on Advances in Networks and Services*, vol. 7, pp. 262–272, 2014. ISSN: 1942-2644.

R. A. Dobrican, G. Neyens, and D. Zampunieris, "A context-aware collaborative mobile application for silencing the smartphone during meetings or important events," *International Journal On Advances in Intelligent Systems*, vol. 9, no. 1&2, pp. 171–180, 2016.

Bibliography

[1] Object Management Group (OMG)., "Production rule representation version 1.0," 2009, [Online; accessed 8-September-2016]. [Online]. Available: http://www.omg.org/spec/PRR/1.0/PDF/

[2] H. Boley, A. Paschke, and O. Shafiq, "Ruleml 1.0: the overarching specification of web rules," *Lecture Notes in Computer Science*, vol. 6403, no. 4, pp. 162–178, 2010.

[3] V. Serrano and T. Fischer, "Collaborative innovation in ubiquitous systems," *Journal of Intelligent Manufacturing*, vol. 18, no. 5, pp. 599–615, 2007.

[4] F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa, "Jade programmers guide," *Jade version*, vol. 4.0, 2010. [Online]. Available: http://jade.tilab.com/doc/programmersguide.pdf

[5] A. Ligeza, *Logical foundations for rule-based systems*. Springer, 2006, vol. 11.

[6] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[7] D. Tennenhouse, "Proactive computing," *Communications of the ACM*, vol. 43, no. 5, pp. 43–50, 2000.

[8] M. Zhao, Y. Ye, Y. Han, Y. Xia, H. Zhu, S. Wang, Y. Wang, D. A. Muller, and X. Zhang, "Large-scale chemical assembly of atomically thin transistors and circuits," *Nature Nanotechnology*, 2016.

[9] I. PRESENT, "Cramming more components onto integrated circuits," *Readings in computer architecture*, vol. 56, 2000.

[10] C. V. N. Index, "Global mobile data traffic forecast update, 20152020," *Cisco white paper*, 2016.

[11] I. D. C. (IDC), "IDC Forecasts Worldwide Shipments of Wearables to Surpass 200 Million in 2019, Driven by Strong Smartwatch Growth

and the Emergence of Smarter Watches," *Press Release*, Mar 2016. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS41100116

[12] D. Evans, "The internet of things how the next evolution of the internet is changing everything. cisco white papers, 2011."

[13] R. A. Dobrican and D. Zampunieris, "Moving towards distributed networks of proactive, self-adaptive and context-aware systems: a new research direction?" *The International Journal on Advances in Networks and Services*, vol. 7, pp. 262–272, 2014, iSSN: 1942-2644.

[14] D. Zampunieris, "Implementation of a proactive learning management system," in *Proceedings of" E-Learn-World Conference on E-Learning in Corporate, Government, Healthcare & Higher Education"*, 2006, pp. 3145–3151.

[15] N. Alechina, B. Logan, N. H. Nga, and A. Rakib, "Verifying resource requirements for distributed rule-based systems," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web*. Springer, 2008, pp. 31–38.

[16] E. Kasanen, K. Lukka, and A. Siitonen, "The constructive approach in management accounting research," *Journal of management accounting research*, vol. 5, p. 243, 1993.

[17] R. A. Dobrican and D. Zampunieris, "Moving towards a distributed network of proactive, self-adaptive and context-aware systems," in *ADAPTIVE 2014-The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2014, pp. 22–26.

[18] "Adaptive 2014 awards - awarded papers," [Retrieved: August, 2016]. [Online]. Available: http://www.iaria.org/conferences2014/AwardsADAPTIVE14.html

[19] T. H. D. Varun Grover, "General perspectives on knowledge management: Fostering a research agenda," *Journal of management information systems*, vol. 18, no. 1, pp. 5–21, 2001.

[20] J. E. Rowley, "The wisdom hierarchy: representations of the dikw hierarchy," *Journal of information science*, 2007.

[21] R. Brachman and H. Levesque, *Knowledge Representation and Reasoning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[22] J. R. Anderson, *Cognitive skills and their acquisition*. Psychology Press, 1981.

[23] E. Turban, J. Aronson, and T.-P. Liang, *Decision Support Systems and Intelligent Systems 7 Edition*. Pearson Prentice Hall, 2005.

[24] Procedural knowledge, "Procedural knowledge/contexts/artificial intelligence — Wikipedia, the free encyclopedia," 2016, [Online; accessed 7-September-2016]. [Online]. Available: https://en.wikipedia.org/wiki/Procedural_knowledge

[25] L. W. Anderson, D. R. Krathwohl, and B. S. Bloom, *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Allyn & Bacon, 2001.

[26] H. Allert, "Coherent social systems for learning: An approach for contextualized and community-centred metadata," *Journal of interactive Media in Education*, vol. 2004, no. 1, 2004.

[27] M. P. Singh and M. N. Huhns, *Service-oriented computing: semantics, processes, agents*. John Wiley & Sons, 2006.

[28] J. Mylopoulos and H. Levesque, "An overview of knowledge representation," in *GWAI-83*. Springer, 1983, pp. 143–157.

[29] M. Minsky, "A framework for representing knowledge," 1974.

[30] J. Sowa, "Knowledge representation: Logical, philosophical, and computational foundations," *Book in preparation. To be published by PWS Publishing Company, Boston, Massachusetts*, 2000.

[31] R. Davis, H. Shrobe, and P. Szolovits, "What is a knowledge representation?" *AI magazine*, vol. 14, no. 1, p. 17, 1993.

[32] E. Mercier-Laurent and D. Boulanger, *Artificial Intelligence for Knowledge Management: First IFIP WG 12.6 International Workshop, AI4KM 2012, Montpellier, France, August 28, 2012, Revised Selected Papers*. Springer, 2014, vol. 422.

[33] Y.-J. Hu, C.-L. Yeh, and W. Laun, "Challenges for rule systems on the web," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web*. Springer, 2009, pp. 4–16.

[34] N. Matsatsinis and Y. Siskos, *Intelligent support systems for marketing decisions*. Springer Science & Business Media, 2012, vol. 54.

[35] M. A. Mach and M. L. Owoc, "Knowledge granularity and representation of knowledge: Towards knowledge grid," in *International Conference on Intelligent Information Processing*. Springer, 2010, pp. 251–258.

[36] D. Li and Y. Du, *Artificial intelligence with uncertainty.* CRC press, 2007.

[37] E. Blasch, I. Kadar, J. Salerno, M. M. Kokar, S. Das, G. M. Powell, D. D. Corkill, and E. H. Ruspini, "Issues and challenges of knowledge representation and reasoning methods in situation assessment (level 2 fusion)," in *Defense and Security Symposium.* International Society for Optics and Photonics, 2006, pp. 623 510–623 510.

[38] B. Bansal, *Symbolic Logic and Logic Processing.* Laxmi Publications, 2012.

[39] Y. Erdani, "Acquisition of human expert knowledge for rule-based knowledge-based systems using ternary grid," Ph.D. dissertation, Universität Duisburg-Essen, Fakultät für Ingenieurwissenschaften Elektrotechnik und Informationstechnik, 2005.

[40] J. Liebowitz, *The handbook of applied expert systems.* cRc Press, 1997.

[41] I. R. M. Association., *Machine Learning: Concepts, Methodologies, Tools and Applications (3 Volumes).* IGI Global, 2012.

[42] P. Langley, "Intelligent behavior in humans and machines," in *American Association for Artificial Intelligence*, 2006.

[43] F. Hayes-Roth, "Rule-based systems," *Communications of the ACM*, vol. 28, no. 9, pp. 921–932, 1985.

[44] J. A. Bernard, "Use of a rule-based system for process control," in *Robotics and IECON'87 Conferences.* International Society for Optics and Photonics, 1987, pp. 835–849.

[45] A. Tsakonas, G. Dounias, J. Jantzen, H. Axer, B. Bjerregaard, and D. G. von Keyserlingk, "Evolving rule-based systems in two medical domains using genetic programming," *Artificial Intelligence in Medicine*, vol. 32, no. 3, pp. 195–216, 2004.

[46] M. Flasiński, *Introduction to artificial intelligence.* Springer, 2016.

[47] R. Davis and J. J. King, "The origin of rule-based systems in ai," *Rule-Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*, 1984.

[48] G. Chryssolouris, *Manufacturing systems: theory and practice.* Springer Science & Business Media, 2013.

[49] J. K. Debenham, *Knowledge systems design.* Prentice-Hall, Inc., 1989.

[50] A. A. Hopgood, *Intelligent systems for engineers and scientists*. CRC press, 2011.

[51] V. E. Barker, D. E. O'Connor, J. Bachant, and E. Soloway, "Expert systems for configuration at digital: Xcon and beyond," *Commun. ACM*, vol. 32, no. 3, pp. 298–318, Mar. 1989. [Online]. Available: http://doi.acm.org/10.1145/62065.62067

[52] J. C. Giarratano *et al.*, "Clips user's guide," *NASA Technical Report, Lyndon B Johnson Center*, 1993.

[53] C. L. Forgy and S. J. Shepard, "Rete: A fast match algorithm," *AI Expert*, vol. 2, no. 1, pp. 34–40, Jan. 1987. [Online]. Available: http://dl.acm.org/citation.cfm?id=24761.24763

[54] M. Campbell, A. J. Hoane, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1, pp. 57–83, 2002.

[55] D. A. Ferrucci, "Introduction to this is watson," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 1–1, 2012.

[56] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[57] J. A. Sokolowski and C. M. Banks, *Modeling and simulation for analyzing global events*. John Wiley & Sons, 2009.

[58] E. A. Feigenbaum, "Expert systems: principles and practice," 1992.

[59] J. Peter, "Introduction to expert systems," 1999.

[60] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems*. Pearson Education, 2005.

[61] E. J. Friedman-Hill *et al.*, "Jess, the java expert system shell," *Distributed Computing Systems, Sandia National Laboratories, USA*, 1997.

[62] M. Proctor, M. Neale, M. Frandsen, S. Griffith, E. Tirelli, F. Meyer, and K. Verlaenen, "Jboss rules user guide," *JBoss Web site: http://labs. jboss. com/jbossrules/docs*, 2007.

[63] T. Livora, "Pmobile access to jbpm console," B.S. thesis, Masaryk University, Faculty of Informatics, Brno, Czech Republic, 2014.

[64] G. Bruce, B. Buchanan, and E. Shortliffe, "Rule-based expert systems: the mycin experiments of the stanford heuristic programming project," 1984.

[65] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, "Dendral: a case study of the first expert system for scientific hypothesis formation," *Artificial intelligence*, vol. 61, no. 2, pp. 209–261, 1993.

[66] R. T. Nakatsu, *Reasoning with Diagrams: Decision-Making and Problem-Solving with Diagrams.* John Wiley & Sons, 2009.

[67] S. Russell, P. Norvig, and A. Intelligence, "A modern approach," *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, vol. 25, 1995.

[68] Y. Shoham, "Agent-oriented programming," *Artificial intelligence*, vol. 60, no. 1, pp. 51–92, 1993.

[69] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The knowledge engineering review*, vol. 10, no. 02, pp. 115–152, 1995.

[70] P. Maes, "Intelligent software," in *Proceedings of the 2Nd International Conference on Intelligent User Interfaces*, ser. IUI '97. New York, NY, USA: ACM, 1997, pp. 41–43. [Online]. Available: http://doi.acm.org/10.1145/238218.238283

[71] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in *International Workshop on Agent Theories, Architectures, and Languages.* Springer, 1996, pp. 21–35.

[72] M. Luck, M. d'Inverno *et al.*, "A formal framework for agency and autonomy." in *ICMAS*, vol. 95, 1995, pp. 254–260.

[73] T. Salamon, *Design of agent-based models.* Eva & Tomas Bruckner Publishing, 2011.

[74] H. S. Nwana, "Software agents: An overview," *The knowledge engineering review*, vol. 11, no. 03, pp. 205–244, 1996.

[75] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach.* Prentice hall Upper Saddle River, 2003, vol. 2.

[76] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Autonomous agents and multi-agent systems*, vol. 1, no. 1, pp. 7–38, 1998.

[77] B. T. Clough, "Metrics, schmetrics! how the heck do you determine a uav's autonomy anyway," DTIC Document, Tech. Rep., 2002.

[78] G. Weiss, *Multiagent systems: a modern approach to distributed artificial intelligence.* MIT press, 1999.

[79] G. Neyens, R.-A. Dobrican, and D. Zampunieris, "Enhancing mobile devices with cooperative proactive computing," in *Proceedings of the 5th International Conference on Advanced Collaborative Networks, Systems and Applications (COLLA 2015)*. IARIA, 2015, pp. 1–9.

[80] X. Wu and X. Lin, "Object-oriented modeling of rule-based programming," 1997.

[81] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *International Symposium on Handheld and Ubiquitous Computing*. Springer, 1999, pp. 304–307.

[82] G. Chen, D. Kotz *et al.*, "A survey of context-aware mobile computing research," Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, Tech. Rep., 2000.

[83] G. Sancho, I. B. Rodriguez, T. Villemur, and S. Tazi, "What about collaboration in ubiquitous environments?" in *2010 10th Annual International Conference on New Technologies of Distributed Systems (NOTERE)*. IEEE, 2010, pp. 143–150.

[84] "Mysql. my structured query language database management system." [Retrieved: May, 2016]. [Online]. Available: http://www.mysql.com/

[85] "Hibernate orm. hibernate object/relation mapping framework." [Retrieved: May, 2016]. [Online]. Available: http://hibernate.org/

[86] J. Kotamraju, "Java api for json processing: An introduction to json." [Retrieved: August, 2016]. [Online]. Available: http://www.oracle.com/technetwork/articles/java/json-1973242.html

[87] "Mysql connector/j 5.1 developer guide." [Retrieved: August, 2016]. [Online]. Available: http://dev.mysql.com/doc/connector-j/5.1/en/

[88] G. Watson, "Ormlite package," 2010.

[89] J. Janeiro, T. Springer, and M. Endler, "A middleware service for coordinated adaptation of communication services in groups of devices," in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[90] H. M. Chung, "Toward implementing a mobile collaborative system," in *Systems and Informatics (ICSAI), 2012 International Conference on*. IEEE, 2012, pp. 1248–1252.

[91] V. Sacramento, M. Endler, H. K. Rubinsztejn, L. S. Lima, K. Goncalves, F. N. Nascimento, and G. A. Bueno, "Moca: A middleware for developing collaborative applications for mobile users," *IEEE Distributed systems online*, vol. 5, no. 10, pp. 2–2, 2004.

[92] M. Caporuscio and P. Inverardi, "Yet another framework for supporting mobile and collaborative work," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on.* IEEE, 2003, pp. 81–86.

[93] N. Cacho, K. Damasceno, A. Garcia, T. Batista, F. Lopes, and C. Lucena, "Handling exceptional conditions in mobile collaborative applications: An exploratory case study," in *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06).* IEEE, 2006, pp. 137–142.

[94] S. Bobek, G. Nalepa, and M. layski, "Challenges for migration of rule-based reasoning engine to a mobile platform," in *Multimedia Communications, Services and Security*, ser. Communications in Computer and Information Science, A. Dziech and A. Czyewski, Eds. Springer International Publishing, 2014, vol. 429, pp. 43–57, [Retrieved: May, 2015]. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-07569-3_4

[95] C. Choi, I. Park, S. J. Hyun, D. Lee, and D. H. Sim, "Mire: A minimal rule engine for context-aware mobile devices," in *Third IEEE International Conference on Digital Information Management (ICDIM), November 13-16, 2008, London, UK, Proceedings*, 2008, pp. 172–177.

[96] "Ifttt (if this then that)," https://ifttt.com/, [Retrieved: May, 2015].

[97] N. Peers. Your online life made simpler, thanks to ifttt. [Retrieved: May, 2015]. [Online]. Available: http://blog.1and1.co.uk/2014/10/02/your-online-life-made-simpler-thanks-to-ifttt/

[98] G. J. Nalepa, "Architecture of the heart hybrid rule engine," in *Proceedings of the 10th International Conference on Artifical Intelligence and Soft Computing: Part II*, ser. ICAISC'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 598–605.

[99] "Parse push," [Retrieved: May, 2016]. [Online]. Available: https://parse.com/products/push

[100] "Pubnub global data stream network," [Retrieved: May, 2016]. [Online]. Available: https://www.pubnub.com/

[101] "Urban airship," [Retrieved: May, 2016]. [Online]. Available: http://urbanairship.com/

[102] "Reliable push notifications," [Retrieved: May, 2016]. [Online]. Available: https://pushy.me/

[103] E. Nava, "Pushy a new alternative to google cloud messaging," January 2015, [Retrieved: May, 2016]. [Online]. Available: http://eladnava.com/pushy-a-new-alternative-to-google-cloud-messaging/

[104] "Battery management, android developers." [Retrieved: May, 2016]. [Online]. Available: http://developer.android.com/reference/android/os/BatteryManager.html

[105] I. Hatzilygeroudis and J. Prentzas, "Integrating (rules, neural networks) and cases for knowledge representation and reasoning in expert systems," *Expert Systems with Applications*, vol. 27, no. 1, pp. 63–75, 2004.

[106] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[107] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The multipurpose incremental learning system aq15 and its testing application to three medical domains," *Proc. AAAI 1986*, pp. 1–041, 1986.

[108] W. Li, J. Han, and J. Pei, "Cmar: Accurate and efficient classification based on multiple class-association rules," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 2001, pp. 369–376.

[109] B. L. W. H. Y. Ma, "Integrating classification and association rule mining," in *Proceedings of the fourth international conference on knowledge discovery and data mining*, 1998.

[110] C. Bădică, A. Bădiţă, M. Ganzha, A. Iordache, and M. Paprzycki, "Rule-based framework for automated negotiation: initial implementation," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web*. Springer, 2005, pp. 193–198.

[111] T. Ishida, "Parallel, distributed and multi-agent production systems-a research foundation for distributed artificial intelligence." in *ICMAS*, 1995, pp. 416–422.

[112] L. O. Hall, "Backpac: A parallel goal-driven reasoning system," *Information sciences*, vol. 62, no. 1, pp. 169–182, 1992.

[113] K. Fischer, "The rule-based multi-agent system magsy," *Proceedings of the CKBS*, vol. 92, 1993.

[114] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. John Wiley & Sons, 2007, vol. 7.

[115] S. K. Rahimi and F. S. Haug, *Distributed database management systems: A Practical Approach*. John Wiley & Sons, 2010.

[116] T. Gross, T. Paul-Stueve, and T. Palakarska, "Sensbution: A rule-based peer-to-peer approach for sensor-based infrastructures," in *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007)*, Aug 2007, pp. 333–340.

[117] R. Dobrican, S. Reis, and D. Zampunieris, "Empirical investigations on community building and collaborative work inside a lms using proactive computing," in *Proc. E-learn*, vol. 1, 2013, pp. 1840–1852.

[118] R.-A. Dobrican and D. Zampunieris, "Supporting collaborative learning inside communities of practice through proactive computing," in *Proceedings of the 5th annual International Conference on Education and New Learning Technologies, Barcelona, Spain 1-3 July, 2013*. IATED, 2013, pp. 5824–5833.

[119] R.-A. Dobrican, "Proactive dynamic community of practice," Ph.D. dissertation, University of Luxembourg, Luxembourg, Luxembourg, 2012.

[120] Facebook, inc. (2012, Dec.) Annual report on form 10-k. Facebook, inc. [Online]. Available: https://www.sec.gov/Archives/edgar/data/1326801/000132680113000003/fb-12312012x10k.htm

[121] T. Fary. Social learning vs. communities of practice. [Online]. Available: http://janetclarey.com/2011/02/22/social-learning-vs-communities-of-practice/

[122] A. Caselles, C. Francois, G. Metcalf, G. Ossimitz, and F. Stallinger, "Awareness and social systems," *Social Systems and the Future. Wien: Berichte der österreichischen Studiengesellschaft für Kybernetik*, pp. 29–42, 2000.

[123] T. Coates, "An addendum to a definition of social software," *retrieved on August*, vol. 27, p. 2009, 2005.

[124] B. L. Andersen, M. L. Jørgensen, U. Kold, and M. B. Skov, "isocialize: investigating awareness cues for a mobile social awareness application," in *Proceedings of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments*. ACM, 2006, pp. 7–14.

[125] J. Ibáñez, O. Serrano, and D. García, "Emotinet: A framework for the development of social awareness systems," in *Awareness Systems*. Springer, 2009, pp. 291–311.

[126] L. Blackall, "Digital literacy: how it affects teaching practices and networked learning futures-a proposal for action research in," *International Journal of Instructional Technology and Distance Learning*, vol. 2, no. 10, 2005.

[127] B. Ozkan and B. McKenzie, "Social networking tools for teacher education," *TECHNOLOGY AND TEACHER EDUCATION ANNUAL*, vol. 19, no. 5, p. 2772, 2008.

[128] J. Rožac, F. Buendía, J. Ballester, A. Kos, and M. Pogačnik, "Integration of learning management systems with social networking platforms. in proceedings of elml 2012," in *The Fourth International Conference on Mobile, Hybrid, and On-line Learning*, 2012, pp. 100–105.

[129] J. J. Rodrigues, F. M. Sabino, and L. Zhou, "Enhancing e-learning experience with online social networks," *IET communications*, vol. 5, no. 8, pp. 1147–1154, 2011.

[130] Z. Du, X. Fu, C. Zhao, Q. Liu, and T. Liu, "Interactive and collaborative e-learning platform with integrated social software and learning management system," in *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*. Springer, 2013, pp. 11–18.

[131] D. Shirnin, S. Reis, and D. Zampunieris, "Design of proactive scenarios and rules for enhanced e-learning," in *Proceedings of the 4th International Conference on Computer Supported Education, Porto, Portugal 16-18 April, 2012*. SciTePress–Science and Technology Publications, 2012, pp. 253–258.

[132] S. Marques Dias, S. Reis, and D. Zampunieris, "Proactive computing based implementation of personalized and adaptive technology enhanced learning over moodle (tm)," in *Proceedings of the 12th IEEE International Conference on Advanced Learning Technologies, Rome, Italy 4-6 July, 2012*. IEEE Computer Society Publications, 2012, pp. 674–675.

[133] R. A. Dobrican, G. Neyens, and D. Zampunieris, "SilentMeet-a prototype mobile application for real-time automated group-based collaboration," in *Proceedings of the 5th International Conference on Advanced Collaborative Networks, Systems and Applications (COLLA 2015)*. IARIA, 2015, pp. 52–56.

[134] ——, "A context-aware collaborative mobile application for silencing the smartphone during meetings or important events," *International Journal On Advances in Intelligent Systems*, vol. 9, no. 1&2, pp. 171–180, 2016.

[135] E. Benítez-Guerrero, C. Mezura-Godoy, and L. G. Montané-Jiménez, "Context-aware mobile collaborative systems: Conceptual modeling and case study," *Sensors*, vol. 12, no. 10, pp. 13 491–13 507, 2012.

[136] E. Williams and J. Gray, "Contextion: A framework for developing context-aware mobile applications," in *Proceedings of the 2nd International Workshop on Mobile Development Lifecycle*. ACM, 2014, pp. 27–31.

[137] S. Elmalaki, L. Wanner, and M. Srivastava, "Caredroid: Adaptation framework for android context-aware applications," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 2015, pp. 386–399.

[138] W. Wang, J. Gu, J. Yang, and P. Chen, "A group based context-aware strategy for mobile collaborative applications," in *Advanced Technology in Teaching*. Springer, 2012, pp. 541–549.

[139] L. Zavala, R. Dharurkar, P. Jagtap, T. Finin, and A. Joshi, "Mobile, collaborative, context-aware systems," in *Proc. AAAI Workshop on Activity Context Representation: Techniques and Languages, AAAI. AAAI Press*, 2011.

[140] V. Sacramento and et al., "MoCA: A Middleware for Developing Collaborative Applications for Mobile Users," *Distributed Systems Online, IEEE*, vol. 5, no. 10, pp. 2–2, Oct 2004.

[141] J. Gabler, R. Klauck, M. Pink, and H. Konig, "uBeeMe - A platform to enable mobile collaborative applications," in *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*, Oct 2013, pp. 188–196.

[142] P. Coutinho and T. Rodden, "The FUSE Platform: Supporting Ubiquitous Collaboration Within Diverse Mobile Environments," *Autom. Softw. Eng*, vol. 9, pp. 167–186, 2002.

[143] A. Gupta, A. Kalra, D. Boston, and C. Borcea, "Mobisoc: a middleware for mobile social computing applications," *Mobile Networks and Applications*, vol. 14, no. 1, pp. 35–52, 2009.

[144] S. Bendel and D. Schuster, "Watchmyphone - providing developer support for shared user interface objects in collaborative mobile applications," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, March 2012, pp. 166–171.

[145] R. Lübke, D. Schuster, and A. Schill, "A framework for the development of mobile social software on android," in *Mobile Computing, Applications, and Services*. Springer, 2011, pp. 207–225.

[146] F. Klompmaker and C. Reimann, "A service based framework for developing mobile, collaborative games," in *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ser. ACE '08. ACM, 2008, pp. 42–45.

[147] "Silence App," 2015, URL: https://play.google.com/store/apps/details?id=net.epsilonlabs.silence.ads [accessed: 2015-05-13].

[148] "Go Silent App," 2015, URL: https://play.google.com/store/apps/details?id=com.eventscheduler [accessed: 2016-05-13].

[149] "Advanced Silent Mode," 2015, URL: https://play.google.com/store/apps/details?id=com.joe.advancedsilentmode [accessed: 2016-05-13].

[150] "Silent Time," 2015, URL: https://play.google.com/store/apps/details?id=com.QuiteHypnotic.SilentTime&hl=en [accessed: 2016-05-13].

[151] "Auto Silent," 2015, URL: https://itunes.apple.com/us/app/autosilent/id474777148?mt=8 [accessed: 2015-05-13].

[152] Alastair Plumb. (2015, May) Slovakian Violist Lukas Kmit Interrupted By Nokia Ringtone, Incorporates It Into Recital. Huffington Post. [Online]. Available: http://www.huffingtonpost.co.uk/2012/01/23/slovakian-violinist-lukas-kmit-nokia-ringtone_n_1223086.html

[153] D. J. Wakin. New york philharmonic interrupted by chimes mahler never intended. [Online]. Available: http://artsbeat.blogs.nytimes.com/2012/01/11/new-york-philharmonic-interrupted-by-chimes-mahler-never-intended/?_r=0

[154] D. A. Boehm-Davis and R. Remington, "Reducing the disruptive effects of interruption: A cognitive framework for analysing the costs and benefits of intervention strategies," *Accident Analysis & Prevention*, vol. 41, no. 5, pp. 1124–1129, 2009.

[155] J. T. Shelton, E. M. Elliott, S. D. Eaves, and A. L. Exner, "The distracting effects of a ringing cell phone: An investigation of the laboratory and the classroom setting," *Journal of environmental psychology*, vol. 29, no. 4, pp. 513–521, 2009.

[156] R. A. Dobrican and D. Zampunieris, "A proactive solution, using wearable and mobile applications, for closing the gap between the rehabilitation team and cardiac patients," in *Proceedings of the IEEE International Conference on Healthcare Informatics 2016 (ICHI 2016)*. IEEE, 2016, pp. 146–155.

[157] D. Mozaffarian, E. J. Benjamin, A. S. Go, D. K. Arnett, M. J. Blaha, M. Cushman, S. de Ferranti, J.-P. Despres, H. J. Fullerton, V. J. Howard *et al.*, "Heart disease and stroke statistics-2015 update: a report from the american heart association." *Circulation*, vol. 131, no. 4, p. e29, 2015.

[158] World Health Organization, "Rehabilitation after cardiovascular diseases, with special emphasis on developing countries: report of a who expert committee," 1993.

[159] H. Gohlke and C. Gohlke-Barwolf, "Cardiac rehabilitation," *European heart journal*, vol. 19, no. 7, pp. 1004–1010, 1998.

[160] The British Association for Cardiovascular Prevention and Rehabilitation. (2012) The BACPR Standards and Core Components for Cardiovascular Disease Prevention and Rehabilitation 2012 (2nd Edition) . Online; accessed 13-December-2015]. [Online]. Available: http://www.bacpr.com/resources/15E_BACPR_Standards_FINAL.pdf

[161] B. S. Heran, J. Chen, S. Ebrahim, T. Moxham, N. Oldridge, K. Rees, D. R. Thompson, and R. S. Taylor, "Exercise-based cardiac rehabilitation for coronary heart disease," *Cochrane Database Syst Rev*, vol. 7, no. 7, 2011.

[162] D. Thompson and D. De Bono, "How valuable is cardiac rehabilitation and who should get it?" *Heart*, vol. 82, no. 5, pp. 545–546, 1999.

[163] H. Antonakoudis, K. Kifnidis, A. Andreadis, E. Fluda, Z. Konti, N. Papagianis, H. Stamou, E. Anastasopoulou, G. Antonakoudis, and L. Poulimenos, "Cardiac rehabilitation effects on quality of life in patients after acute myocardial infarction," *Hippokratia*, vol. 10, no. 4, p. 176, 2006.

[164] R. Arena, M. Williams, D. E. Forman, L. P. Cahalin, L. Coke, J. Myers, L. Hamm, P. Kris-Etherton, R. Humphrey, V. Bittner *et al.*, "Increasing referral and participation rates to outpatient cardiac rehabilitation: The valuable role of healthcare professionals in the inpatient

and home health settings a science advisory from the american heart association," *Circulation*, vol. 125, no. 10, pp. 1321–1329, 2012.

[165] P. D. Thompson, B. A. Franklin, G. J. Balady, S. N. Blair, D. Corrado, N. M. Estes, J. E. Fulton, N. F. Gordon, W. L. Haskell, M. S. Link *et al.*, "Exercise and acute cardiovascular events placing the risks into perspective: a scientific statement from the american heart association council on nutrition, physical activity, and metabolism and the council on clinical cardiology," *Circulation*, vol. 115, no. 17, pp. 2358–2368, 2007.

[166] D. R. Thompson and R. J. Lewin, "Management of the post-myocardial infarction patient: rehabilitation and cardiac neurosis," *Heart*, vol. 84, no. 1, pp. 101–105, 2000.

[167] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, "A review of wearable sensors and systems with application in rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 9, no. 1, pp. 1–17, 2012. [Online]. Available: http://dx.doi.org/10.1186/1743-0003-9-21

[168] J. R. Windmiller and J. Wang, "Wearable electrochemical sensors and biosensors: A review," *Electroanalysis*, vol. 25, no. 1, pp. 29–46, 2013. [Online]. Available: http://dx.doi.org/10.1002/elan.201200349

[169] U. Anliker, J. A. Ward, P. Lukowicz, G. Tröster, F. Dolveck, M. Baer, F. Keita, E. B. Schenker, F. Catarsi, L. Coluccini *et al.*, "Amon: a wearable multiparameter medical monitoring and alert system," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 8, no. 4, pp. 415–427, 2004.

[170] M. Bächlin, M. Plotnik, D. Roggen, I. Maidan, J. M. Hausdorff, N. Giladi, and G. Tröster, "Wearable assistant for parkinsons disease patients with the freezing of gait symptom," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, no. 2, pp. 436–446, 2010.

[171] J. Hernandez, D. McDuff, and R. W. Picard, "Biowatch: estimation of heart and breathing rates from wrist motions," in *Proceedings of the 9th International Conference on Pervasive Computing Technologies for Healthcare*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2015, pp. 169–176.

[172] J. Allen, "Photoplethysmography and its application in clinical physiological measurement," *Physiological measurement*, vol. 28, no. 3, p. R1, 2007.

[173] J. J. McMurray, S. Adamopoulos, S. D. Anker, A. Auricchio, M. Böhm, K. Dickstein, V. Falk, G. Filippatos, C. Fonseca, M. A.

Gomez-Sanchez *et al.*, "Esc guidelines for the diagnosis and treatment of acute and chronic heart failure 2012," *European journal of heart failure*, vol. 14, no. 8, pp. 803–869, 2012.

[174] A. C. of Sports Medicine *et al.*, *ACSM's guidelines for exercise testing and prescription*. Lippincott Williams & Wilkins, 2013.

[175] M. F. Piepoli, V. Conraads, U. Corra, K. Dickstein, D. P. Francis, T. Jaarsma, J. McMurray, B. Pieske, E. Piotrowicz, J.-P. Schmid *et al.*, "Exercise training in heart failure: from theory to practice. a consensus document of the heart failure association and the european association for cardiovascular prevention and rehabilitation," *European journal of heart failure*, vol. 13, no. 4, pp. 347–357, 2011.

[176] N. F. Gordon, M. Gulanick, F. Costa, G. Fletcher, B. A. Franklin, E. J. Roth, and T. Shephard, "Physical activity and exercise recommendations for stroke survivors an american heart association scientific statement from the council on clinical cardiology, subcommittee on exercise, cardiac rehabilitation, and prevention; the council on cardiovascular nursing; the council on nutrition, physical activity, and metabolism; and the stroke council," *Stroke*, vol. 35, no. 5, pp. 1230–1240, 2004.

[177] P. Kokkinos and J. Myers, "Exercise and physical activity clinical outcomes and applications," *Circulation*, vol. 122, no. 16, pp. 1637–1648, 2010.

[178] T. Moholdt, E. Madssen, Ø. Rognmo, and I. L. Aamot, "The higher the better? interval training intensity in coronary heart disease," *Journal of Science and Medicine in Sport*, vol. 17, no. 5, pp. 506–510, 2014.

[179] V. M. Conraads, N. Pattyn, C. De Maeyer, P. J. Beckers, E. Coeckelberghs, V. A. Cornelissen, J. Denollet, G. Frederix, K. Goetschalckx, V. Y. Hoymans *et al.*, "Aerobic interval training and continuous training equally improve aerobic exercise capacity in patients with coronary artery disease: the saintex-cad study," *International journal of cardiology*, vol. 179, pp. 203–210, 2015.

[180] D. S. Siscovick, N. S. Weiss, R. H. Fletcher, and T. Lasky, "The incidence of primary cardiac arrest during vigorous exercise," *New England Journal of Medicine*, vol. 311, no. 14, pp. 874–877, 1984.

[181] S. Giri, P. D. Thompson, F. J. Kiernan, J. Clive, D. B. Fram, J. F. Mitchel, J. A. Hirst, R. G. McKay, and D. D. Waters, "Clinical and angiographic characteristics of exertion-related acute myocardial infarction," *Jama*, vol. 282, no. 18, pp. 1731–1736, 1999.

[182] W. Nieuwland, M. A. Berkhuysen, D. J. van Veldhuisen, J. Brügemann, M. L. Landsman, E. van Sonderen, K. Lie, H. J. Crijns, and P. Rispens, "Differential effects of high-frequency versus low-frequency exercise training in rehabilitation of patients with coronary artery disease," *Journal of the American College of Cardiology*, vol. 36, no. 1, pp. 202–207, 2000.

[183] A. C. of Sports Medicine *et al.*, "ACSM's guidelines for exercise testing and prescription," in *Adaptations to cardiorespiratory exercise training*, D. P. Swain and C. A. Brawner, Eds. Oxford: Oxford University Press, 2013, ch. 32, pp. 496–510.

[184] R. W. Kusuma, R. A. A. Abbie, and P. Musa, "Design of arrhythmia detection device based on fingertip pulse sensor," in *Proceedings of Second International Conference on Electrical Systems, Technology and Information 2015 (ICESTI 2015)*. Springer, 2016, pp. 363–372.

[185] S. Magalhães, M. M. Ribeiro, A. Barreira, P. Fernandes, S. Torres, J. L. Gomes, and S. Viamonte, "Long-term effects of a cardiac rehabilitation program in the control of cardiovascular risk factors," *Revista Portuguesa de Cardiologia (English Edition)*, vol. 32, no. 3, pp. 191–199, 2013.

[186] A. Sarela, J. Salminen, E. Koskinen, O. Kirkeby, I. Korhonen, and D. Walters, "A home-based care model for outpatient cardiac rehabilitation based on mobile technologies," in *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, April 2009, pp. 1–8.

[187] E. Kyriacou, P. Chimonidou, C. Pattichis, E. Lambrinou, V. Barberis, and G. Georghiou, "Post cardiac surgery home-monitoring system," in *Wireless Mobile Communication and Healthcare*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, J. Lin and K. Nikita, Eds. Springer Berlin Heidelberg, 2011, vol. 55, pp. 61–68. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20865-2_9

[188] J. Maitland and M. Chalmers, "Self-monitoring, self-awareness, and self-determination in cardiac rehabilitation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 1213–1222. [Online]. Available: http://doi.acm.org/10.1145/1753326.1753508

[189] V. Gay, P. Leijdekkers, and E. Barin, "A mobile rehabilitation application for the remote monitoring of cardiac patients after a heart attack or a coronary bypass surgery," in *Proceedings*

of the 2Nd International Conference on PErvasive Technologies Related to Assistive Environments, ser. PETRA '09. New York, NY, USA: ACM, 2009, pp. 21:1–21:7. [Online]. Available: http://doi.acm.org/10.1145/1579114.1579135

[190] D. Phan, L. Y. Siong, P. Pathirana, and A. Seneviratne, "Smartwatch: Performance evaluation for long-term heart rate monitoring," in *Bioelectronics and Bioinformatics (ISBB), 2015 International Symposium on*, Oct 2015, pp. 144–147.

[191] J. M. Kang, T. Yoo, and H. C. Kim, "A wrist-worn integrated health monitoring instrument with a tele-reporting device for telemedicine and telecare," *Instrumentation and Measurement, IEEE Transactions on*, vol. 55, no. 5, pp. 1655–1661, Oct 2006.

[192] Y. Shahar, D. Goren-Bar, D. Boaz, and G. Tahan, "Distributed, intelligent, interactive visualization and exploration of time-oriented clinical data and their abstractions," *Artificial intelligence in medicine*, vol. 38, no. 2, pp. 115–135, 2006.

[193] C. Combi, G. Pozzi, and R. Rossato, "Querying temporal clinical databases on granular trends," *Journal of biomedical informatics*, vol. 45, no. 2, pp. 273–291, 2012.

[194] K. Wasserman, "The anaerobic threshold: definition, physiological significance and identification," 1986.

[195] Polar Electro, "How to do OwnZone determination," March 2016. [Online]. Available: http://support.polar.com/en/support/tips/How_to_do_OwnZone_determination

[196] L. A. Sebastian, S. Reeder, and M. Williams, "Determining target heart rate for exercising in a cardiac rehabilitation program: A retrospective study," *Journal of Cardiovascular Nursing*, vol. 30, no. 2, pp. 164–171, 2015.

[197] S. M. Fox, J. P. Naughton, and W. Haskell, "Physical activity and the prevention of coronary heart disease," in *Ann Clin Res*, 1971, pp. 404–432.