UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2018-55
The Faculty of Sciences, Technology and Communication

DISSERTATION

Presented on 27/08/20018 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Alejandro SANCHEZ GUINEA
Born on 8 June 1984 in Mexico City (Mexico)

ENGINEERING SMART SOFTWARE SERVICES FOR INTELLIGENT PERVASIVE SYSTEMS

Dissertation defense committee

Dr. Jacques Klein, chairman
Senior Research Scientist, University of Luxembourg, Luxembourg (Luxembourg)

Dr. Gregory Nain, vice chairman
R&D Software Engineer, DataThings, Luxembourg (Luxembourg)

Prof. Dr. Yves Le Traon, dissertation supervisor
Professor, University of Luxembourg, Luxembourg (Luxembourg)

Prof. Dr. Kary Främling, external member
Professor, Aalto University, Espoo (Finland)

A-Prof. Dr. Sylvain Kubler, external member
Associate Professor, Université de Lorraine, Nancy (France)

Acknowledgements

The present work has been supported by POST Luxembourg as part of a partnership project between the Interdisciplinary Center for Security, Reliability and Trust (SnT) of University of Luxembourg and POST Luxembourg.

I would like first to thank my dissertation supervisor Prof. Dr. Yves Le Traon. It has been a truly enriching experience to conduct my doctoral studies under his supervision, receiving encouragement, support, and thoughtful guidance during these years.

I would also like to thank the members of the SerVal research group for their support and great teamwork. In particular I would like to thank all the team that contributed directly or indirectly in the project related to my doctoral work.

I would like to express my appreciation to Mr. François Erasmy and Mr. Sergio Sousa from POST Luxembourg, for promoting and effective and successful collaboration project.

Finally, I would like to thank the members of my dissertation defence committee Dr. Jacques Klein, Dr. Gregory Nain, Prof. Dr. Kary Främling, A-Prof. Dr. Sylavin Kubler, and my dissertation supervisor Prof. Dr. Yves Le Traon, for their time invested and feedback provided in reviewing my work.

Luxembourg, August 2018 Alejandro S. Guinea

Abstract

Pervasive computing systems, envisioned as systems that blend with the physical environment to enhance the quality of life of its users, are rapidly becoming a not so distant reality. However, many challenges must be addressed before realizing the goal of having such computing systems as part of our everyday life. One such challenge is related to the problem of how to develop in a systematic way the software that lies behind pervasive systems, operating them and allowing them to intelligently adapt both to users' changing needs and to variations in the environment. In spite of the important strides done in recent years concerning the engineering of software that places the actual, immediate needs and preferences of users in the center of attention, to the best of our knowledge no work has been devoted to the study of the engineering process for building software for pervasive systems.

In this dissertation we focus on the engineering process to build smart software services for pervasive systems. Specifically, we first introduce as our first major contribution a model for the systematic construction of software for pervasive systems, which has been derived using analytical, evidence-based, and empirical methodologies. Then, on the basis of the proposed model, we investigate two essential mechanisms that provide support for the engineering of value-added software services for smart environments, namely the learning of users' daily routines and the continuous identification of users. For the case of learning users' daily routines, we propose what is our second main contribution: a novel approach that discovers periodic-frequent routines in event data from sensors and smart devices deployed at home. For the continuous identification of users we propose what is our third major contribution: a novel approach based on behavioral biometrics which is able to recognize identities without requiring any specific gesture, action, or activity from the users. The two approaches proposed have been extensively evaluated through studies in the lab, based on synthetic data, and in the wild, showing that they can be effectively applied to different scenarios and environments.

In sum, the engineering model proposed in this dissertation is expected to serve as a basis to further the research and development efforts in key aspects that are necessary to build value-added smart software services that bring pervasive systems closer to the way they have been envisioned. Furthermore, the approaches proposed for learning users' daily routines and recognizing users' identities in smart environments are aimed at contributing to the investigation and development of the data analytics technology necessary for the smart adaptation and evolution of the software in pervasive systems to users' needs.

Contents

List of Figures

List of Tables

Part I

Opening

Chapter 1

Introduction

1.1 Motivation

Ubiquitous or pervasive computing have been envisioned as a paradigm that takes into account the natural human environment and pushes the computing devices and services to vanish into the background in order to support all aspects of our everyday life [Weiser, 1991].

While under some definitions pervasive computing refers only to the idea of having computing devices and sensors as omnipresent objects in the environment [Al-Muhtadi et al., 2003], the support to everyday life that is expected from such systems requires that pervasive computing applications adapt to users' changing needs over time [Conti et al., 2012]. This makes it essential for pervasive systems to feature smart functionality with input-output channels that continuously obtain information about the users, the environment, and other context factors. The smart functionality of pervasive systems is mainly implemented through software, focusing on self-awareness, self-adaptation, and self-optimization capabilities in a context-aware manner, where the information about the situation of users, the environment, and the state of the system itself are considered to adapt the behavior of the system accordingly [Dey, 2001].

Therefore, one of the major challenges to bring about the vision of pervasive systems is related to the problem of how to develop in a systematic way the software that lies behind pervasive systems, operating them and allowing them to intelligently adapt both to users' changing needs and to variations in the environment.

Given the importance that software has in today's digitalization era as the main driver of the services delivered to users, the engineering of software is moving towards a model in which the focus is no longer on technical issues but on identifying new ways to deliver value to users by building what really matters for them. That is, more and more engineering methods are seeking to build value-added software products and services (i.e., software that delivers value to users).

While some works have dealt with the problem of software engineering for pervasive systems (e.g., [Bures et al., 2017]), we could not find any work that provides an engineering model that can serve as a framework for building software for pervasive systems in a way that value is delivered to users through smart functionality.

The overall *goal* of this dissertation is to investigate how to engineer in a systematic way software for pervasive computing systems, which can provide value-added services to users through software-based smart functionality.

1.2 Challenges

The research work herein presented has faced a number of challenges that we have had to overcome in order to achieve the goal of the dissertation.

Challenge 1. The lack of previous works that examine the engineering of software for pervasive systems in a holistic way poses the first challenge. This challenge implies that while lots of research and development efforts have been made around software for pervasive systems, no specific groundwork has been organized about a comprehensive model of engineering. To tackle this challenge we try to establish an idea of the phases of the development cycle based on the main approaches found in the literature, which we combine with an engineering model derived for a broader and more studied type of systems, namely software-intensive systems.

Challenge 2. The large amount of proposed approaches for different aspects of the engineering of software for pervasive systems, which lack proper empirical evaluation to ground their claimed contributions, represents another challenge for our work. This is because it makes it difficult to recognize the works that are relevant and sound, which can be considered for further developments. To overcome this challenge we have used a systematic review approach to methodically search, select, and organize relevant and sound works from the literature, which are to be considered for our research.

Challenges 3. A major challenge of our work, especially for the research behind Chapter 5 and Chapter 6, is related to the empirical evaluation of our approaches for smart pervasive systems and its associated difficulties. The significance of empirical evaluation has been stressed as a way to produce theories that rely on verifiable facts [Greenberg and Buxton, 2008]. However, it has also been highlighted the many difficulties related to pervasive computing evaluation [Hazlewood et al., 2011, Favela et al., 2010]. To cope with such challenge, we have prepared our evaluation methodology following previous works that provide lessons learned and guidelines, e.g., [Brown et al., 2011, Brush et al., 2011, Gustarini et al., 2013]. Thus, in general our evaluations have been conducted considering carefully designed, planned, and implemented experiments, ethics policies compliance[1], privacy concerns of participants, proper organization, and public data availability for replication purposes (see Section 5.4.4 (Chapter 5) and Section 6.4.4 (Chapter 6)).

Challenge 4. The absence of approaches that are comparable to our methods presented in Chapter 5 (i.e., approach to discover periodic-frequent routines at home) and Chapter 6 (i.e., approach for the continuous identification of users in smart spaces) poses a challenge for our work. This is because in this case it is not possible to assess our approaches by comparing directly their performance to the ones of other approaches previously proposed. Furthermore, we could not opt to use existing datasets, since all we found was

[1] Our evaluation studies are compliant with the University of Luxembourg Policy on Ethics

either too restrictive or was lacking the necessary ground truth to make possible a proper assessment of our approaches. To deal with this issue we conducted comprehensive evaluation efforts for the approaches presented in Chapter 5 and Chapter 6, which took into account different configurations, environments, and potential scenarios that the approaches may face in real-world situations.

1.3 Research Methodology

To better understand and address the goal of the dissertation we divide it into the following two main objectives:

- Investigate the engineering of software for pervasive systems and devise a model for the systematic development of smart software services for this type of systems.
- Identify the key parts of the engineering process that allow the software services for pervasive systems to provide smart functionality. From these, investigate the mechanisms that are particularly important in providing higher value to users.

1.3.1 Research Questions

In order to achieve our objectives we propose to answer the following research questions:

RQ1: How to develop smart software services for pervasive systems in a systematic way?

RQ2: How to provide support to the development of value-added services for smart environments?

Question *RQ1* is related to the objective that focuses on the engineering process as a whole. The main concern in this case is on identifying the general mechanism that can lead to producing software with smart capabilities for pervasive systems in a systematic manner. In order to solve one by one each of the main concerns of *RQ1* we divide it into the following questions:

RQ1-a: How to engineer value-added software for software-intensive systems in a systematic way?

RQ1-b: What has been considered in the literature as the implicit engineering cycle to build software for pervasive systems?

RQ1-c: What are the key mechanisms to support the development of software for pervasive systems featuring smart functionality?

In *RQ2* we narrow down the focus to smart environments to stress the need for providing smart functionality. Smart functionality here is understood as such that enables the computing system to be adaptive, configurable, dynamic, and reactive [Esposito et al., 2018]. The answer is expected to dig deep into some specific relevant mechanisms of the engineering process, rather than looking at the whole engineering cycle. To address one by one each of the main concerns of *RQ2* we break it down into the following questions:

RQ2-a: What are the key mechanisms in the engineering of software for pervasive systems to provide value-added services to users through smart functionality?

RQ2-b: How to automatically learn users' routines at home, even when they are only frequent on specific periodicities?

RQ2-c: How to automatically learn to recognize the identities of users in a continuous manner in smart environments?

RQ2-d: What are examples of application services that can be built from the mechanisms of question *RQ2-a*?

1.3.2 Research Process

To answer *RQ1* we combine three methodologies: analytical, empirical, and evidence-based. Our methodology here aims to compensate for the lack of existing engineering models with a similar target to ours.

Our first step is to focus our attention on the development of software for a much more studied type of systems, which encompass pervasive systems under their umbrella. These are software-intensive systems, defined as such in which the software interacts with other software and systems, as well as with devices, sensors, and people [Wirsing et al., 2008]. At this respect, we target to answer *RQ1-a* (in Chapter 2) to gain insights into aspects related to the systematic development of software and the production of software that delivers value to users.

The next step is to investigate through evidence-based methodology the view that the literature reflects implicitly about the engineering process to build software for pervasive systems. That is, we target to answer *RQ1-b* (in Chapter 3). Thus, based on a systematic review we search for the most representative state-of-the-art approaches for any of the phases of the development cycle, to know what are the phases that have received more attention and what are the most relevant existing methods and their limitations. Furthermore, we gather the main research challenges that have been identified by relevant previous works, to better understand the engineering process as a whole, spotting its key components.

Our third step is focused on the key mechanisms of the engineering process for producing software for pervasive systems that features smart functionality. That is, we target to answer *RQ1-c* (in Chapter 4). Thus, we analytically derive the answer using the results of the literature review as a basis.

The final step to answer *RQ1* is to define the model for engineering software for pervasive systems by framing the answers to *RQ1-b* and *RQ1-c* using the proposed model for software-intensive system (i.e., the solution to *RQ1-a*). This is accomplished analytically, specifying the aspects that characterize the particular case of the systematic development of value-added software for pervasive systems.

The answer to *RQ2* is expected to be reached by providing solution to each of the questions in which it was broken down.

First, we narrow down the focus on the mechanisms for smart functionality identified

in *RQ1-c* to only consider those mechanisms that can be seen as key for delivering value to users. That is, we target to answer *RQ2-a* (in Chapter 4) . The answer to this question lead the way into the rest of the answer for *RQ2*, by focusing on the mechanisms that learn about the dynamic characteristics of users.

We focus only on two key mechanisms that serve to support the development of software services for smart environments. These are: a mechanism to learn about the periodic-frequent routines of users at home, and a mechanism to continuously recognize the identities of users. For each of these cases we propose a novel approach related to the corresponding mechanism, providing answer to the questions *RQ2-b* (in Chapter 5) and *RQ2-c* (in Chapter 6), respectively. We conduct extensive empirical evaluation to assess the performance of these approaches in terms of accuracy, robustness, and practicality towards being used in real-world scenarios. As an additional step to better examine the dimension of the mechanisms proposed, we investigate what are some application services that can be built on top of them. That is, we answer *RQ2-d* (in Chapter 7) based on the literature and on a small-scale example implementation.

1.4 Contributions

The main contributions of this dissertation to the state of the art are:

Contribution 1. A systematic literature review on different relevant aspects of the development cycle to build software for pervasive (aka ubiquitous) systems. Specifically, the main goal of the review is to provide information about what are the phases of the development cycle that have been considered by the main state-of-the-art approaches. Thus, for each phase considered, our review describes the main approaches and their limitations. Furthermore, from the studies selected by the systematic methodology, our review provides information on the main research challenges that have been identified in the literature. Concerning the research methodology of the dissertation, this contribution is directly related to question *RQ1-b* and in a broader sense to research question *RQ1*. We present this contribution in full length in Chapter 3.

Contribution 2. A model for the systematic development of value-added smart software services for pervasive systems. The model considers a data analytics component that, based on machine-learning, semi-supervised and unsupervised techniques, is expected to provide support for the software development cycle through feedback. The feedback is a constant during the whole cycle between any of the standard phases (e.g., implementation, design, testing) and the learnings being obtained about the users, the environment, the system itself, and other context factors. This contribution corresponds to the answer to the research question *RQ1* of the dissertation and it is presented in Chapter 4.

Contribution 3. A novel knowledge-discovery approach to automatically find the periodic-frequent routines of users at home from smart devices and sensors event data. Specifically, the approach is based on an extension we made of the standard mining algorithm called Generalized Sequential Pattern (GSP). This approach is the first of its kind that is able to discovery routines that are only frequent for specific periodicities. Furthermore, it can find routines that were not always performed in exactly the same way

by the home inhabitants. We conducted an extensive evaluation in the lab, in the wild, and based on synthetic data. Our results show that the approach has a high recall-precision performance ($\sim$ 0.9) and it is robust for different configurations and under different environments. This contribution corresponds to the answer to question *RQ2-b* and in a broader sense to research question *RQ2*. We present this contribution in full in Chapter 5.

Contribution 4. A novel approach for the continuous identification of users in indoor environments based on behavioral biometrics. Specifically, the approach uses arm and hand motion patterns to form the profiles that allow to recognize the identity of each user. The data is collected in the form of time series of quaternions from an Inertial Measurement Unit (IMU) that contains an accelerometer, a magnetometer, and a gyroscope. We then apply wavelet transform on the time series to obtain the observations that are passed to a machine-learning classification stage, where the identities are assigned. The novelty of the approach stems from its ability to perform the identification without requiring any specific gesture, action, or activity being performed by the user. Thus, it allows users to perform their daily routines without any kind of restrictions while it recognizes their identities. We conducted an empirical evaluation of the approach considering lab and real-world environments, namely home and office. Our results show that our approach is able to identify users with an accuracy of 0.88 for office environments and of 0.71 for the average size of a household.

In addition to the main contributions, we are making publicly available the data collected during the empirical evaluations of approaches of Contribution 3 and Contribution 4. We hope that this data contributes to advance research and development efforts in the domains of pattern discovery for smart environments and behavioral biometrics identification. Furthermore, the engineering model for software-intensive systems that we present in Chapter 2 has been developed also as part of the doctoral work of this dissertation.

The research work of this dissertation has produced the following articles:

- A. S. Guinea, A. Boytsov, L. Mouline, and Y. Le Traon. "Continuous Identification in Smart Environments Using Wrist-Worn Inertial Sensors," in *15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2018)*, November 2018.
- A. S. Guinea, A. Boytsov, L. Mouline, and Y. Le Traon. "Smart Discovery of Periodic-Frequent Human Routines for Home Automation," submitted to *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2019.
- A. S. Guinea, G. Nain, and Y. Le Traon. "A systematic review on the engineering of software for ubiquitous systems," *Journal of Systems and Software*, 118, pp. 251–276, August 2016. DOI: 10.1016/j.jss.2016.05.024
- F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch. "The RIGHT Model for Continuous Experimentation," *Journal of Systems and Software*, 123, pp.292–305, January 2017. DOI: 10.1016/j.jss.2016.03.034

1.5 Organization of the dissertation

The rest of the dissertation is organized as follows. The main body of the dissertation is divided into two parts: *i*) Part II devoted to our proposed model for the engineering of value-added smart software for pervasive systems (i.e., answer to *RQ1*), and *ii*) Part III which presents data analytics mechanisms to support the development of value-added software services for smart environments (i.e., answer to *RQ2*).

Part II is divided into three chapters: chapter 2, 3, and 4. In Chapter 2 we present an engineering model for the systematic development of value-added software for software-intensive systems (i.e., answer to *RQ1-a*). In Chapter 3 we present a systematic literature review on the engineering of software for pervasive systems (i.e., answer to *RQ1-b*). Specifically, the review corresponds to an investigation to find evidence about the development cycle based on state-of-the-art approaches that have focused in one or several of its phases. In Chapter 4, we first investigate the key mechanisms necessary to support the creation of software for pervasive systems, which features smart functionality (i.e., answer to *RQ1-c*) and delivers value to users (i.e., answer to *RQ2-a*). Then, in the same chapter we introduce our proposed engineering model for the systematic development of value-added smart software for pervasive systems (i.e. answer to *RQ1*).

Part III, also divided into three chapters (5,6, and 7), is developed with respect to our proposed model in Chapter 4 (Part II), particularly around what we found to be the main data analytics mechanisms to support the development of smart software services that deliver value to users. In Chapter 5, we present a novel knowledge-discovery-in-data (KDD) approach to automatically find periodic-frequent patterns of people from event data collected by smart devices and sensors deployed at home. In Chapter 6, we present a novel approach for the continuous identification of users in indoor environments, based on behavioral biometrics and machine-learning (ML) classification techniques. In both Chapter 5 and Chapter 6, we describe the extensive empirical evaluation that we conducted in each case to assess the performance of our proposed approaches under different configurations and settings. In Chapter 7, we present some application services where the support of the mechanisms proposed in chapters 5 and 6 would be essential, including an implementation of a home automation service that we build on top of the approach presented in Chapter 5, and of which we describe our approach and the results of an in-the-wild evaluation.

The main body of the dissertation is followed by a closing part, where we present an overall summary of the dissertation, potential future research directions, a synopsis of our answers to the research questions, and a discussion about the validity of the research of the dissertation.

Part II

Engineering Model

In this part we study the process of engineering software for pervasive systems. We first focus in a broader engineering process such as the one related to software-intensive systems in general, to establish a theoretical framework that can serve later as basis for the study of pervasive systems in particular. Then, we focus on the engineering process related to software for pervasive systems, including the main phases that have been considered, state-of-the-art approaches that deal with different parts of the process, as well as limitations, and research challenges that require further development. Finally, we present our proposed model for engineering value-added smart software services for pervasive systems.

The sections in this part are based on the work that has been presented in the following papers:

- *(Section 2) F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch. "The RIGHT Model for Continuous Experimentation,"* Journal of Systems and Software*, 123, pp.292–305, January 2017.*
  DOI: 10.1016/j.jss.2016.03.034
- *(Section 3) A. S. Guinea, G. Nain, and Y. Le Traon. "A systematic review on the engineering of software for ubiquitous systems,"* Journal of Systems and Software*, 118, pp. 251–276, August 2016.*
  DOI: 10.1016/j.jss.2016.05.024

Chapter 2

General Model for Engineering Software-Intensive systems

2.1 Introduction

This chapter is aimed at providing an answer to the first of the questions in which we have broken down our research question *RQ1*, that is, *RQ1-a*: How to engineer software for software-intensive systems in a systematic way?

Software is becoming increasingly important for a wider range of industry sectors, thanks to the accelerating pace and reach of the digitalization era that today's world is experiencing. As software becomes the main driver of the services delivered to customers, the focus in building software is moving from identifying and solving technical problems to deliver value to customers by building what really matters for them. At this respect, a family of generic approaches has been proposed for the systematic development of value-added software (i.e., software that delivers value to users). One of such approaches is the Learn Startup methodology [Ries, 2011], which proposes a general three-step cycle (viz., build, measure, and learn) that is described as a high-level template that may guide the development of software based on experimentation (i.e., experiment-based software development). In spite of the growing acceptance that this type of methodologies have enjoyed in recent years, prior to our work no framework had been proposed for conducting systematic experiment-based software development.

In this chapter we present the most important building blocks of a framework for the systematic development of value-added software for software-intensive systems, where value is delivered to users based on the learnings obtained from a continuous cycle created around the analysis of real-time use of the software, denominated as continuous experimentation. This framework has implications for the technical product infrastructure, the software development process, the requirements regarding skills that software developers need to design, execute, analyze, and interpret experiments, and the organizational capabilities needed to operate and manage a company based on experimentation in research and development. Methods and approaches for continuous experimentation with software product and service value should itself be based on empirical research.

The rest of the chapter is organized as follows. First in Section 2.2, we present the main previous works that have dealt with experimentation in the process of engineering software, works that have proposed models to deliver value to users in a systematic way, and works on approaches about software development around the concept of continuous experimentation. Then, in Section 2.3 we describe the research methodology we follow in order to derive the proposed engineering model for software-intensive systems. In Section 2.4 we present our results, namely, the engineering model divided into two main components: a process model and an infrastructure model. We then present our analysis on some specific instantiations we made of the engineering model on software development projects. In Section 2.5, we present two lines of discussion: the validity of our research methodology and the limitations we identify on the proposed engineering model. Finally, Section 2.6 presents a summary of the chapter.

2.2 Related Works

In this section, we describe what according to the literature are the main models for systematic value delivery and approaches for using experiments in the engineering of software.

2.2.1 Experimentation in the engineering of software

Experimentation has been established in software engineering since the 1980s. The seminal work in [Basili et al., 1986] was among the first to codify a framework and process for experimentation. More recently, in [Juristo and Moreno, 2013] and in [Wohlin et al., 2012] the authors presented updated overviews on experimentation in software engineering.

Experimentation in software engineering may encompass both controlled experiments and explorative endeavors which are aimed at discovery and knowledge creation [Wohlin et al., 2012]. The logic of controlled experiments and case studies is essentially different, where the first one rely on statistical analysis whereas the second typically include qualitative elements that are generalized analytically [Yin, 2009]. Qualitative methods may also be used alone, such as studies based on interviews or observations.

In [Kohavi et al., 2012, Kohavi et al., 2013] it is noted that running experiments requires addressing multiple challenges in three areas: cultural/organizational, engineering, and trustworthiness. The organization needs to learn the reasons for running controlled experiments and the trade-offs between controlled experiments and other methods of evaluating ideas. Even negative experiments which degrade user experience in the short term should be conducted, due to their learning value and long-term benefits. When the technical infrastructure supports hundreds of concurrent experiments, each with millions of users, classical testing and debugging techniques no longer apply because there are millions of live variants of the system in production. Instead of heavy up-front testing, the authors in [Kohavi et al., 2012] report having used alerts and post-deployment fixing. Furthermore, experiments help to avoid many negative features even when key stakeholders initially support them, saving large amounts of money as a result.

Experimentation also has an important relationship with company culture. In [Kohavi et al., 2009] the authors describe a platform for experimentation built and used at Microsoft, noting the cultural challenges involved in using experiment results, rather than opinions from persons in senior positions, as the basis of decisions. They suggest, for example, that one should avoid trying to build features through extensive planning without early testing of ideas, that experiments should be carried out often, that a failed experiment is a learning opportunity rather than a mistake, and that radical and controversial ideas should be put forward and tried. All these suggestions are challenging to put into practice in organizations that are not used to experimentation-based decision-making.

2.2.2 Models for systematic value delivery

Lean manufacturing and the Toyota Production System [Ohno, 1988] has inspired the definition of Lean software development. This approach provides comprehensive guidance for the combination of design, development, and validation built as a single feedback loop focused on discovery and delivery of value [Poppendieck and Cusumano, 2012]. The main ideas of this approach are encompassed by seven principles: optimize the whole, eliminate waste, build quality in, learn constantly, deliver fast, engage everyone, and keep getting better [Poppendieck and Poppendieck, 2003].

The Lean Startup methodology [Ries, 2011] provides mechanisms to ensure that product or service development effectively addresses what customers want. The methodology is based on a Build-Measure-Learn loop which establishes learning about customers and their needs as the unit of progress. It proposes to apply scientific method and thinking to startup businesses in the form of learning experiments. As the results of experiments are analyzed, the company has to decide to 'persevere' on the same path or 'pivot' in a different direction while considering what has been learned from customers. The Build-Measure-Learn cycle holds clear similarities with the Plan-Do-Check-Adjust (PDCA) method popularized by W. Edwards Deming, where in fact lean methods can be seen as involving a progression of PDCA cycles [Liker and Franz, 2011]. PDCA is a known method which is used for maintenance, improvement, and innovation of products, services and processes. It converges in two ways with the lean methodology [Silva et al., 2013]: *i*) performing successive changes in operational or administrative processes, with successive gains obtained without investment, through incremental and continuous improvement of an activity to create more value with less resource-consuming activities (known in the business as Kaizen or continuous improvement); and *ii*) designing a new process to achieve the desired goal or making substantial changes to existing processes (known as Kaikaku), which lead to great advances, radical improvement, and new investments..

The Customer Development methodology [Blank, 2013] emphasizes the importance of not only doing product development activities but also learning and discovering the potential customers of the company. Customer Development is divided into a search and an execution phase. In the search phase the company performs customer discovery, testing whether the business model is correct (product/market fit), and customer validation, which develops a replicable sales model. The execution phase is devoted to the creation of demand and to take the company into a model for cost-efficient delivery of validated

products/services.

In a work along the lines of the Lean Startup methodology, the authors in [Olsson et al., 2012] describe a sequence of four stages (called 'stairway to heaven') which a software organization should traverse towards the objective establishing a development system with the ability to continuously learn from the real-time customer usage of software. The last stage (i.e., the objective) is achieved when the software organization functions as an R&D experiment system. The stages on the way to achieving the target are: traditional development, agile R&D organization, continuous integration, and continuous deployment. One of their main findings is that the transition towards Agile development requires shifting to small development teams and focusing on features rather than on components. Furthermore, the transition towards continuous integration requires an automated build and test system, a main version control branch to which code is continuously delivered, and modularize the development. Their findings reveal that in order to move from continuous integration to continuous deployment, organizational units such as product management must be fully involved, and work in close collaboration with a very active lead customer is needed when exploring the product concept further. The authors suggest that in order for a software organization to reach the R&D experiment system stage two key actions should be in place: the product must be instrumented to allow that field data can be collected in actual use, and proper organizational capabilities must be developed in order to effectively use the collected data for testing new ideas with customers.

In [Ståhl and Bosch, 2014] the authors studied the continuous integration stage, pointing out that there is no homogeneous practice of continuous integration in the industry. They propose a descriptive model that allows studying and evaluating the different ways in which continuous integration can be viewed. An architecture that supports continuous experimentation in embedded systems is presented in [Eklund and Bosch, 2012]. There, the authors explore the goals of an experiment system, develop experiment scenarios, and construct an architecture that supports the goals and scenarios. The architecture combines an experiment repository, data storage, and software to be deployed on embedded devices via over-the-air data communication channels. However, the main type of experiment is confined to A/B testing, and the architecture is considered mainly from the perspective of a software development team rather than a larger product development organization.

The Hypothesis Experiment Data-Driven Development (HYPEX) is presented in [Olsson and Bosch, 2014]. The goal of this model is to shorten the feedback loop to customers. It consists of a loop where potential features are generated into a feature backlog, from which features are selected and a corresponding expected behavior is defined. The expected behavior is used to implement and deploy a minimum viable feature (MVF). Observed and expected behavior is compared using a gap analysis, and if a sufficiently small gap is identified, the feature is finalized. On the other hand, if a significant gap is found, hypotheses are developed to explain it, and alternative MVFs are developed and deployed, after which the gap analysis is repeated. The feature may also be abandoned if the expected benefit is not achieved.

2.2.3 Continuous experimentation

Previous works have presented case studies that exhibit different aspects concerning continuous experimentation. In [Steiber and Alänge, 2013], it is presented a study of the continuous experimentation model followed by Google, analyzing a success story of this approach. In [Tang et al., 2010], the authors describe an overlapping experiment infrastructure, developed at Google, that allows web queries in a search engine to be part of multiple experiments, thus facilitating more experiments to be carried out at a faster rate. In [Adams et al., 2013], the authors present a case study on the implementation of Adobe's Pipeline, which is a process that is based on the continuous experimentation approach.

The differences between traditional development and the continuous approach are analyzed in [Bosch, 2012], showing that in the context of the new, continuous software development model, R&D is best described as an "innovation experiment system" approach where the development organization constantly develops new hypotheses and tests them with certain groups of customers. This approach focuses on three phases: pre-deployment, non-commercial deployment, and commercial deployment. The authors present a first systematization of this so-called "innovation experiment system" adapted for software development for embedded systems. It is argued that aiming for an "innovation experiment system" is equally valid for embedded systems as it is in the case of cloud computing and Software-as-a-Service (SaaS), and that the process could be similar in both cases. That is, requirements should evolve in real time based on data collected from systems in actual use with customers.

We define our engineering model for the systematic development of value-added software for software-intensive systems, based on the "innovation experiment system" mentioned above. We denominate our engineering model as the Rapid Iterative value creation Gained through High-frequency Testing (RIGHT) model for continuous experimentation. In our model, experiments are derived from business strategies and are aimed at assessing assumptions derived from those strategies, potentially invalidating or supporting the strategy. Previous works have explored the application of a framework for linking business goals and strategies to the software development activities (e.g., [Basili et al., 2007, Münch et al., 2013]). However, those works have not considered the particular traits of an experiment system. The model presented also describes the platform infrastructure that is necessary to establish the whole experiment system.

The building blocks presented in this paper, although generalizable with certain limitations, are derived from startup environment where the continuous experimentation approach is not only well suited but possibly the only viable option for companies to grow. Our work has similarities to the "Early Stage Startup Software Development Model" (ESSSDM) in [Bosch et al., 2013], which extends existing Lean Startup approaches offering more operational process support and better decision-making support for startup companies. Specifically, ESSSDM provides guidance on when to move product ideas forward, when to abandon them, and what techniques to use, in the process of validating product ideas.

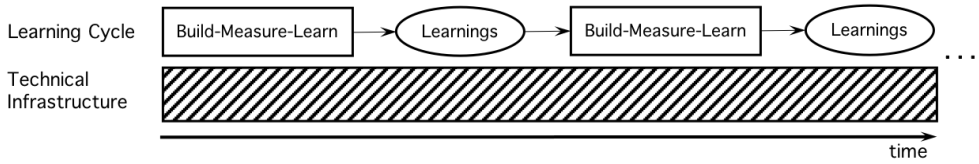


Figure 2.1: Initial model for engineering value-added software software-intensive systems

2.3 Research Methodology

To derive the proposed model we use a design science research approach [Von Alan et al., 2004], which purpose is to derive a technological rule that can be used in practice to achieve a desired outcome in a certain field of application [Aken, 2004]. Specifically, we first build an initial model based mainly on what can be found in the literature, to then refine such model based on the observations gained through empirical evaluation. The initial model is shown in Figure 2.1, depicting a learning cycle for the process and technical infrastructure (composed of a data repository, analysis tools, and continuous delivery system). The empirical evaluation is conducted according to a multiple case study [Yin, 2009], in which we put to test the initial model and apply progressively adjustments based on our observations until reaching the model we present in our results in Section 2.4.

Our initial model follows the idea of continuous experimentation. That is, an engineering approach to build software-intensive systems which is based on field experiments with relevant stakeholders, typically customers or users, but potentially also with other stakeholders such as investors, third-party developers, and software ecosystem partners. The model consists of repeated Build-Measure-Learn blocks, supported by an infrastructure. Each Build-Measure-Learn block yields learnings which are used as input for the next block.

The Build-Measure-Learn model provides structure to the activity of conducting experiments, and connects product/service vision, business strategy, and technological product/service development through experimentation. In particular, the conventional phases of the software engineering cycle (i.e., requirements, design, implementation, testing, deployment, and evolution/maintenance) are integrated and aligned by empirical information gained through experimentation. The model can be considered as a facilitator for incremental innovation as defined by Henderson and Clark [Henderson and Clark, 1990], but the model itself, as well as the transition to continuous experimentation in general require significant organizational capabilities that depart from traditional engineering models.

2.3.1 Study setup

A Environment

The case projects considered for the study took place in the Software Factory laboratory at the Department of Computer Science of University of Helsinki. The Software Factory is an educational platform for research and industry collaboration [Fagerholm et al., 2013].

In Software Factory projects, teams of Master-level students use contemporary tools and processes to deliver working software prototypes in close collaboration with industry partners. The goal of Software Factory activities is to provide students with means for applying their advanced software development skills in an environment with working life relevance and to deliver meaningful results for their customers [Münch et al., 2013].

B Cases 1, 2, and 3

B.1 Software company involved. Tellybean Ltd.[1] is a small Finnish startup that develops a video calling solution for the home TV set. The company was a customer in three Software Factory projects with the aim of creating an infrastructure to support measurement and management tasks of the architecture of their video calling service. Tellybean aims at delivering a life-like video calling experience. Their value proposition is defined as: "the new home phone as a plug and play experience." Their target is late adopter customers who are separated from their families. The company puts special emphasis on discovering and satisfying needs of the elderly, making ease of use the most important non-functional requirement of their product.

The primary means for service differentiation in the marketplace for Tellybean are affordability, accessibility, and ease of use. For the premiere commercial launch and to establish the primary delivery channel of their product, the company aims at partnering with telecom operators. The company had made an initial in-house architecture and partial implementation during a pre-development phase prior to the Software Factory projects. A first project was conducted to extend the platform functionality of this implementation. A second project was conducted to validate concerns related to the satisfaction of operator requirements. After this project, a technical pivot took place, with major portions of the implementation being changed; the first two projects contributed to this decision. A third project was then conducted to extend the new implementation with new features related to the ability to manage software on already delivered products, enabling continuous delivery. The launch strategy can be described as an MVP launch with post-development adaptation. The three projects conducted with this company are connected to establishing a continuous experimentation process and building capabilities to deliver software variations on which experiments can be conducted. They also provided early evidence regarding the feasibility of the product for specific stakeholders, such as operator partners, developers, and release managers.

B.2 Product/Service. The Tellybean video calling service has the basic functionalities of a home phone: it allows making and receiving video calls and maintaining a contact list. The product is based on an Android OS set-top-box (STB) that can be plugged into a modern home TV. The company maintains a backend system for mediating calls to their correct respondents. While the server is responsible for routing the calls, the actual video call is performed as a peer to peer connection between STBs residing in the homes of Tellybean's customers.

[1] https://www.tellybean.com/

Table 2.1: Scope of each of the projects of cases 1, 2, and 3

Project	High-level goal	Motivation
Project 1	Allow operators to see metrics for calls made by the video call product's customers	Extract and analyze business critical information
Project 2	Ensure to developers that the product's system architecture is scalable and robust	Know the limitations of the system
	Allow developers to know the technical weaknesses of the system	Predict the needs for scalability of the platform
	Allow developers to receive suggestions for alternative technical architecture options	Consider future development options
Project 3	Allow the technical manager to push an update to the Tellybean set-top-boxes with a single press of a button	Deploy upgrades to the software on one or multiple set-top-boxes

B.3 Projects 1, 2, and 3. The company played the role of a product owner in three Software Factory projects. The aim of the first two projects was to create new infrastructure for measuring and analyzing usage of their product in its real environment. This information was important in order to establish the product's feasibility for operators and for architectural decisions regarding scalability, performance, and robustness. From the perspective of the proposed engineering model, the first two projects were used to validate the steps required to establish a continuous experimentation process. The third project at Software Factory delivered an automated system for managing and updating the STB software remotely. This project was used to investigate factors related to the architecture needs for continuous experimentation. Table 2.1 summarizes the goals and motivations of the projects in detail. A team of students of between three to seven members was dedicated to each project, with an additional company representative interacting continuously with the team.

Project 1. The goal of Tellybean on Project 1 was to build the means for measuring performance of their video calling product in its real environment. Thus, the development focused on building a browser-based business analytics system. The team was also assigned to produce a back-end system for storing and managing data related to video calls, in order to satisfy operator monitoring requirements. The project was carried out in seven weeks by a team of four students. Competencies required in the project were database design, application programming, and user interface design.

After the project had been completed, both students and the customer deemed that the product had been delivered according to the customer's requirements. Despite the fact that some of the foundational requirements changed during the project due to discoveries of new technological solutions, the customer indicated satisfaction with the end-product. During the project, communication between the customer and the team was frequent and flexible.

Overall, Project 1 constituted a first attempt at conducting continuous experimentation. The goal of the experiment was to gain information about the performance of the system architecture and its initial implementation. The experiment arose from operator needs to monitor call volumes and system load, which is a requirement that Tellybean's product developers deemed necessary to be able to partner with operators. It was clear that there existed a set of needs arising from operator requirements, but it was not clear how the information should be presented and what functionality was needed to analyze it. From a research perspective, however, the exact details of the experiment were less important than the overall process of starting experimentation.

Project 2. This project was aimed at performing a system-wide stress test for the company's video calling service infrastructure. The Software Factory team of four students produced a test tool for simulating very high volumes of calls. The tool was used to run several tests against Tellybean's existing call mediator server. The test software suite included a tool for simulating video call traffic.

The purpose of the experiment was a counterpart to the experiment in Project 1. In Project 2 the focus was on the implication of the experiments for developers, while Project 1 was focused on the operator requirements. The initial system architecture and many of the technical decisions had been questioned. The project aimed to provide evidence for decision-making when revisiting these initial choices.

The team found significant performance bottlenecks in Tellybean's existing proof-of-concept system and analyzed their origins. Solutions for increasing operational capacity of the current live system were proposed and some of them were also implemented. Towards the end of the project, the customer suggested for the second experiment to have another round in the continuous experimentation cycle where findings from the first cycle resulted in a new set of questions to consider for experimentation.

Project 3. For this project Tellybean aimed to create a centralized infrastructure for updating their video calling product's software components. The new remote software management system would allow the company to quickly deploy software updates to already delivered STBs. The functionality was business critical to the company and its channel partners, since it allowed updating the software without having to travel on-location to each customer to update their STBs. The new instrument enabled the company to establish full control of their own software and hardware assets.

The project was followed by a team of five students. The team delivered a working prototype for rapid deployment of software updates. In this project, the need for a support system to deliver new features or software variations was addressed. We considered the architectural requirements for a continuous delivery system that would support continuous experimentation.

C Case 4

C.1 Software company involved. Memory Trails Ltd. (Memory Trails) is a small Finnish startup that develops a well-being service which helps users define, track, and receive assistance with life goals. For a period of three months, the company participated in a Software Factory project that aimed to develop a backend recommendation engine for the service, improve the front-end user experience, and validate central assumptions in the service strategy. The service targets adults who wish to improve their quality of life and change patterns of behaviour to reach different kinds of life goals.

Differently from the other company of cases 1, 2, and 3, where the focus was on establishing a continuous experimentation process and building capabilities to deliver software variations for experimentation, the project with this company focused on some of the details related to deriving experiments. In particular, our objective was to uncover how assumptions can be identified in initial product or service ideas, where the assumptions are main candidates to formulate experiments.

C.2 Product/Service. Memory Trails provided an initial user interface and backend system prototype which demonstrated the general characteristics of the application from a user perspective. The users interact with photos which can be placed in different spatial patterns to depict emotional aspects of their goals. The application then guides the users to arrange the photos as a map, showing the goal, potential steps towards it, and aspects that qualify the goals. For example, a life goal may be to travel around the world. Related photos could depict places to visit, moods to be experienced, items necessary for travel such as tickets, etc. The photos could be arranged, e.g., as a radial pattern with the central goal in the middle, and the related aspects around it, or as a timeline with the end goal to the right and intermediate steps preceding it.

C.3 Project 4. In this project, two high-level assumptions were identified. The customer assumed that automatic, artificial intelligence-based processing in the backend could be used to automatically guide users towards their goals, providing triggers, motivation, and rewards on the way. Also, the customer assumed that the motivation for the regular use of the application would come from interacting with the photo map. Since the automatic processing depended on the motivation assumption, the latter became the focus of experimentation in the project. The customer used versions of the application in user tests during which observation and interviews were used to investigate if the assumption was true. For the purposes of our multiple-case study, we used the project to validate the link in our model between product vision, business model and strategy, and experiment steps.

2.3.2 Study design

The main *goal* of our multiple-case study is to ground our model for the systematic development of value-added software for software-intensive systems in empirical observations. To this end, we collected information that would help us understand the prerequisites for performing continuous experimentation, the associated constraints and challenges, and

the logic of integrating experiment results into the business strategy and the development process.

To guide our process we aim at answering the following questions:

ch2-res-Q1 What is a suitable process model for performing systematic, experiment-based engineering of software for software-intensive systems?

ch2-res-Q2 What is a suitable infrastructure architecture for performing systematic, experiment-based engineering of software for software-intensive systems?

We used four different sources of data in our analysis: participant observation, analysis of project artifacts, group analysis sessions, and individual interviews. We subsequently discuss the details of the data collection and analysis.

During the projects, we observed the challenges that the companies faced related to follow the continuous experimentation model. At the end of each project, an in-depth debriefing session was conducted to gain retrospective insights into the choices made during the project, and the reasoning behind them. In addition to these sources, we interviewed three company representatives from Tellybean to understand their perception of the projects and to gain data which could be matched against our model. We also conducted a joint analysis session with the project team and two representatives from Memory Trails to further match insights around the experimentation process in their projects with our model.

During the analysis phase, the project data was examined for information relevant to the research question. We categorized the pieces of evidence according to their relation to the process or the infrastructure. We sought to synthesize the observations made and understanding gained during the projects with evidence from the retrospective sessions and interviews to improve the confidence of the evidence. The evidence obtained was then matched with our initial model. We adjusted the model and introduced new process steps and infrastructure components that supported the aspects required according to the evidence. When all the evidence had been considered, we evaluated the result as a whole and made some adjustments and simplifications based on our understanding and judgment.

2.4 Results

The main result from our analytical and empirical approach is our engineering model (based on the continuous experimentation process) for the systematic development of value-added software for software-intensive systems. We call our model the *Rapid Iterative value creation Gained through High-frequency Testing (RIGHT)* model for engineering software for software-intensive systems. Next, we introduce first the specific process followed by the engineering model, and then the infrastructure on top of which the process is to take place.

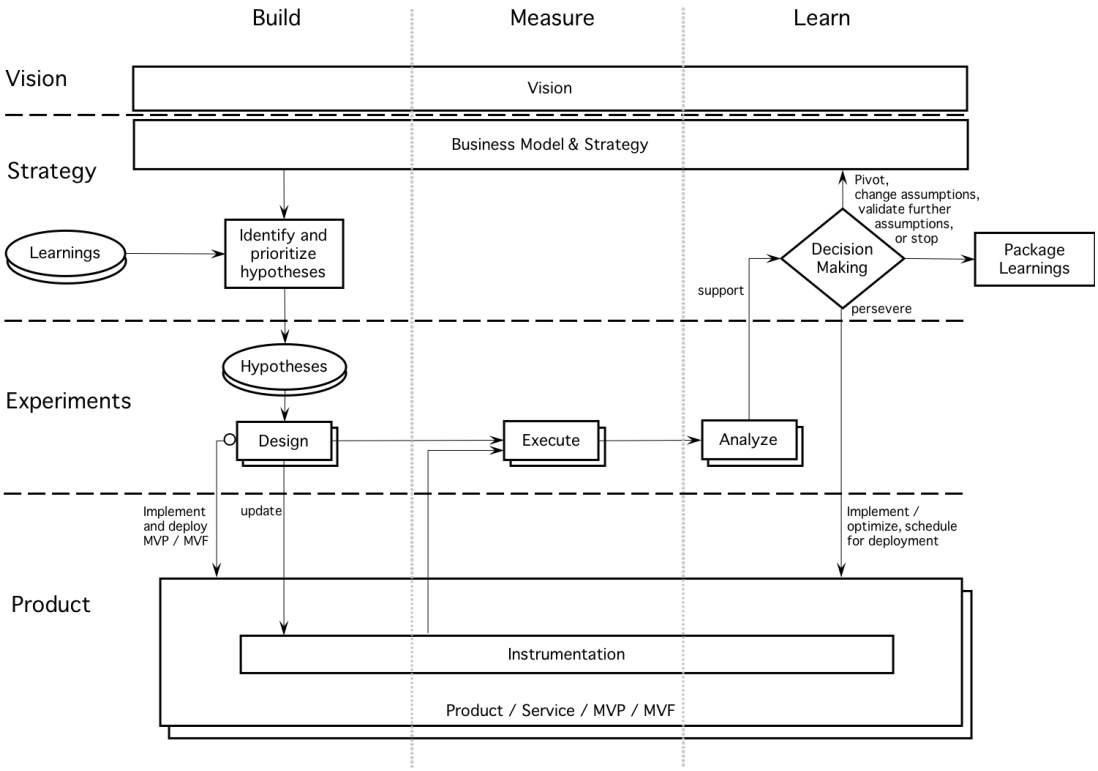


Figure 2.2: Process model for engineering value-added software for software-intensive systems

2.4.1 Process model

In this section we describe the process model that corresponds to the initial engineering model introduced above with the additional details that we delineated based on our empirical observations. The model is shown in Figure 2.2. A general vision of the product or service is assumed to exist from the beginning. According to the Lean Startup methodology [Ries, 2011], this vision is in general stable and is based on knowledge and beliefs held by the entrepreneur. The vision is connected to the business model and strategy, which is a description of how to execute the vision. The business model and strategy are more flexible than the vision, and consist of multiple assumptions regarding the actions required to bring a product/service to market that fulfills the vision and is sustainably profitable. However, each assumption has inherent uncertainties. The workaround proposed to reduce the uncertainties is to conduct experiments. An experiment operationalizes the assumption and states a hypothesis that can be subjected to experimental testing in order to gain knowledge regarding the assumption. The highest-priority hypotheses are selected first. The hypothesis can be used to implement and deploy either a Minimum Viable Product (MVP) or Minimum Viable Feature (MVF), which is used in the experiment and is associated to the necessary instrumentation. Simultaneously, an experiment is designed to test the hypothesis. The experiment is then executed and data from the MVP/MVF is collected in accordance with the experimental design. The resulting data is analyzed, concluding the experimental activities.

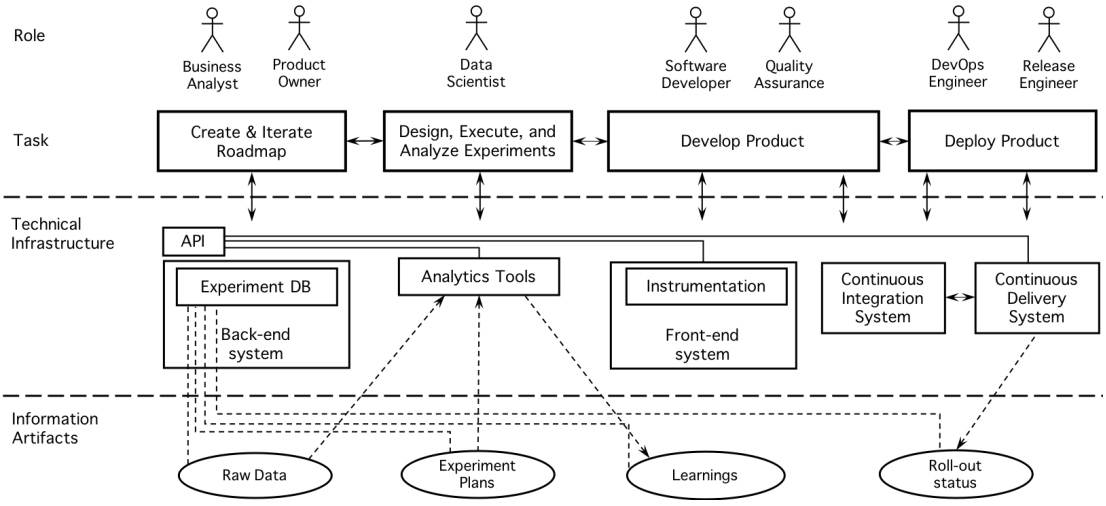


Figure 2.3: Infrastructure model for engineering value-added software for software-intensive systems

Once the experiment and corresponding analysis have taken place, the results are used at the strategy level to support decision-making. According to the Lean Startup terminology [Ries, 2011], the decision can be to either to "pivot" or "persevere", with a third alternative being: to change the assumptions in the light of new information. If the experiment provides support to the hypothesis, and consequently to the assumption at the strategy level, a full product/service or feature is produced. The strategic decision in this case is to persevere with the current strategy. If, on the other hand, the hypothesis was proved to be false, invalidating the assumption at the strategy level, the decision is to pivot and alter the strategy. Alternatively, the tested assumption could be changed, but not completely rejected, depending on the aim of the experiment and its results.

2.4.2 Infrastructure model

The experiments of the continuous experimentation process need to be conducted over a specific infrastructure. Figure 2.3 outlines such infrastructure, including roles and associated tasks, technical infrastructure, and information artifacts. In a small company, such as a startup, a small number of persons will handle the different roles and one person may have more than one role. In a large enterprise, the roles are handled by multiple teams. Seven roles are defined to handle four types of tasks. A business analyst and a product owner, or a product management team, together handle the creation and iterative updating of the strategic roadmap. In order to achieve this, they consult existing experimental plans, results, and learnings, which are to be found in a back-end system. As plans and results accumulate and are stored, they may be reused in further development of the roadmap. The business analyst and product owner work with a data scientist role, which is usually a team with diverse skills, to communicate the assumptions of the roadmap, outlining sources of uncertainty that need to be tested.

The data scientist is devoted to the design, execution, and analysis of the experiments. For this purpose, a variety of tools are used, which access raw data in the back-end

system. Conceptually, raw data and experiment plans are retrieved, analysis performed, and results produced in the form of learnings, which are stored back into the back-end system. The data scientist also communicates with the developer and quality assurance roles. These roles handle the development of MVPs, MVFs, and the final product. The collaboration between the developer and quality assurance roles and the data analyst role begins by establishing the necessary instrumentation into the front-end system, which is part of the software that is delivered to the user. In the case of a "persevere" decision, these roles work to fully develop or optimize the feature/product/service and submit it for deployment into production.

MVPs, MVFs, and final products/services are deployed to users after having passed through continuous integration and continuous delivery systems. A DevOps engineer acts as the mediator between the development team and operations, and the release engineer role may oversee and manage the releases currently in production. Importantly, the continuous delivery system provides information on software roll-out status, allowing other roles to monitor the experiment execution and, among other activities, gain an understanding of the conditions under which the software was deployed to users and of the sample characteristics and response rate of the experiment. Crosscutting concerns such as user experience may require additional roles working with several of the roles mentioned here. To simplify the schematic of Figure 2.3, we have omitted the various roles that relate to operations (e.g., site reliability engineer).

The back-end system consists of an experiment database which (conceptually) stores raw data (collected from the software instrumentation), experiment plans (including programmatic features of sample selection and other logic needed to conduct the experiment), and experiment results. The back-end system and the database are accessible through a general API, which in practice may consist of multiple APIs, databases, servers, etc. The experiment database enables a product architecture in which the deployed software is configured for experiments to take place at runtime. Therefore, it is not always required that a new version of the software or the accompanying instrumentation is shipped to users prior to an experiment. That is, the experimental capability can be built into the delivered software as a configurable variation scheme. The delivered software fetches configuration parameters for new experiments, reconfigures itself, and sends back the resulting measurement data, eliminating the need to perform the develop-product and deploy-product tasks. For larger changes, a new software version may be required, and the full set of tasks should be performed.

2.4.3 Applicability of the model

Each of the four case projects that we consider relates to different aspects of the proposed engineering model. Our findings support the need for systematic integration of all levels of software product/service development, especially when dealing with rapid development. The key issue is to develop a products/services that customers are willing to acquire.

We found that in general the vision of the product/service remains stable, not requiring major changes, throughout the evolution of the software company. The change should rather concern the strategy by which the vision is realized, including the features

that should be implemented, their design, and the technological platform on which the implementation is based. Specifically, in the case of Tellybean, in spite of the various adaptations that the product/service and strategy has experienced, the main vision of the company remains the same.

Focusing in particular on the experiments, although the design of an experiment seems self-evident in hindsight, developing one based on the information available in actual software projects, especially new product/service development, is not an easy task. There are multiple possibilities for what to experiment on, and it is not obvious how to choose the first experiment or each next experiment after that. Our case projects showed that initiating the continuous experimentation process is a significant task in its own right and involves much learning.

A Project 1

In the first project, the new business analytics instrument allowed Tellybean to obtain insights on their system's statistics, i.e., a way of getting feedback. They were able to gain a near real-time view on call related activities, yielding business critical information for deeper analysis. The presence of the call data could be used as input for informed decisions. It also allowed learning about service quality and identifying customer call behaviour patterns. Based on the customer's comments, such information would be crucial for decision-making regarding the scaling of the platform. Excess capacity could thus be avoided and the system would be more profitable to operate while still maintaining a good service level for end users. The primary reason for the wish to demonstrate such capabilities was the need to satisfy operator needs. To convince operators to become channel partners, the ability to respond to fluctuations in call volumes was identified as critical. Potential investors would be more inclined to invest in a company that could convince channel operators of the technical viability of the service.

The high-level goal of the first project could be considered as defining a business hypothesis to test the business model from the viewpoint of the operators. The project delivered the needed metrics as well as a tool-supported infrastructure to gather the necessary data. These results could be used to set up an experiment to test the business hypotheses.

Table 2.2 and Table 2.3 show the parts of our process and infrastructure models, respectively, that were instantiated in Project 1.

The project instantiated a few basic elements of the RIGHT process model. The chosen business model and strategy was to offer the video calling service through operator partnerships. In order for the strategy to be successful, the company needed to demonstrate the feasibility of the service in terms of operator needs and requirements. This demonstration was targeted to operators themselves but also to other stakeholders, such as investors, who assessed the business model and strategy. The hypothesis that was set to be tested was not precisely defined in the project, but could be summarized as "operators will require system performance management analysis tools in order to enter a partnership". The experiment, which was obviously not a controlled one but rather conducted as part of investor and operator negotiations, used the analytics instrument

Table 2.2: Process Model instantiations in Project 1

Process model instantiation	
Vision	Video calling in the home
Business model and strategy	Offer video calling through operator partnerships (+ assumptions about architecture and product implementation strategies)
Hypotheses	"Operators will require performance management analysis tools in order to enter a partnership"
Design, execute, analyze	Rudimentary
MVF	Analytics instrument
Decision making	Start architectural pivot (continued in Project 2)
	Start product implementation strategy pivot (continued in Project 2)
	Validate further assumptions (regarding architecture and product implementation)

Table 2.3: Infrastructure Model instantiations (only applicable parts) in Project 1

Infrastructure model instantiation	
Roles	Business analyst, product owner (played by company leadership), software developer (played by Software Factory students)
Technical infrastructure	Analytics Tools (MVF developed in project)
Information artifacts	Learnings (not formally documented in project)

developed in the project to assess whether the assumption was correct, thus instantiating an MVF, and making a rather basic experiment execution and analysis. Based on this information, some decisions were made, such as to start investigating alternative architectures and product implementation strategies.

B Project 2

In the second project, Tellybean was able to learn the limitations of their current proof-of-concept system and its architecture. An alternative call mediator server and an alternative architecture for the system were very important for the future development of the service. The lessons learned in the second project, combined with the results of the first, prompted them to pivot heavily regarding the technology, architectural solutions, and development methodology.

The high-level goals of the second project could be considered to define and test a solution hypothesis that addresses the feasibility of the proposed hardware-software solution. The project delivered an evaluation of the technical solution as well as improvement

Table 2.4: Process Model instantiations in Project 2

Process model instantiation	
Vision	Video calling in the home
Business model and strategy	Offer video calling through operator partnerships (+ assumptions about architecture and product implementation strategies)
Hypotheses	"Product should be developed as custom hardware-software codesign" and "Architecture should be based on Enterprise Java technology and be independent of TV set (which acts only as display)"
Design, execute, analyze	Prototype implementation; evaluate current solution proposal
MVF	Alternative call mediator server; alternative system architecture
Decision making	Architectural pivot (Android-based COTS hardware and OS)
	Product implementation strategy pivot (do not develop custom hardware)

proposals. The analysis showed that the initial architecture and product implementation strategy were too resource-consuming to carry out fully. The results were used by the company to modify their strategy. Instead of implementing the hardware themselves, they opted for a strategy where they would build on top of generic hardware platforms and thus shorten time-to-market and development costs. Table 2.4 and Table 2.5 show the parts of our process and infrastructure models, respectively, that were instantiated in Project 2.

Table 2.5: Infrastructure Model instantiations (only applicable parts) in Project 2

Infrastructure model instantiation	
Roles	Business analyst, product owner (played by company leadership), software developer (played by Software Factory students)
Technical infrastructure	Analytics tools (from previous project)
Information artifacts	Learnings (not formally documented in project)

C Project 3

In the third project, the capability for continuous deployment was developed. The STBs could be updated remotely, allowing new features to be pushed to customers at very low cost and with little effort. The implications of this capability are that the company is able to react to changes in their technological solution space by updating operating system

and application software. Furthermore, the company can react to emerging customer needs by deploying new features and testing feature variants continuously.

The high-level goals of the third project could be considered as developing a capability that allows for automating the continuous deployment process. The prerequisite for this is a steady and controlled pace of development where the focus is on managing the amount of work items that are open concurrently in order to limit complexity. At Tellybean, this is known as the concept of one-piece flow, which they define as finishing one thing before moving on to the next.

The parts of our model instantiated by Project 3 are shown in Table 2.6 (process) and Table 2.7 (infrastructure). In particular, the project focused on the role of a continuous delivery system in relation to the tasks that need to be carried out for continuous experimentation, meaning that top and rightmost parts of Figure 2.3 were instantiated, as detailed in Table 2.7.

Table 2.6: Process Model instantiations in Project 3

Process model instantiation	
Vision	Video calling in the home
Business model and strategy	Offer video calling through operator partnerships (+ assumptions about architecture and product implementation strategies)
Hypotheses	"Capability for automatic continuous deployment is needed for incremental product development and delivery"
Design, execute, analyze	Project focused on instantiating parts of infrastructure architecture model and did not include a product experiment
MVF	Prototype for rapid deployment of software updates
Decision making	Persevere

Table 2.7: Infrastructure Model instantiations (only applicable parts) in Project 3

Infrastructure model instantiation	
Roles	Business analyst, product owner (played by company leadership), software developer (played by Software Factory students), DevOps engineer, release engineer (played by company CTO and other technical representatives; also represented by user stories with tasks for these roles)
Technical infrastructure	Continuous integration system, continuous delivery system (MVF developed in project)
Information artifacts	Roll-out status

D Project 4

In this project, it was initially difficult to identify what the customers considered to be the main assumptions. However, once the main assumptions became clear, it was possible to focus on validating them. This supports the idea that although it is straightforward in theory to assume that hypotheses should be derived from the business model and strategy, it may not be straightforward in practice. In new product/service development, the business model and strategy is not finished, and, especially in the early cycles of experimentation, it may be necessary to try several alternatives and spend effort on modelling assumptions until a good set of hypotheses is obtained. We therefore found it useful to separate the identification and prioritization of hypotheses on the strategy level from the detailed formulation of hypotheses and experiment design on the experiment level. Table 2.6 (process) and Table 2.7 (infrastructure) show the instantiations of our model in Project 4.

Table 2.8: Process Model instantiations in Project 4

Process model instantiation	
Vision	Well-being service for defining, tracking, and receiving assistance with life goals
Business model and strategy	Product and service recommendations, automated recommendation engine for motivating progress towards goals
Hypotheses	"Motivation for continued use comes from interacting with photo map"
	"Automatic recommendation engine will automatically guide users to reach goals" (depends on first hypothesis)
Design, execute, analyze	User tests with observation and interviews
MVF	HTML 5-based table-optimized application
Decision making	Product implementation strategy pivot (focus on social interaction rather than automated recommendations)

There were two main assumptions in Project 4: *a*) the interaction with the photo map would retain users, and *b*) an automated process of guiding users towards goals

Table 2.9: Infrastructure Model instantiations (only applicable parts) in Project 4

Infrastructure model instantiation	
Roles	Business analyst, product owner (played by company leadership), software developer (played by Software Factory students)
Technical infrastructure	Instrumentation, front-end system
Information artifacts	Learnings

is feasible.. The assumption that the regular use of the application would come from interacting with the photo map was shown to be incorrect. Users would initially create the map, but would not spend much time interacting (e.g., adding or changing photos, rearranging the map, adding photo annotations) with it. Instead, users reported a desire for connecting with other users to share maps and discuss life goals. Also, they expressed willingness to connect with professional or semi-professional coaches to get help with the implementation of their life goals. The social aspect of the service had been overlooked. Understanding if this was due to familiarity with existing social media applications was left uninvestigated. In any case, the assumption was invalidated and, as a result, the assumptions regarding automated features for guiding users towards goals were also invalidated. The investigation indicated that users were motivated by the potential for interaction with other users, and that these interactions should include the process of motivating them to reach their goals. It is important to note that it was possible to invalidate the two hypotheses because they were dependent. The process of identifying and prioritizing hypotheses separately from detailed formulation of hypotheses and experiment design makes it possible to choose the order of experiments in a way that gains the maximum amount of information with the minimum number of experiments. Testing first the most fundamental assumptions, where all others rely on, makes it possible to eliminate other assumptions with no additional effort.

Project 4 also revealed challenges related to the way of instrumenting the application for data collection. It was difficult to separate the process of continuous experimentation from the technical prerequisites for instrumentation. In many cases, substantial investments into technical infrastructure are needed before experiments can be carried out. These findings led to the roles, the high-level description of the technical infrastructure, and the information artifacts in the infrastructure we are proposing (see Figure 2.3).

Many experiments are also possible without advanced instrumentation. The results of Project 4 indicate that experiments may typically be large, or target high-level questions, in the beginning of the product/service development cycle. They may address questions and assumptions that are central to the whole product or service concept. Later stages of experimentation may address more detailed aspects, and may be considered optimization of an existing product or service.

2.5 Discussion

2.5.1 Validity of the research methodology

A particular limitation of our multiple-case study is the use of relatively short projects with student participants. Students carried out the technical software development and analysis tasks in the projects, while the researchers handled tasks related to identification of assumptions, generation of hypotheses, and higher-level planning tasks together with customer representatives. While it is reasonable to expect that professional software developers would have reached a different level of quality and rigour in the technical tasks, we consider it likely that the findings are applicable beyond student projects, since the focus of this paper is not on the technical implementation but on the integration

of experiment results in the product development cycle and the software development process. The length of the projects implies that at most one experimental cycle could be carried out in a single project. Thus the first case company completed three, and the second case company one experimental cycle. In a real setting, multiple experimentation rounds would be carried out over an extended period of time, proceeding from experiments addressing the most important assumptions with the highest impact towards increasing detail and optimization. The findings of this study should be considered to apply mostly in the early stages of experimentation.

2.5.2 Limitations of the model

The main limitations of the proposed engineering model that we have identified are:

- Although continuous experimentation may help to deliver value to users by obtaining as much information as possible to take decisions, it is not guaranteed that the right decision will be taken.
- The interpretation of the experiments is in many cases challenging and may not necessarily derive on the correct assessment.
- Although the model calls for the prioritization of the assumptions, there might be situations in which is not clear which assumption is more relevant than the others.
- In some cases an experimental approach may not be suitable at all. For example, certain kinds of life-critical software or software that is used in environments where experimentation is prohibitively expensive, may preclude the use of experiments as a method of validation. However, it is not clear how to determine the suitability of an experimental approach in specific situations, and research on this topic could yield valuable guidelines on when to apply a model like the one that is proposed.

2.6 Summary

In this chapter we presented our answer to *RQ1-a*. That is, we introduced an engineering model to build value-added software for software-intensive systems in a systematic way.

We have introduced the engineering model through a process and an infrastructure models, which, although are not to be considered final, provide key aspects that have to be in place to successfully deliver value to users. The creation of value is based on the application of a continuous experiment-based development cycle, which we propose to apply in a systematic manner. The experiments centered around the users are the main mechanisms of the engineering model we propose, with new features or products/services being pushed to the users to learn from their reaction and decide on the next steps of the development.

The proposed models were derived following analytical and empirical methodologies. Specifically, we began with an initial engineering model based on continuous experimentation models found in the literature. Then, we developed further and consolidated our engineering model based on the analysis of the results of a multiple case study we conducted with two startup software companies.

Chapter 3

Phases of the Engineering of Software for Pervasive Systems

3.1 Introduction

In this chapter we give solution to one of the questions in which we have divided our research question *RQ1*, namely, *RQ1-b*: What has been considered in the literature as the implicit engineering cycle to build software for pervasive systems?

To this end we present a systematic literature review on the engineering of software for pervasive systems. We use the systematic review methodology in the light of the considerable size and heterogeneity of the literature in this area (e.g., a quick Google search may produce not less than ten thousand results), and that existent works mostly present only informal literature surveys.

We search for clues about the implicit engineering cycle, since to the best of our knowledge no work has proposed a model for engineering software, specifically devoted for pervasive systems. The findings of this chapter are intended to help us to better understand the engineering process through the state-of-the-art approaches that have been proposed around one or more phases of the cycle. We describe the main concerns and limitations of the selected approaches and provide the main research challenges that the corresponding studies have identified.

This chapter is organized as follows. The methodology employed, i.e., systematic review, is detailed in Section 3.2, including the definition of the research questions, the search and selection processes, and the synthesis approach. In Section 3.3 we present the results of the review, describing the focus of each of the approaches considered and the limitations identified by their authors. This is followed by a description of the open issues and research challenges that according to the selected works are currently relevant in the domain of software development for pervasive systems. Section 3.4 is where we discuss about the validity and limitations of our systematic review, as well as about the importance that it may have for practitioners and researchers. The last section, Section 3.5 provides a summary of the chapter.

3.2 Research Methodology

To delineate the model for engineering software for ubiquitous systems we use as research methodology an evidence-based approach, specifically a systematic review of the literature. Our systematic review seeks to find, evaluate, and synthesize the most relevant approaches that have been proposed to date in the literature for any of the issues related to the software development life cycle for ubiquitous systems. We classify the approaches according to the phase they relate to in the development life cycle and the general concerns they address.

3.2.1 Background

A systematic literature review is a systematic approach to building a body of knowledge about a particular topic or research question(s), and to identify problems for future research and support decision making and technological selection [Zhang and Babar, 2013].

Reviews, surveys, and state of the art overviews have been prevalent in the literature on software development for ubiquitous systems (e.g., [Shi et al., 2011a, Raychoudhury et al., 2013, Krupitzer et al., 2015, Martin et al., 2011]). Lots of articles have been published about the challenges of this type of development, most of the time based on rather informal literature reviews and the authors' experience (e.g., [Abowd, 2012, Conti et al., 2012, Tang et al., 2011, Cook et al., 2009, Want and Pering, 2005, Edwards and Grinter, 2001, Abowd, 1999]).

The reviews that can be found in the literature related to the development of software for ubiquitous systems have typically focused on general concepts such as autonomic computing [Huebscher and McCann, 2008] and self-adaptive systems [De Lemos et al., 2013], as well as on particular implementations of ubiquitous systems such as smart home [Solaimani et al., 2015, Alam et al., 2012, Chan et al., 2008] and ambient intelligence [Cook and Das, 2007, Cook et al., 2009], specific phases of the development life cycle (e.g., feedback [Brun et al., 2009], design [Tang et al., 2011]), specific tasks such as situation identification [Ye et al., 2012], specific concerns such as application mobility [Yu et al., 2013], or particular techniques or tools such as middleware [Raychoudhury et al., 2013, Martin et al., 2011] and models at runtime [Szvetits and Zdun, 2013].

Among the reviews we found, only one deals with software engineering approaches but targets a more general type of systems than ubiquitous systems, that is, self-adaptive systems [Krupitzer et al., 2015]. All reviews we found followed an informal process except for two, one about smart home [Solaimani et al., 2015] and one about models at runtime [Szvetits and Zdun, 2013].

3.2.2 Research questions

We follow the systematic literature review methodology in order to answer the following questions:

ch3-res-Q1. What are the stages of the development life cycle of software for ubiquitous systems that have been mainly considered?

ch3-res-Q2. What are the main approaches that have been proposed for each of these stages and what are their limitations?

ch3-res-Q3. What are the main research challenges for the development of software for ubiquitous systems?

3.2.3 Research process

In order to conduct this review, we broadly followed the guidelines by Kitchenham and Charters [Kitchenham and Charters, 2007] and their updated version by Kitchenham and Brereton [Kitchenham and Brereton, 2013]. We first performed a search and selection process to obtain the primary studies that constituted the basis for obtaining our results. The primary studies were passed through a quality assessment step to provide validation of the quality of each study. Next, data related to approaches' characteristics and limitations, future work, and general open issues was extracted from the primary studies. Finally, we synthesized the data to obtain the results of the review.

A Search and selection process

The process followed to search and select primary studies is depicted in Fig. 3.1. We have devised this process inspired by the multi-stage process used by Kitchenham and Brereton, in what can be considered the updated guidelines for conducting SLR's in software engineering [Kitchenham and Brereton, 2013].

First, an informal search step (`prel1` in Fig. 3.1) was performed to ensure having enough papers for the review and enough interest towards the topic, as expressed by the works in the literature. Then, we performed a search process (`I` in Fig. 3.1) divided into three phases: *i*) a hybrid initial setup (`srch1` in Fig. 3.1) based on three different types of pilot searches; *ii*) a search phase divided into manual search (`srch2.1` in Fig. 3.1), automatic search (`srch2.2` in Fig. 3.1), and snowballing (`srch2.3` in Fig. 3.1); and *iii*) a final phase (`srch3` in Fig. 3.1) in which the outcomes from the previous phase were collated. At that point we had collected not less than 5377 papers. These papers were passed to a selection process (`I` in Fig. 3.1) where, we apply inclusion/exclusion criteria (`sel1` in Fig. 3.1) and quality assessment (QA) screening (`sel2` in Fig. 3.1) with which we selected 122 papers. After that, a complementary backward snowballing process (`sel2` in Fig. 3.1) was performed to obtain a comprehensive final set. From this last step we obtained 6 more papers, making a final total of 128 papers selected as primary studies for the obtention of the results of the review.

In Section A.1 we detail how the hybrid initial setup was performed. In Section A.2 we explain how we conducted our main search process. Finally, Section A.3 explains how the results of the search process were analyzed and selected to obtain the the set of primary studies.

A.1 Hybrid initial setup. To setup our main search processes (`srch2.1`, `srch2.2`, and `srch2.3` in Fig. 3.1) we followed a hybrid approach in which we performed three pilot searches independently (one automatic, one manual, and one snowballing), and, from the combination of the outcomes of the three pilots, we determined the initial setup

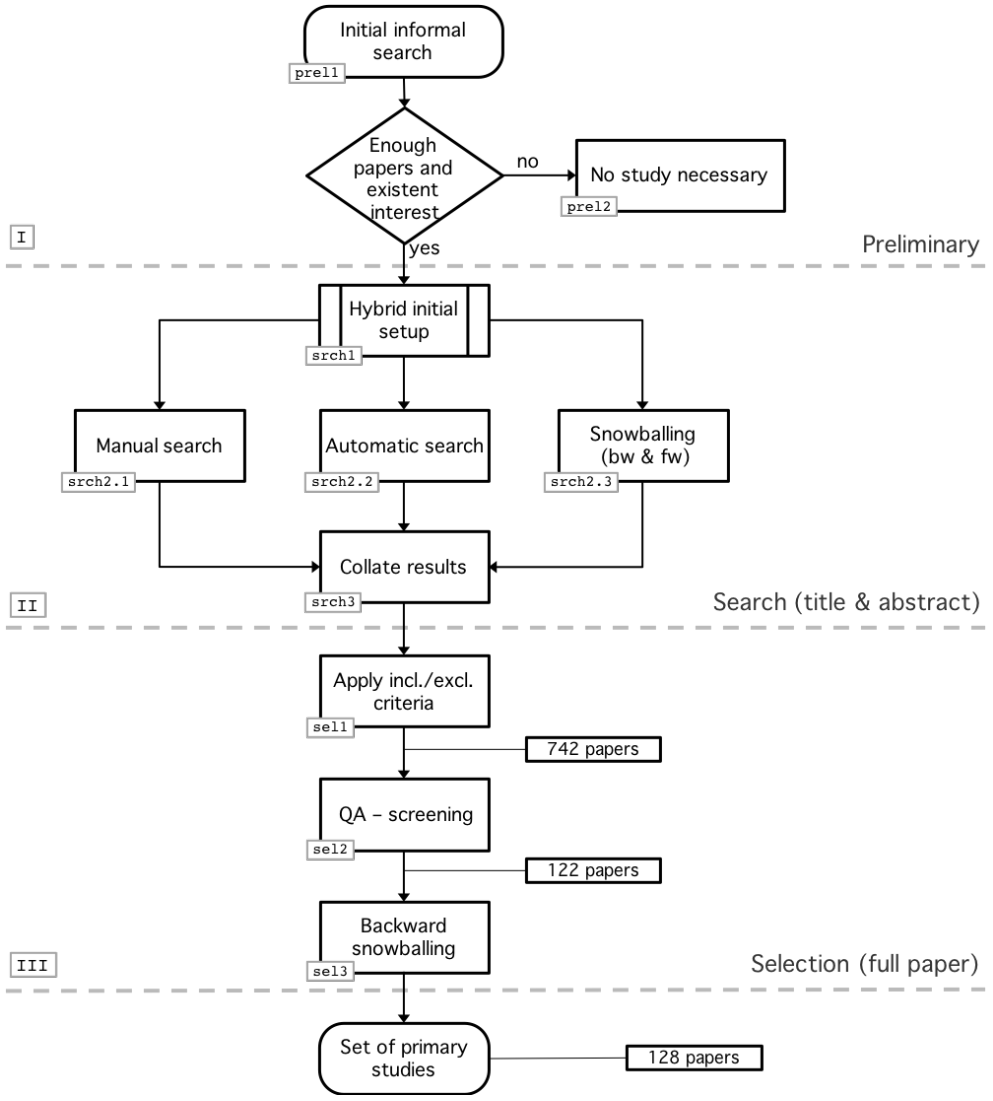


Figure 3.1: Search and selection process (Literature Review)

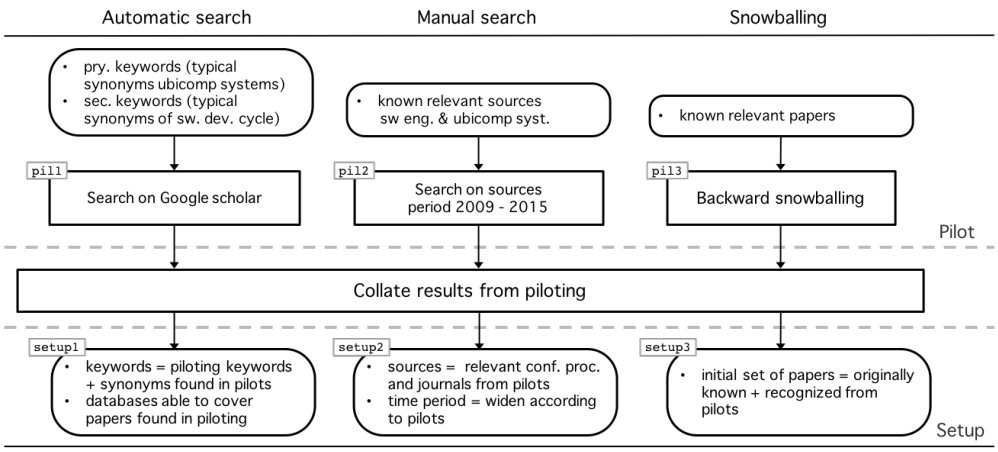


Figure 3.2: Hybrid initial setup (Literature Review)

for each of the main search processes. The goal of our hybrid approach was to try to eliminate, right before start, the possible bias caused by our previous knowledge, which could affect the findings of the search. Thus, instead of deciding on the sources for the search based only on our previous knowledge, we let the literature complement, correct, and enrich our setups by using results from pilot search processes.

Our hybrid initial setup process is summarized in Fig. 3.2. As it can be observed, the process is divided into three high-level phases across the three processes of search. The phases are: pilot searches, collate results from pilots, and setup of main search processes. Next we describe these three stages in detail.

Stage 1 – Pilot. For each type of search process (i.e., manual, automatic, and snowballing) we performed a pilot as follows:

- Pilot of automatic search (`pil1` in Fig. 3.2): This pilot was performed using search strings that combined any of the terms 'ubiquitous systems', 'pervasive system', or 'cyber-physical system', with any of the terms 'software development life cycle' or 'software engineering'. We considered Google Scholar as source, since it has been recognized as a good alternative to avoid bias in favor of a particular publisher [Wohlin, 2014]. In this way, we benefited from a broad view of the whole spectrum of available publications.
- Pilot of manual search (`pil2` in Fig. 3.2): This pilot was performed using as sources the proceedings of 4 conferences and 4 journals. The conference proceedings considered were Ubiquitous Computing (UbiComp), Conference on Pervasive Computing and Communications (PerCom), International Conference on Software Engineering (ICSE), and Foundations of Software Engineering (FSE). The journals used were Transactions on Software Engineering (TSE), Journal of Systems and Software (JSS), Pervasive and Mobile Computing, and Transactions on Software Engineering and Methodology (TOSEM). For these sources the time period considered was years 2010–2015. The main idea was not to cover all sources we knew relevant, but a small number that included the most relevant according to our initial knowledge. The time period was chosen to be short and recent, aiming for a fast process able to deliver relevant results for setting up our main search processes.
- Pilot of snowballing (`pil3` in Fig. 3.2): We formed our start set of papers for this pilot based on papers we knew relevant beforehand, but only from the same relevant sources and period of time we considered for the pilot of manual search. Then, we performed backward snowballing by checking the references of the start set of papers and selecting relevant papers. The decision to spot a relevant paper was based on formal qualitative assessment of title, abstract, and structure of the paper, as well as on a minimalistic inclusion/exclusion criteria that excluded papers not based on empirical research (as seen from its abstract) and papers less than three pages long.

Stage 2 – Collate Results. After the pilot stage, the results were collated to support

the setup of the search processes. This included removing duplicates as well as extracting and organizing for each paper data related to publishing source (i.e., journal or conference proceedings where the paper appeared), year, and publisher (or digital library that contains the paper). Thus, the setup of the search processes was based on this information and on the selected papers from the pilot stage.

Stage 3 – Setup. The setup of each of our main search processes was done based on the collated results from the three pilot searches. Thus, the setup of the main process of one type of search was optimized based on the outcomes of the pilots of the other types of search. Specifically, the initial setup for each of the search processes was optimized in the following way:

- For *automatic search setup* (`setup1` in Fig. 3.2), based on the collated results from the pilots, we were able to target specific digital libraries aiming at covering the papers found in the pilot process. These are: ACM DL, IEEE XPLORE, Wiley Online Library, Science Direct, and Springer. We were also able to optimize the search strings aiming at covering larger amount of papers more closely related to our research goals. On the one hand, we included more terms related to ubiquitous systems such as 'home automation' and 'smart building'. On the other hand, we removed the term 'software development life cycle' after noticing it was not useful, and we kept the term 'software engineering', considering, in addition, the possibility of finding each of their elementary terms, i.e., 'software' and 'engineering', as separate terms.
- For the *manual search setup* (`setup2` in Fig. 3.2), we benefited from the collated results of the piloting stage by broadening the time period and sources considered as to cover more papers that could be considered relevant.
- The *snowballing setup* (`setup3` in Fig. 3.2) was done based on the guidelines by Wohlin [Wohlin, 2014], using as tentative start set of papers being those from the collated results of the pilot stage. Then, as suggested by Wohlin's guidelines, we looked for a definite start set of papers featuring diversity and relevancy. To this end, we selected papers only from publishing sources and years that were relevant for the manual search setup, we established a maximum number of papers to be considered per publishing source, and used the number of citations of the papers to rank them and decide on which ones to keep in case too many of a single source were found.

A.2 Search process. The search phase of our review (`II` in Fig. 3.1) was composed of three different search processes: one based on automatic search on digital libraries (`srch2.1` in Fig. 3.1), one based on manual search on the most relevant journal and conference proceedings in the topic (`srch2.2` in in Fig. 3.1), and one based on snowballing (`srch2.3` in Fig. 3.1), both backward and forward [Wohlin, 2014].

We considered articles published between 1999 and 2015. The initial date was decided based on the publication year of the first relevant article about the development of

ubiquitous systems appearing in the proceedings of a top-tier conference in software engineering (specifically, The International Conference on Software Engineering, ICSE). During this stage only title and abstract were considered for each of the collected papers.

Automatic search (`srch1` in Fig. 3.1)**.** This search process was performed using predefined search strings over specific electronic databases.

- Source selection: Following the updated guidelines for performing systematic reviews proposed by Kirchenham and Brereton [Kitchenham and Brereton, 2013] we selected IEEE XPLORE, ACM DL, and SCOPUS as electronic databases for our automatic search. This accounts for the two major digital libraries and an indexing system (i.e., SCOPUS), which, as it has been pointed out by Chen et al. [Chen et al.,] and Dybå et al. [Dyba et al., 2007], reduces the need for searching some publishers' sites such as Springer Link and Wiley Interscience.
- Search string: Our search strings were combinations between primary search terms and secondary search terms, in both cases considering the most relevant synonyms that could be recognized. Therefore we have:

  $$search\,string = primary\,term\ \&\ secondary\,term$$

  where *primary term = {ubiquitous system, pervasive system, cyber-physical system, ambient intelligence, smart building, smart home, home automation}*, and *secondary term = {software, engineering, software engineering}*.

Manual search (`srch2` in Fig. 3.1)**.** This search process was conducted respecting the time period we established for all search processes (i.e., 1999–2015) over the most relevant conference proceedings and journals on ubiquitous systems and software engineering, according to our previous experience and the results obtained during the setup process. The full list of sources is presented in Table 3.1. The list of sources was expanded with respect to the one used during the pilot of manual search in the setup process described in Section A.1.

Snowballing (`srch3` in Fig. 3.1)**.** Our process of snowballing is inspired by the guidelines by Wohlin [Wohlin, 2014]. The starting set of papers necessary to begin the process was formed during the setup process as described in Section A.1. The overall process is detailed in Figure 3.3, and we describe next how we performed backward and forward snowballing.

The steps we followed in our *backward snowballing* process were:

1. Go through the reference list of the current paper α and exclude papers that do not fulfill the following basic criteria: publication year between 1999–2015, language of the paper English, and type of publication peer reviewed papers.
2. Remove papers that have been already examined because found earlier through either backward or forward snowballing in this or previous iteration

Table 3.1: Manual search sources

Type	Abbreviation	Name	Publisher
conference	ICSE	International Conference on Software Engineering	IEEE/ACM
	FSE	Foundations of Software Engineering	ACM
	UbiComp	Conference on Ubiquitous Computing	ACM
	PerCom	International Conference on Pervasive Computing and Communications	IEEE
	Mobiquitous	International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services	ACM
	ICCPS	International Conference on Cyber-Physical Systems	IEEE/ACM
journal	TSE	Transactions on Software Engineering	IEEE
	TOSEM	Transactions on Software Engineering and Methodology	ACM
	–	Software: Practice and Experience	Wiley
	IST	Information and Software Technology	Elsevier
	JSS	Journal of Systems and Software	Elsevier
	–	Pervasive Computing	IEEE
	–	Pervasive and Mobile Computing	Elsevier
	–	Personal and Ubiquitous Computing	Springer
magazine	–	Software	IEEE
	–	Computer	IEEE

3. For each of the candidate papers, look for the place where it is being referenced in α to evaluate if it is relevant for our purpose based on what it is said about it.
4. For the remaining candidate papers find each paper and browse through and read its most relevant parts to evaluate its relevancy for our review (i.e., if it is about empirical research and if it is about any part of the cycle of software development for ubiquitous systems)

For our *forward snowballing* process we went through the following steps:

1. Identify candidate papers based on those citing paper α being examined (using Google Scholar)
2. From the candidate papers exclude any that does not fulfill the basic criteria used in step (1) of *backward snowballing*
3. Same as step (2) of *backward snowballing*
4. For each of the remaining candidate papers, look for the place in such paper where α is being cited to evaluate how relevant is the link to α and decide to keep or remove the candidate.

input: Starting set of papers S_0 built during hybrid setup
Define set S of selected papers ;
Define set E of papers for examination ;
Define set S_c of papers selected in current iteration ;
Define set R of excluded (or rejected) papers ;
$E \leftarrow S_0$;
while *E is not empty* **do**
  foreach *paper in E* **do**
    /* avoid duplicates from S, S_c, and R */
    backward snowballing and put the result in S_c ;
    forward snowballing and put the results in S_c ;
  end
  $R \leftarrow R \cup (E - S_c)$;
  $E \leftarrow S_c$;
  $S \leftarrow S \cup S_c$;
end
return S

Figure 3.3: Snowballing process

5. Same as step (4) in *backward snowballing*

A.3 Study selection strategy. Our selection strategy consisted of three steps as shown in Fig. 3.1: apply inclusion/exclusion criteria (`sel1` in Fig. 3.1), apply quality assessment screening (`sel2` in Fig. 3.1) to guarantee minimum quality on the studies considered, and a complementary backward snowballing process (`sel3` in Fig. 3.1) to try to ensure that no relevant paper was missed. Next, we detail each of these steps.

Step 1 – Study selection based on incl./excl. criteria (`sel1` in Fig. 3.1)**.** We selected papers for inclusion if they satisfy all of the following: they were written in English, they were primary studies and not secondary or tertiary studies(i.e., not reviews), not shorter than four pages, they presented empirical research, and they were published in any of the sources selected for the manual search process. We excluded papers when we found that the main focus of the paper was not on developing software for ubiquitous systems, and if the publishing source could not be considered relevant for the purpose of our review.

Step 2 – QA screening (`sel2` in Fig. 3.1)**.** We apply a quality assessment screening step similarly to what Dybå and Dingsøyr proposed in [Dybå and Dingsøyr, 2008] to ensure minimum quality on the selected studies. The screening consisted on applying the first three items of the quality criteria in Table 3.2 to each of the papers selected after having applied the inclusion/exclusion criteria. To pass the screening a paper had to be assessed positively on criterion (1) and on either criterion (2) or (3).

Step 3 – Backward snowballing with screening (`sel3` in Fig. 3.1)**.** This process was followed as a complementary step for the entire search and selection process. The

Table 3.2: Quality criteria

1. Is the study reported empirical research?
2. Were the aims and objectives clearly reported (including a rationale for why the study was undertaken)?
3. Was there an adequate description of the context in which the research was carried out?
4. Is the presented approach clearly explained?
5. Are threats to validity taken into consideration?
6. Are the findings of the research clearly stated?

intention of this step was to find any relevant paper that had not been considered based on papers already assessed as relevant. The backward snowballing in this case was performed as described in Section A.2, however complemented with our inclusion/exclusion criteria and *QA-screening*. That is, the papers selected in this case had to satisfy the criteria from the backward snowballing and the ones from inclusion/exclusion and *QA-screening*. After this step, 6 more papers were included into our set of selected primary studies.

B Quality assessment

Once the set of selected papers was formed, we performed a quality assessment of the empirical study presented in each paper. Note that the existence of an empirical study or evaluation was guaranteed as all selected papers passed through the QA-screening (`sel2` in Fig. 3.2). We assessed the quality of each study by grading it according to 6 criteria, where the score was obtained by counting each criterion fulfilled equally. Our quality criteria (Table 3.2) was based on three criteria used by Dybå and Dingsøyr in [Dybå and Dingsøyr, 2008] (1,2,3 in Table 3.2) and three criteria used by Unterkalmsteiner et al. in [Unterkalmsteiner et al., 2012] (4,5,6 in Table 3.2).

The results of the quality assessment are shown in Table 3.3, where we have grouped the selected papers according to the QA score they obtained. Overall the average QA score is considerably good, with most of the papers getting 4 or more points.

C Data extraction

Data was extracted from each of the 128 selected papers using an extraction form that considers data about the approach being presented in the paper and about the study that is used to evaluate it. The items of the extraction form are depicted in Table 3.4.

The items for data extraction (Table 3.4) concerning study characteristics were inspired by those used by Dybå and Dingsøyr [Dybå and Dingsøyr, 2008]. Bibliographic reference refers to author, year, title, and publishing source. Design of study describes if the study is a case study or a experiment, and the rest of the items refer to information typically expected from an empirical study.

On the other hand, the items about approach characteristics were directly aimed at

Table 3.3: Quality assessment

QA score	Papers	Number
6	P4, P13, P19, P21, P22, P25, P36, P80, P83, P117, P126	11
5	P1, P2, P5, P6, P7, P8, P9, P11, P15, P17, P20, P23, P26, P27, P30, P31, P32, P33, P34, P37, P41, P44, P47, P48, P49, P50, P54, P56, P57, P59, P62, P64, P67, P73, P79, P81, P84, P85, P86, P87, P89, P90, P92, P94, P95, P96, P102, P104, P105, P106, P109, P113, P114, P115, P116, P119, P120, P121, P122, P124, P127, P128	62
4	P3, P10, P12, P14, P16, P18, P24, P38, P39, P45, P46, P52, P55, P60, P61, P65, P66, P74, P82, P91, P93, P97, P98, P99, P100, P103, P107, P108, P110, P118, P125	31
3	P28, P29, P35, P40, P43, P51, P53, P58, P63, P68, P69, P70, P71, P72, P75, P76, P77, P78, P112, P123	20
2	P101, P111	2
	Avg. QA score = 4.48	

answering the research questions of the review. The item called phase in the development cycle together with year and publishing source from bibliographic reference were meant to describe how much attention each of the phases have received throughout time, thus answering ch3-res-Q1. During the extraction of the data, we had several approaches classified in more than one phase and in some cases we marked the classification as "unsure", since we could not reach a definite decision. All such cases were settled during the synthesis process, as we describe later.

Three of the items of the data extraction form were aimed at answering ch3-res-Q2. These are: approach identifier, approach aims, and main focus or concern. The rationale was to obtain a characterization for each of the approaches found within the selected papers that allows to understand its importance and fit within its corresponding phase and in relation to the whole development of software for ubiquitous systems.

To answer the limitations for ch3-res-Q2 we gathered, from each selected paper, any problem or limitation of the proposed approach, as recognized by the authors of the corresponding paper.

D Synthesis of findings

Our approach for synthesizing findings is based on the synthesis method 'thematic analysis/synthesis' [Braun and Clarke, 2006], with the difference that instead of identifying themes emanating from the findings reported in each selected paper, we consider the focus or concern that the approach in question is meant to address.

In answering ch3-res-Q1 we aimed to find out what phases of the development cycle have received attention in the literature and, also, how much attention each phase has received over time. For the first part, the synthesis was straightforward as we only had to put together a list of the phases found during data extraction. For the second part,

Table 3.4: Data extraction form

General concern	Specific data
Study characteristics	paper identifier
	bibliographic reference
	study goals
	design of study
	data collection
	data analysis
	setting of study
	limitations and threats to validity
Approach characteristics	approach identifier
	approach aims
	main focus or concern
	phase in the development cycle
	limitations
	future work
	related approaches

however, we required to do more since, as we described in Section 3.3, the classification of some approaches was marked as "unsure". To this end, we used data originally targeted to answer ch3-res-Q2 related to approach's aims and focus to help us reach a final decision on the phase(s) to which each approach was to be classified. The data collected for answering ch3-res-Q2 was synthesized in such a way that the focuses extracted directly from the primary studies were combined to obtain a set of focuses that contained no repetitions and where similar or closely related focuses from the primary studies were placed together under a single focus that summarizes them. In the end, we obtained a characterization of the main focuses or concerns that have been the target in the research of each of the phases of the development cycle.

3.3 Results

We identified 132 approaches, which we first classified according to the phase of the development cycle they are targeting. This classification allows to answer to ch3-res-Q1 in Section 3.3.1. Some approaches were classified in more than one phase. Therefore, the total number of approaches reaches 134 when summing approaches of each phase. After the first step of classification, we categorized the approaches in each phase based on their focus, which may correspond to type, activity, or concern, depending on the particular phase. These results together with the limitations that were recognized by the authors of each approach (i.e., the answer to *ch3-res-Q2*) are presented in Section 3.3.2.

3.3.1 Phases of the engineering process (answer to *ch3-res-Q1*)

We recognized a total of 8 phases as those that have received attention in the literature. These are: requirements, design, implementation, verification and validation (v&v), testing, deployment, evolution/maintenance, and feedback. Most approaches have been directed to address one or more of these phases, while only few of them have claimed to address the entire development cycle (as defined by their authors).

Except for feedback, all phases that we have identified reflect what is recognized as a software development life cycle, both in general and in the context of ubiquitous systems [Cassou et al., 2012]. Concerning feedback, our results show that it has received considerable attention in the literature, placing it as a phase in its own right, which reflects its undoubted importance in the development of software for ubiquitous systems [Brun et al., 2009].

Table 3.5 shows the number of approaches we found per phase (counting the whole cycle as one phase) and the percentage each represents with respect to the total number of approaches. We can see that implementation, evolution/maintenance, and feedback have received most of the attention in the literature. This may be due to the natural dynamicity of ubiquitous systems, in which the feedback is essential to adapt the system, and the evolution and implementation are challenging the usual development and release methods.

Figure 3.4 shows how much attention the phases of the development cycle have received over the time span from 2003 to 2015, period of time during which the approaches considered in this review were published. We can see that four of the phases (i.e., evolution/maintenance, feedback, testing, and implementation) received attention throughout the whole time span. On the other hand, three of the phases were addressed constantly until the end of the time span but not from the beginning, with v&v starting in 2005, design in 2007, and requirements in 2009. Of the rest, deployment received continuous attention except for the period between 2005 and 2006, and the whole development cycle received attention starting in 2005 and until 2012, except from 2007 to 2008.

3.3.2 Approaches and limitations (answer to *ch3-res-Q2*)

In this section we present the approaches we have found directed to address the phases of the development cycle that we previously listed in Section 3.3.1. Our results here are aimed at providing a characterization of the approaches and their limitations, thus answering ubiq-Q2 and ubiq-Q3. In all cases the information we present is based only on the claims made by the authors of the paper where each approach was found. Therefore, any information omitted by the authors of the papers considered is reflected in our results. This is particularly noticeable in the case of approaches' limitations, since, as we will see, many of the approaches reported no limitations.

A Whole cycle

We found 5 approaches claiming to address some aspect of the the development cycle seen as a whole. Table 3.6 shows each of these approaches classified by their focus, providing the percentage that each focus accounts within the phase. We identified three

Table 3.5: Approaches per phase

Phase	Focus	No. of approaches	Percent
Whole cycle	framework; middleware; tools	5	4
Requirements	analysis; specification; modeling; system modeling; model transformation	7	5
Design	modeling; prog./design language; error detection; end-user programming	10	8
Feedback	information sharing; user feedback; spatial context; context aggregation; context inconsistency; context representation; data management; schedule management; process selection	22	16
Implementation	system reconfiguration; communications; message broker; error recovery; code generation; service management; coordination; implementation management; interactions; integration; distribution; data management	37	28
V&V	runtime verification; model checking; fault detection	6	4
Testing	simulator; context-aware testing; test adequacy	10	8
Deployment	dynamic deploy; end-user deploy; multi-platform deploy	7	5
Evol./Maint.	monitoring; dynamic configuration; adaptation; consistency management; self-maintenance; end-user maintenance	30	22
Total		134	100

approaches focused on proposing a framework (A15, A25, A121), while, of the other two, one proposes a tool (A2), and the other a middleware (A78).

A.1 Approaches. Of the approaches proposing a framework, approach A15 (called MUSIC) combines a modeling language modeling language, middleware components, and supporting tools to enable automation of the adaptation of the software to the changing needs and operating conditions at runtime. Approach A25 consists of an integrated architecture-driven framework for modelling, analysis, implementation, deployment, and run-time migration of software systems executing on distributed, mobile, heterogeneous computing platforms. Approach A121 is a conceptual framework meant to support the characterization of ubiquitous software projects according to their ubiquity adherence level.

Concerning approaches that are around tools or middleware, DiaSuite (A2) is a suite

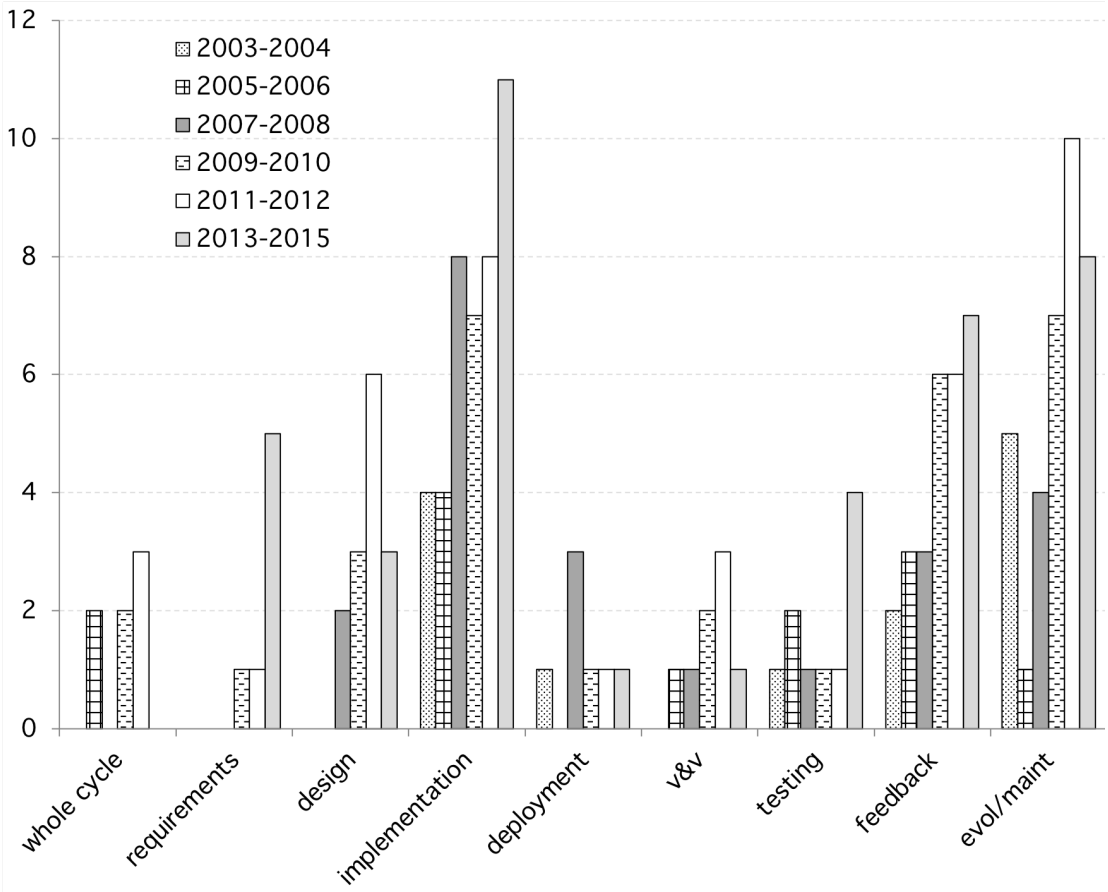


Figure 3.4: Number of approaches in primary studies per phase

of tools that provides support design, implementation, testing, deployment, and evolution, which according to its authors are the phases of the development of a pervasive computing applications. On the other hand, approach A78 is about a dynamic reconfigurable situation-aware middleware that simplifies the development and facilitates the execution and reconfiguration of trustworthy ubicomp application software with situation-awareness and security requirements.

A.2 Limitations. Only two of approaches of targeting the whole development cycle reported limitations, while the other three did not recognize any limitation. These are presented in Table 3.7.

B Requirements engineering

We identified 7 approaches targeting the requirements phase, where the focus of each approach was related to one particular activity of requirements engineering [Aurum and Wohlin, 2003]. In general, the distribution of approaches per activity was considerably even (see Table 3.8), with analysis and modeling being the focus of two approaches each, and specification, system modeling, and model transformation, being each the focus of one approach.

Table 3.6: Approaches for the whole development life cycle

Focus	Approaches	Percent
Framework	A15 [Hallsteinsen et al., 2012], A25 [Malek et al., 2010], A121 [Spínola and Travassos, 2012]	60
Tools	A2 [Cassou et al., 2012]	20
Middleware	A78 [Yau et al., 2006]	20

Table 3.7: Limitations of approaches for the whole cycle

Approach	Limitation
A2	◇ Lack of support for automatically generated programming
A15	◇ Provides only limited basic support for security, which should be complemented with more features
Approaches not reporting limitations: A25, A121, and A78.	

B.1 Approaches. We found two approaches focused on requirements analysis: approach A16 which is a problem-oriented approach that enables software engineers to represent and reason about changes in the physical environment of software systems and assess their impact in requirements satisfaction, and approach A105 which is a goal-based framework for context consistency analysis and conflict analysis. Concerning requirements modeling, we found two approaches as well, where approach A106 is a goal-based framework for modeling and reasoning on systems operating in and reflecting on varying contexts, and approach A132 (called REUBI) is a goal-based requirements engineering method for representing the influence of context and adverse situations.

For each the activities of requirements specification, system modeling, and model transformation we identified one approach. Approach A122 provides mechanisms to structure and represent mobile privacy requirements in a way that they can be under-

Table 3.8: Approaches for Requirements phase

Focus	Approaches	Percent
Analysis	A16 [Salifu et al., 2012], A105 [Ali et al., 2013]	29
Modeling	A106 [Ali et al., 2010], A132 [Ruiz-López et al., 2013]	29
Specification	A122 [Thomas et al., 2014]	14
System modeling	A124 [Dalpiaz et al., 2013]	14
Model transformation	A130 [Moros et al., 2013]	14

Table 3.9: Limitations of approaches for requirements phase

Approach	Limitation
A106	◇ Issues on scalability of the automated support tool
	◇ Issues on the amount of effort an analyst is required to pay to construct the proposed models and use the analysis techniques
A122	◇ Scalability: To apply the privacy requirements distillation proposed by the approach, software engineers need to be familiar with qualitative data analysis techniques
	◇ Reliability: The thematic coding in the distillation approach is subjective and depends on the software engineer's interpretation and therefore can be biased
	◇ Generalizability: Distillation critically relies on emotions, and negative behavior patterns within the qualitative data, to analyse privacy requirements, which makes it difficult to apply to other datasets
A124	◇ Customization: The approach does not consider how different human agents are in terms of skills and preferences
	◇ User interfaces: the approach does not pay attention to how users interact with technical systems
Approaches not reporting limitations: A16, A105, A130, and A132.	

stood and implemented by software engineers and designers. Approach A124 proposes an architecture for adaptive socio-technical systems, to collect data about changes in the operational environment and diagnose failures by checking monitored data against requirements models. Finally, approach A130 generates, from the requirements specification, an initial application model according to a domain-specific language, using model transformation rules, while simultaneously registering the traces between the requirements and the domain concepts generated.

B.2 Limitations. For the requirements phase three approaches were recognized to have limitations, whereas 4 approaches did not report any limitation. Table 3.9 summarizes these limitations.

C Design

We found 10 approaches around the design phase. From these we have identified 4 different focuses: modeling and programming/design language (4 approaches), error detection (1 approach), and end-user programming (1 approach). This categorization is shown in Table 3.10.

C.1 Approaches. Concerning modeling, approach A5 proposes interaction contracts that allow the architect to precisely specify the interactions between components, ap-

Table 3.10: Approaches for Design phase

Focus	Approaches	Percent
Modeling	A5 [Cassou et al., 2011], A50 [Serral et al., 2010], A53 [Achilleos et al., 2010], A72 [Lézoray et al., 2011]	40
Prog/design language	A1 [Cassou et al., 2012], A28 [Kulkarni et al., 2012], A68 [Weis et al., 2007], A98 [Salvaneschi et al., 2012]	40
Error detection	A81 [Wang et al., 2013]	10
End-user programming	A73 [Guo et al., 2011]	10

proach A50 (called PervML) is a modeling language that allows to represent context-aware pervasive systems at a high level of abstraction, approach A53 is a model-driven methodology to support the overall creation of pervasive services, and approach A72 is an adaptive medium approach to specify an adaptive and dynamic architecture for pervasive ambient assistive living systems. A programming or design language is the focus of 4 approaches: approach A28 is a domain-specific design model for programming context-aware collaborative applications, approach A1 (called DiaSpec) is a design language for describing both a taxonomy of area-specific entities and pervasive computing application architectures, approach A68 (called VisualRDK) is a high-level programming language for prototyping pervasive applications, and approach A98 (called CONTEXTERLANG) is aimed at implementing context-aware software in a highly concurrent and distributed setting.

The other two approaches we found for the design phase are around error detection and end-user programming, respectively. Approach A81 (called L1Simplex) proposes a safety monitoring system to detect physical failures and examine the scale of the uncertainty in the system due to failures, and approach A73 (called OPEN) proposes an ontology-based programming framework which allows a broad category of users' participation and cooperation in the development of context-aware applications.

C.2 Limitations. For the design phase 5 approaches were recognized to have limitations, while the rest (the other 5 approaches) did not report any limitation. Table 3.11 summarizes these limitations.

D Feedback

A total of 22 approaches were identified for the feedback phase. Among these approaches we found 9 different focuses. The relation between focus and approaches is shown in Table 3.12.

D.1 Approaches. We identified 5 approaches related to context inconsistency: approach A6 (called CINA) is a constraint instability analysis tool that systematically

Table 3.11: Limitations of approaches for the design phase

Approach	Limitation
A1	◇ Lack of non-functional layers
A50	◇ Limited due to its inability to infer and make context-aware adaptation decisions automatically
A73	◇ User-friendliness: It is based on a text-based configuration interface as opposed to a graphical one.
A81	◇ Inability to adjust dynamically the stability envelope
	◇ Not considering timing issues given real-time systems
A98	◇ Overhead due to context management
Approaches not reporting limitations: A5, A28, A53, A68 and A72.	

suppresses the detection of unstable context inconsistencies, approach A37 proposes a consistency management framework to capture inconsistent contexts based on a formal semantic matching and inconsistency-triggering model, approach A43 is aimed at resolving context inconsistencies by identifying problematic contexts based on information available from the context-aware application, approach A104 entails a side effect measurement framework that measures the significant side effect caused by context inconsistency resolution on ubiquitous applications, and approach A105 is a goal-based contextual framework to detect inconsistencies between contexts specified in a goal model. Context representation was the target of three approaches: approach A4 (called TOTA) proposes distributed tuples spread over a network to enable representing contextual information in an expressive way suitable for easy gathering, approach A18 (called TOTAM) allows the developer to determine upper boundaries on the availability of the injected tuples in the system independent of the connectivity, and approach A56 (called CTL3) proposes a 3-valued Computation Tree Logic to enable formal specification of temporal contextual properties in asynchronous pervasive computing environments. We found three approaches focused on data management: approach A58 proposes a system to provide functional support in data management for context-aware application development, approach A85 is an approach that seeks to provide a desirable degree of fidelity, and A87 describes a smart sensor query-processing architecture using database technology that allows users to specify the data they want to collect.

Two approaches have focused on information sharing: approach A26 proposes a hybrid model of context-aware service provisioning to access and share information about services and content available in our daily environment, and approach A95 is a content and service composition framework for distributing and presenting composite multimedia content to users on different devices. Feedback from users was found to be the focus of two approaches: approach A125 (called iPlumber) proposes a user-oriented management system that provides a collaborative environment allowing users to share and reuse the context-aware applications they have created as well as their management experiences, and approach A55 is a context-aware approach based on user behavior that

Table 3.12: Approaches for Feedback phase

Focus	Approaches	Percent
Context inconsistency	A6 [Xu et al., 2015], A37 [Xu and Cheung, 2005], A43 [Chen et al., 2011], A104 [Xu et al., 2012b], A105 [Ali et al., 2013]	23
Context representation	A4 [Mamei and Zambonelli, 2009], A18 [Boix et al., 2014], A56 [Wei et al., 2012]	14
Data management	A58 [Xue et al., 2013], A85 [Payton et al., 2012], A87 [Gehrke and Madden, 2004]	14
Information sharing	A26 [Riva and Toivonen, 2007], A95 [Nahrstedt et al., 2005]	9
User feedback	A55 [Pallapa et al., 2014], A125 [Guo et al., 2010]	9
Spatial context	A32 [Murukannaiah and Singh, 2015], A94 [Holzmann and Ferscha, 2010]	9
Context aggregation	A90 [Meier et al., 2009], A97 [Malandrino et al., 2010]	9
Process selection	A107 [Füller et al., 2012], A103 [Chan and Chuang, 2003]	9
Schedule management	A63 [Driver and Clarke, 2008]	4

exploits the history of interactions between users and the environment to obtain the amount of privacy associated with specific requests. Likewise, two approaches were found focusing on spatial context: approach A32 (called Platys) is a framework for place-aware application development able to reason about places from sensor data, and approach A94 presents a software framework designed to run on embedded systems and facilitate the development of spatially aware applications.

Context aggregation has been the focus of two approaches: approach A90 (called iTransIT) is a spatial programming framework aimed at providing a standardized way to build context-aware global smart space applications using information distributed across independent systems and related services, and approach A97 (called MIMOSA) is a distributed framework that seeks to provide an effective and efficient solution for the adaptation of Internet services on the basis of a comprehensive notion of context. We found two approaches focused on process selection: approach A107 is a context driven approach that selects processes suitable for a system's context and integrates the system into the chosen processes, while approach A108 proposes a universal infrastructure to complement context aware techniques by adding a single initial selection phase in which processes are chosen according to their contexts. Finally, we identified one approach for schedule management, namely, approach A 63 which is an application framework that provide structure and behavior to support context-based activity schedule composition.

Table 3.13: Limitations of approaches for the feedback phase

Approach	Limitation
A4	◇ Lack of mechanisms and policies to compose tuples with each other, so as to enable the expression of unified distributed data structures from multiple sources
A56	◇ Issues on scalability, specifically in terms of efficiency of the checking algorithm.
A58	◇ Limited expressiveness on the attribute-value context model and the SQL query language
Approaches not reporting limitations: A6, A18, A26, A32, A37, A43, A55, A63 A85, A87, A90, A94, A95, A97, A103, A104, A105, A107, and A125.	

D.2 Limitations. Compared to the large proportion of approaches that the feedback phase has, only a very small number of approaches were spotted for limitations by their authors. These are summarized in Table 3.13.

E Implementation

We identified 37 approaches for the implementation phase. From these we recognized 12 different concerns, with most of the approaches focusing on six of them as follows: integration (6), system configuration (5), communications (4), code generation (4), service management (4), and interactions (4). The complete relation between concerns and approaches is shown in Table 3.14.

E.1 Approaches. We found 6 approaches focused on integration: approach A45 (called Plan B) is a constraint-based file system import mechanism that allows the system to adapt to changes in the environment and permits users to customize the environment, approach A61 is a middleware platform for application assembly which provides a target for applications whose structure is dynamically specified and reconfigured by runtime processes, approach A75 proposes an architectural framework called 'breadboard architecture' that enables seamless addition and removal of different components, approach A79 is a middleware solution called DigiHome that enables the integration of heterogeneous computational entities by relying on the service component architecture, approach A88 (called UCM or Ubicomp Common Model) is a meta middleware that enables the integration of existent ubiquitous computing (ubicomp) systems, and approach A93 is a method that allows to smoothly integrate highly heterogeneous devices into an ambient ecology. System configuration was the main concern of 5 approaches: approach A7 (called SATIN) is a lightweight component metamodel instantiated as a middleware to reconfigure the software system by dynamically transferring code, approach A36 (called Prism-MW) is middleware platform that provides native implementation-level support for arbitrary architectural styles, allowing software developers to transfer directly architectural decisions into implementations, approach A44 (called ACoMS) is a model-based,

Table 3.14: Approaches for Implementation phase

Focus	Approaches	Percent
Integration	A45 [Ballesteros et al., 2006], A61 [Pham et al., 2009], A75 [Soldatos et al., 2007], A79 [Romero et al., 2013], A88 [Blackstock et al., 2008], A93 [Rashid et al., 2012]	16
System reconfiguration	A7 [Zachariadis et al., 2006], A36 [Malek et al., 2005], A44 [Hu et al., 2008], A74 [Paspallis and Papadopoulos, 2014], A92 [Schuhmann et al., 2010]	14
Communications	A32 [Murukannaiah and Singh, 2015], A33 [Meier and Cahil, 2010], A48 [Aitenbichler et al., 2007], A57 [Arnaboldi et al., 2014]	11
Code generation	A3 [Cassou et al., 2012], A28 [Kulkarni et al., 2012], A50 [Serral et al., 2010], A109 [Harrington and Cahill, 2011]	11
Service management	A31 [Ballesteros et al., 2012], A38 [Morris et al., 2015], A64 [Paluska et al., 2008], A110 [Robinson et al., 2008]	11
Interactions	A5 [Cassou et al., 2011], A8 [Julien and Roman, 2006], A54 [Pallapa et al., 2014], A57 [Arnaboldi et al., 2014]	11
Message broker	A19 [Kiani et al., 2013], A42 [Becker et al., 2003]	5
Coordination	A41 [Mamei and Zambonelli, 2004], A82 [Castelli et al., 2015]	5
Distribution	A70 [Gu et al., 2004], A118 [Füller et al., 2012]	5
Data management	A96 [Hess and Campbell, 2003], A101 [Roussaki et al., 2010]	5
Error recovery	A43 [Chen et al., 2011]	3
Impl. management	A64 [Paluska et al., 2008]	3

scalable, and autonomic context management system that provides self-configuration of context sources, approach A74 is a component-based middleware architecture that allows easy configuration according to the needs of the deployed applications and the capabilities of the deployment platform, and approach A92 is a hybrid configuration method to efficiently exploiting the available computation resources in heterogeneous environments. We found 4 approaches focused on communications: approach A32 (called Platys) is a framework for place-aware application development that provides the necessary communication channels between the users of a location-aware mobile application and its developers, approach A33 suggests techniques that can be used by event-based middleware to support collaboration in location-aware mobile applications. approach A48 (called MundoCore) entails a communication middleware that provides harmonization of all the important communication abstractions needed in advanced distributed smart environments, and approach A57 (called CAMEO) is a light-weight context-aware

middleware platform designed to allow mobile social networks to generate and share content with peer-to-peer communications based on user and device characteristics.

We identified code generation as the focus of 4 approaches: approach A3 is a code generator called DiaGen that automatically generates a Java programming framework from both a taxonomy definition and an architecture description, approach A28 is a domain-specific programming framework aimed at creating the execution environments of context-aware applications, approach A50 is a model-driven development implementation framework to provide support to translation from PervML models into code at runtime, and approach A109 is a model-driven approach that seeks to address the task of generating code to implement planning and optimization algorithms in pervasive computing domains. Service management was found to be the main concern of 4 approaches: approach A31 (called Upperware) provides new services directly to the OS to manage heterogeneous software and hardware, approach A38 is a method that allows for greater flexibility in service delivery while reducing the complexity of application development for the user, approach A64 is meant to automate high-level implementation decisions in a pervasive application, and approach A110 is a service composition system called Scooby that provides an effective method for combining services to meet the needs of users in a variety of pervasive computing scenarios. Having interactions as their main concern we found 4 approaches: approach A5 proposes interaction contracts that express in high-level terms what interactions a given component can perform, approach A8 (called EgoSpaces) entails a middleware targeted to allow an individual application to limit the portion of the context with which it interacts, approach A54 is a context-aware hybrid approach aimed at reducing the interactions between the user and the system to improve user experience, approach A57 (called CAMEO) is light-weight context-aware middleware platform aimed at allowing mobile social networks to generate and share content with peer-to-peer communications based on user and device characteristics.

We found two approaches presenting message brokers: approach A19 is a federated broker based context provisioning system aimed at providing asynchronous communication interfaces to clients and neighbor brokers for exchanging contextual and administrative information related to the registered clients and their capabilities, and approach A42 (called BASE) is a micro-broker-based middleware that allows uniform access to device capabilities and services through proxies and the integration of different interoperability protocols. Likewise, we found two approaches focused on coordination: approach A41 (called TOTA) provides agents with effective contextual information that can facilitate the contextual activities of application agents for the definition of complex distributed coordination patterns, and approach A82 (called SAPERE) is aimed at facilitating the decentralized and situated execution of self-organizing and self-adaptive pervasive computing services. We also found two approaches for distribution: approach A70 is a context recognition network toolbox that enables fast implementation of activity and context recognition systems with mechanisms for distributed processing and support for mobile and wearable devices, and approach A118 is an automatic application partitioning for allowing users to describe the location of system resources.

We identified two approaches for data management: approach A96 is a context-aware

Table 3.15: Limitations of approaches for the implementation phase

Approach	Limitation
A3	◇ Impossibility to define logical expressions across attributes (e.g., it is not possible for a query to specify that a device must have a particular value for an attribute or another value for another attribute)
A5	◇ Lack of non-functional layers and of automatically generated support
A8	◇ Available data depends on the connectivity which cannot always be guaranteed at any time
A28	◇ The generative approach does not permit modifications and extensions of an application's execution environment that has already been created
A31	◇ The approach assumes permanent availability of network connections, which is not always possible
A36	◇ The approach needs to store and check abstract class references even when they are not used in a given PRISM-MW class implementation
A38	◇ The approach does not offer any mechanism to select among various components which achieve the same end, but instead, relies on the presence of a specific component which undertakes predefined tasks
A41	◇ Lack of an underlying methodology, enabling engineers to map a specific coordination policy into the corresponding definition of tuples and of their shape
A50	◇ Lacks tools to allow users to reconfigure the system
A70	◇ The approach introduces some processing overhead
A118	◇ The automatic partitioning proposed does not provide automatic parallelization
Approaches not reporting limitations: A7, A19, A32, A33, A42, A43, A44, A45, A48, A54, A57, A61, A64, A74, A75, A79, A82, A88, A92, A93, A96, A101, A109, and A110.	

file system that enables mobile users to perform a variety of manipulations on their context data, and approach A101 is a Context Distributed DataBase Management System (CDDBMS) aimed at processing context information as if this is maintained by a single database, while in fact the context information is stored and controlled by multiple administrative domains. For of each of the tasks error recovery and implementation management we found one approach: approach A43 is aimed at restoring applications from an error state caused by problematic contexts, and approach A64 proposes to automate a certain number of high-level implementation decisions in a pervasive application.

E.2 Limitations. A total of 11 approaches of the implementation phase were reported to have limitations, while 24 did not report any limitation. Table 3.15 summarizes these limitations.

Table 3.16: Approaches for the verification and validation phase

Focus	Approaches	Percent
Model checking	A11 [Sama et al., 2010], A14 [Xu et al., 2006], A23 [Xu et al., 2010]	50
Fault detection	A22 [Sama et al., 2008], A27 [Liu et al., 2013]	33
Run time verification	A9 [Coronato and De Pietro, 2012]	17

F Verification and Validation (V&V)

For this phase we have identified 6 approaches and three different focuses corresponding to a particular activity addressed. The focuses are model checking, fault detection, and runtime verification. The categorization between focus and approaches is shown in Table 3.16.

F.1 Approaches. Focusing on model checking we found three approaches: approach A11 (called A-FSM) is an adaptation finite state machine that enables the detection of faults caused by both erroneous adaptation logic and asynchronous updating of context information, approach A14 consists of a consistency checking technique for incrementally checking first order logic rules to address the problem of context inconsistency detection, and approach A23 is a partial constraint checking approach for context inconsistency detection able to distinguish reusable checking results of constraints. Fault detection was found to be the concern of two approaches: approach 22 is a finite-state model of adaptive behavior of context-aware adaptive applications that enables the detection of faults caused by erroneous adaptation logic and asynchronous updating of context information, while approach A27 (called AFChecker) is a tool to automatically deriving domain and environment models for rule-based context-aware applications to improve the effectiveness of the existing adaptation fault detection techniques. We identified a single approach focused on runtime verification, approach A9 which proposes to verify dynamically the correctness of the implementation and statically the correctness of the specification.

F.2 Limitations. Two approaches of the verification and validation phase were spotted for limitations, whereas 4 approaches did not report any limitation. These limitations are presented in Table 3.17.

G Testing

We have identified a total of 10 approaches targeting the testing phase. We classify them by their focus or concern and we obtained 5 dedicated to context-aware testing, 3 to simulators, and 2 to test adequacy. Table 3.18 shows this classification between focus and

Table 3.17: Limitations of approaches for the verification and validation phase

Approach	Limitation
A14	◇ The link generation process may produce redundant links when a pattern appears more than once in a rule ◇ Considerable space is necessary for keeping last checking results
A23	◇ Considerable space cost for storing previous checking results
Approaches not reporting limitations: A9, A11, A22, and A27.	

Table 3.18: Approaches for Testing phase

Focus	Approaches	Percent
Context-aware testing	A20 [Wang et al., 2007], A52 [O'Neill et al., 2013], A83 [Wang et al., 2014], A111 [Morla and Davies, 2004], A129 [Huang et al., 2010]	50
Simulator	A51 [McGlinn et al., 2014], A66 [Nishikawa et al., 2006], A76 [Bruneau and Consel, 2013]	30
Test adequacy	A21 [Lu et al., 2006], A24 [Lu et al., 2008]	20

approaches.

G.1 Approaches. We identified 5 approaches focused on context-aware testing: approach A20 is an approach that improves the context-awareness of an existing test suite by providing an integrated solution to identify when context changes may be relevant and a control mechanism to guide the execution of given tests into potentially interesting contextual scenarios, approach A52 (called InSitu) is a situation-based testing approach that seeks to examine the effect of exhibited pervasive system behavior in a simulated physical deployment environment, approach A83 is an approach meant to select test cases for constructing test suites that are adequate with respect to the data-flow testing criteria, approach A111 is a test environment to support the evaluation of key aspects of location-based applications without the extensive resource investment necessary for a full application implementation and deployment, and approach A129 (called CIRST) is a cyber-physical instrument for real-time hybrid structural testing that provides specialized infrastructure to enforce type safety and timing properties within and between components.

We found three simulators: SimCon (A51) is a context simulator to support evaluation of smart building applications when faced with uncertainty, UbiREAL (A66) is a smart space simulator that provides a virtual testbed for ubiquitous applications in a 3D space, and DiaSim (A76) is simulator meant to allow for the same application code to be simulated

Table 3.19: Limitations of approaches for the testing phase

Approach	Limitation
A20	◇ The approach generates infeasible drivers
A52	◇ Lack of support for traceability
	◇ InSitu does not enable the designer to examine the internal process of the system
A76	◇ Lack of support to ease the simulation of physical phenomena
A83	◇ The proposed algorithm remains inapplicable for hybrid systems with derivative dependencies
A111	◇ It considers aspects that are too narrow as part of the simulations
Approaches not reporting limitations: A21, A24, A51, A66 and A129.	

Table 3.20: Approaches for Deployment phase

Focus	Approaches	Percent
Dynamic deploy	A34 [Verbelen et al., 2011], A71 [Hoareau and Mahéo, 2008], A103 [Chan and Chuang, 2003]	43
Multi-platform deploy	A74 [Paspallis and Papadopoulos, 2014], A75 [Soldatos et al., 2007]	29
End-user deploy	A67 [Kawsar et al., 2008]	14
Large-scale deploy	A59 [Gopalan and Znati, 2010]	14

or executed in the real environment. Furthermore, we found two approached focused on test adequacy: approach A21 is a family of context-aware data flow test adequacy measurement criteria for context-aware middleware-centric applications, and approach A24 is a family of test adequacy criteria to test context-aware applications in the presence of context inconsistency resolution services.

G.2 Limitations. A total of 5 of the approaches pertaining to the testing phase have been spotted for limitations, while the other 5 approaches did not report any limitation. The limitations are presented in Table 3.19.

H Deployment

We identified 7 approaches for the deployment phase. From these, three approaches focus on dynamic deployment, two approaches focus on multi-platform deployment, and end-user deployment and large-scale deployment are each the focus of one approach. The classification between focus and approaches is shown in Table 3.20.

Table 3.21: Limitations of approaches for the deployment phase

Approach	Limitation
A59	◇ Lack of support for security
A67	◇ Lack of support for spatially distributed deployment
Approaches not reporting limitations: A34, A71, A74, A75, and A103.	

H.1 Approaches. We found three approaches focused on dynamic deployment: approach A103 (MobiPADS) supports dynamic adaptation at both the middleware and application layers to provide flexible configuration of resources to optimize the operations of mobile applications, approach A34 proposes a middleware framework to select and deploy the configuration offering the best quality possible for the current connectivity and available resources, and approach A71 is a purely descriptive language aimed at specifying deployment descriptors that allow for a context-aware deployment. Multi-platform deployment was found as the focus of two approaches: approach A74 is a component-based middleware architecture to facilitate the deployment of context-aware applications via reusable components, and approach A75 is an Architectural framework called 'breadboard architecture' that facilitates the deployment of pervasive applications based on components contributed by different technology providers. For each of the concerns end-user deployment and large-scale deployment we found one approach: approach A67 consists of an infrastructure and a tangible deployment tool to provide the foundation for end-user deployment, while approach A59 (called SARA) proposes a unified, overlay-based service architecture to support large-scale service and application deployment in pervasive and ubiquitous environments

H.2 Limitations. Only two of the approaches pertaining to the deployment phase have been spotted for limitations, while the other 5 approaches did not reported any limitation. The limitations are presented in Table 3.21.

I Evolution/maintenance

We identified a total of 30 approaches for this phase. We found most of the approaches focusing on adaptation (12 approaches) and dynamic configuration (9 approaches). The classification between focus and approaches is shown in Table 3.22.

I.1 Approaches. We identified 12 approaches focused on adaptation: approach A29 is a middleware that aims to solve the critical issue of the long reconfiguration time of context-aware reflective middleware, approach A35 (POISED) is is a general quantitative method to automatically making adaptation decisions under internal uncertainty, approach A40 (PCOM) is component system that supports automatic adaptation in cases where the execution environment changes to the better or to the worse, approach A49 (CAMPUS) is a context-aware middleware aimed to dynamically derive adaptation decisions according to run-time contextual information, approach A50 is a model-driven

Table 3.22: Approaches for evolution/maintenance phase

Concern/Type	Approaches	Percent
Adaptation	A29 [Liu and Cheng, 2011], A35 [Esfahani et al., 2011], A40 [Becker et al., 2004], A49 [Wei and Chan, 2013], A50 [Serral et al., 2010], A60 [VanSyckel et al., 2014], A86 [Cooray et al., 2013], A97 [Malandrino et al., 2010], A99 [Schuhmann et al., 2013], A103 [Chan and Chuang, 2003], A111 [Morla and Davies, 2004], A113 [Chuang and Chan, 2008]	40
Dynamic configuration	A13 [Poladian et al., 2004], A30 [Morin et al., 2009], A44 [Hu et al., 2008], A70 [Gu et al., 2004], A100 [Seinturier et al., 2012], A114 [Capra et al., 2003], A116 [Gámez and Fuentes, 2011], A120 [Sethi et al., 2014], A131 [Gamez and Fuentes, 2013]	30
Monitoring	A10 [Schreiber et al., 2012], A12 [Forte et al., 2008], A84 [Gui et al., 2011]	10
Consistency management	A37 [Xu and Cheung, 2005], A123 [Inverardi and Tivoli, 2013]	7
Self-maintenance	A4 [Mamei and Zambonelli, 2009], A62 [Ou et al., 2007], A117 [Cetina et al., 2009]	7
End-user maintenance	A125 [Guo et al., 2010]	3

development method for supporting the adaptation of the system according to the context information, approach A60 (COMITY) includes an adaptation interface to resolve detected interferences by instructing applications to adapt, approach A86 is a resilient situated software system devised to continuously analyze and dynamically adapt the software to deal with changes in the execution context that could degrade its reliability, approach A97 (called MIMOSA) is a distributed framework to allow programmers to modularly build complex adaptive services starting from simple ones, approach A99 is a configuration approach devised for efficient and adaptive composition of distributed applications, approach A103 (called MobiPADS) includes a reflective-based mobile middleware that seeks to support dynamic adaptation, approach A112 is an adaptation framework that supports adaptation behavior evolution, and approach A 113 is a middleware framework to effectively adapt mobile services at the middleware level to match the dynamics of mobile environments.

We identified 9 approaches related to dynamic configuration: approach A13 is an analytical model for dynamic configuration of resource-aware services in ubiquitous computing environments, approach A30 is an approach for automatically determining a safe transition to make the system evolve from its current configuration to the target configuration, approach A44 (ACoMS) is an autonomic context management system devised to support dynamic configuration and re-configuration of the set of context

information sources required by applications, approach A70 is an adaptive offloading system to enable dynamic partitioning of the application and efficient offloading of part of its execution to a nearby surrogate, approach A100 (called FRASCATI) is a platform that provides the required capabilities to manage at run-time a component configuration, approach A114 (called CARISMA) is a mobile computing middleware that provides software engineers with primitives to describe how context changes should be handled using policies, approach A116 (called FamiWare) is a family of configurable middleware that allows the generation of different middleware configurations, approach A120 proposes a user-assisted protocol to provide a one-pass secure configuration procedure for wireless displays, and approach A131 (called Hydra) is an approach targeted to automatically propagate the evolution changes of the middleware family into the existing configurations where the middleware is already deployed.

We found three approaches focused on monitoring: approach A10 (called PerLa) is a SQL-like language that allows end-users and high-level applications to gather and process information without any knowledge of the underlying pervasive system, approach A12 (called EICAF) is an Internet content adaptation framework that seeks to allow the use of ontologies and Web services in the development of applications for the content adaptation domain, and approach A84 (ACCADA) is an architectural framework targeted to monitor a running system's run-time properties. Having consistency management as their concern we found two approaches: approach A37 proposes a proactive repairing mechanism to realize automatic inconsistency repairing, while A123 consists of a method that supports connector evolution and automated generation of the connector's implementation code. We identified three approaches around self-maintenance: approach A4 (called TOTA) entails distributed tuples spread over a network that enforce dynamic and adaptive coordination patterns in a structured and modular way, approach A62 proposes a lightweight and efficient offloading middleware to provide runtime offloading services for resource constrained mobile devices, and approach A117 allows to model the effort made at design time to not only be useful for producing the system but also provides a richer semantic base for autonomic behavior during execution. We found one approach focused on end-user maintenance, namely, approach A125 (called iPlumber) which is a user-oriented management system that enables users to act as software-plumbers to install and maintain systems.

I.2 Limitations. A total of 8 approaches of the evolution/maintenance phase were reported to have limitations, whereas 22 approaches did not report any limitation. Table 3.23 summarizes these limitations.

3.3.3 Research challenges (answer to *ch3-res-Q3*)

In this section we answer what we find to be the main research challenges for the development of software for pervasive systems according to state-of-the-art approaches.

Empirical research. There is a need to rely more on empirical research and proper reporting of results to lay a more solid ground that supports further research and benefit

Table 3.23: Limitations of approaches for the evolution/maintenance phase

Approach	Limitation
A4	◇ Lack of proper access control models
A12	◇ Scalability: Lack of support for the execution of processes that rely on conditionals such as If-Then-Else and Repeat-Until
	◇ Service composition: Missing a service with multiple profiles
A13	◇ Model proposed depends on accurate prediction of applications' resource demand, for which there is no perfect proving method
	◇ Proving that independence conditions hold in each case, as it is needed by the model, is not always possible
A37	◇ The generation of all possible inconsistencies is impractical
A49	◇ Lack of terminating condition in the decision model
A84	◇ Decreasing of performance of adaptation as the number of installed components increases
	◇ There is no way, currently, to prevent the framework from indiscriminately accepting invalid tactics and strategies from context-specific reasoners, which may bring the system into adverse state
A103	◇ Reduction of overall processing rate caused by adding more mobilets
A114	◇ The approach does not cope with the issue of having the possibility of storing only a minimum set of behaviors on a device
Approaches not reporting limitations: A10, A29, A30, A35, A40, A44, A50, A60, A62, A70, A86, A86, A97, A100, A111, A113, A116, A117, A120, A123, A125, and A131.	

future implementations. Only 35% of the approaches of this review have been recognized to have limitations by their authors, with some phases having as few as 14% (feedback), 27% (evolution/maintenance), and 29% (deployment) of the approaches recognizing limitations. This may have a negative impact on future research as it is in many cases unclear how effective and reliable are the approaches found in the literature.

Context related issues. It being such an important concept in the development of pervasive systems, there is a need for more research on issues related to the concept of context, important for different phases of the development cycle. In relation to design, it is necessary to further investigate how to design systems that can automatically adapt based on their context, as well as to study light-weight context modeling techniques to avoid overheads and low performance of the running system. Concerning the feedback phase, more research is necessary on context modeling techniques that allow adequate expressiveness levels. Study how to add to the system the ability of predicting its context remains a challenge related to the evolution and maintenance of this type of systems. Further research is needed to be able to simulate context in a realistic way to test systems before being implemented.

Runtime support. Making systems able to adapt to different circumstances at runtime is a concern of the development of pervasive systems that needs further investigation. In relation to the design phase, more work is required to investigate how to design systems that are aware of time and can track and handle changes without interrupting their operation. Other runtime capabilities that affect the evolution of a system need further research, such as how to better monitor and adapt to context and how to fix errors at runtime.

User interaction. A relevant concern that needs further attention from the research community refers to issues related to how people interact with pervasive systems. It is necessary to further investigate how users interact with technical system, how different users may differ in terms of skills and preferences, and find ways to allow the configuration and reconfiguration of the system by the users according to their needs. Furthermore, there is a need to explore how to generalize results obtained in research contexts with specific data sets and subjects to cover real-world cases where parameters may vary considerably.

Support for developers. An important challenge for researchers is to investigate how to better support developers in building pervasive systems. There is a need to investigate how to better build systems that can be transparent and easy to read for developers and other stakeholders. Further research is required on methods for automatically generated programming. More research is necessary on techniques and methodologies that help developers design and deploy their applications to different pervasive systems. It is necessary to investigate how to better gather, manage, and use data that is available in pervasive domains. Furthermore, a tool that can support the entire development cycle is a concern that have to be target by future research.

Security and privacy. These concerns have been the target of many research efforts, however issues are still to be addressed such as those derived from the increasingly pervasive use of presence-based contextual data in multiple enterprise and provider domains.

Self-adaptation. Different aspects related to self-adaptation of pervasive systems require further investigation such as self-optimizing and self-protecting of system features. There is a need to investigate how to apply/adapt existing machine-learning methods for enhancing systems' self-adaptation.

Systems' requirements, behavior and constraints. There is a need for further research on how to better assess and understand pervasive systems in terms of level of abstractions, priorities, etc., avoiding impractical methods and scenarios. Simulation requires more research work to investigate how to better simulate physical phenomena as well as a larger number of scenarios that vary in complexity. Finally, it is necessary

to favor the use of graphical user interfaces right from the beginning of research and development efforts to explore new methods and build systems around users' needs.

Systems' performance, operation, and data. Issues related to the performance, operation, and data exchange and storage have to be targeted by future research efforts. Further research is needed on how to maintain the consistency and performance of pervasive systems after adaptation and inclusion of more devices and functionalities. It is necessary to investigate possible ways that allow pervasive systems and subsystems to access relevant data even when connectivity is not available, as well as solutions to optimize the storage and exchange of data.

3.4 Discussion

3.4.1 Strength of evidence

It is important to know how much confidence one can have in the results obtained, bearing in mind that the studies from where the results were obtained have been carefully selected from the entire space of studies in the literature.

To grade the strength of evidence of the studies selected for the review we were inspired by the process followed by Dybå and Dingsøyr [Dybå and Dingsøyr, 2008], using the GRADE (Grading Recommendations Assessment, Development and Evaluation) working group definitions [Group et al., 2004]. The studies selected are assessed on the study design, study quality, consistency (i.e., the similarity of estimates of effect across studies), and directness (i.e., the extent to which the people, interventions, and outcome measures are similar to those of interest), with a scale that comprises high, moderate, low, and very low grades.

Study design. Most of the papers claim to present the evaluation of the proposed approach based on one or more case studies. However, we found that there is in general major issues related to partial or complete misunderstanding on what a case study is and how it is to be conducted and presented. This issue has been prevalent in the area of software engineering since many years now as pointed out in an ICSE tutorial from 2004 [Perry et al., 2004]. This becomes worse when dealing with areas such as software intensive systems where people with diverse backgrounds work together.

Quality of studies. Methods of the selected studies were in general not well described. Papers published on journals were more detailed and in most cases of higher quality. Studies rarely consider issues of bias, validity, and reliability, lacking in general clear and detailed explanation about data collection and analysis procedures. In addition, findings were not compared with others in a systematic and reliable manner.

Consistency. We found hard to relate the studies since none of the studies followed guidelines nor share methods for gathering and synthesizing results.

Directness. Our premise for directness is that the ultimate interest on developing approaches for the engineering of software for ubiquitous systems is to have solutions that can be applied to real scenarios delivering value to users. At this respect, we found that most studies are based on evaluations that are far from covering the issues related to real scenarios, either by not considering physical phenomena in an extent that can be useful (usually considering too few variables), or by only considering narrow experiments that do not contemplate the full range of cases that a system may encounter in terms of users, networks, real environments, etc. We can say that there are important uncertainties about the directness of the included studies.

The combination of the four key elements for qualifying the evidence provided by the studies considered (viz., study design, study quality, consistency, and directness) allows us to establish that the strength of evidence in the review herein presented, regarding the applicability of approaches, and the validity of their claims in terms of benefits and limitations, is *very low*. Thus, we can say that one cannot rely on the estimates of effect that are based on evidence provided about approaches and their limitations for developing software for ubiquitous systems. This problem, we think, could be greatly alleviated if researchers in this area followed systematic guidelines for conducting the studies and strive to apply their approaches to real scenarios, including more variables.

3.4.2 Limitations of this review

The main limitations of this review relate to publication selection bias, inaccuracy in data extraction, and misclassification.

To avoid selection bias we use a hybrid setup for the search processes that guided the initial selection of studies based on the literature rather than only on our experience and knowledge. At this respect we also performed three separate search processes of different type each. Finally, we distributed the work load in such a way that the search and selection tasks were performed by more than one researcher and checked by one researcher not involved in the original process. This last point was aimed also at increasing the accuracy in data extraction. However, the data extraction was highly dependent on the way studies were reported and on how authors of the different papers explained concepts and methodologies, as well as the kind of terminology they used. This in many cases hindered relevant information, which may have derived in certain level of inaccuracy in the data.

Although the classifications we performed over the data were based on what was found on the selected papers, the decision remained subjective in that it depends on how the original authors decided to present their results and what we could consider relevant on them.

3.4.3 Implications for research and practice

This systematic review can serve both practitioners and researchers. Practitioners can use this review to get a quite representative state of the research landscape in the approaches for engineering of software for ubiquitous systems. The results and studies presented in the papers listed in this review might not be in some cases directly applicable in industry,

but they can be useful to feed the reflections when looking for solutions during the process of development.

For research, the review shows that while some phases of the development cycle have received much attention due to their importance (viz., feedback and evolution/maintenance) or direct practical usability (viz., implementation), other phases equally important still need more and better work (e.g., testing). Based on the review researchers can direct their research more precisely according to the open issues identified and research efforts repartition that has been described in this review.

3.5 Summary

In this chapter we provided solution to *RQ1-b*. Specifically, we presented some relevant aspects of the development cycle to build software for pervasive systems, as seen in the literature: *i*) phases of the development cycle, *ii*) main state-of-the-art approaches and their limitations for each of the phases of the development cycle, and *iii*) open issues and research challenges that have been identified in the past.

Our findings show that the development phases that have been considered are: design, implementation, deployment, validation and verification, testing, feedback, and evolution and maintenance. From there, the feedback phase is particularly noteworthy, as the literature highlights its importance for the case of the development of software for pervasive systems.

We have found that the works in the literature recognize the need for further investigation around context-aware issues, self-adaptability, machine learning approaches, runtime support, among others. It has been also highlighted the importance that empirical research has for this type of development and the gap that still exist between optimal empirical evaluations and the studies presented in the literature.

To obtain the results we presented in this chapter, we followed a systematic review methodology. We combined automatic, manual, and snowballing search techniques, considering the main electronic databases, as well as the most relevant conference proceedings and journals on ubiquitous computing and software engineering. We carefully selected the studies ultimately considered to obtain our results, based on a well-defined criteria to only include in our review evidence based on sound research efforts.

Chapter 4

Engineering Model for Software for Ubiquitous Systems

4.1 Introduction

The goal of this chapter is threefold:

i) Provide solution to the last question in which we divided *RQ1*, namely *RQ1-c*: What are the key mechanisms to support the development of software for pervasive systems featuring smart functionality?

ii) Combine the answers to *RQ1-a*, *RQ1-b*, and *RQ1-c* to give solution to *RQ1*

iii) Lead the way from the answer to *RQ1* and into the second research question of the dissertation *RQ2* by providing solution to question *RQ2-a*

This chapter is organized as follows. In Section 4.2 we first describe what are the data analytics mechanisms that, according to the literature are essential to support the creation of software services featuring smart functionality in general (i.e., answer to *RQ1-c*). Then, we narrow down the focus to specific mechanisms, from those identified previously, which are important in delivering value to users (answer to *RQ2-a*). In Section 4.3 we present our model for the systematic engineering of value-added smart software services for pervasive systems (i.e., answer to *RQ1*). The last section, Section 4.4, summarized the content of the chapter

4.2 Analytics for smart functionality and value

In this section we provide answer to *RQ1-c* and *RQ2-a*.

In order for pervasive systems to comply with the way they have been envisioned (i.e., computing systems that consider the natural human environment and push the devices and services to vanish into the background in order to support all aspects of our everyday life [Weiser, 1991]), they need to feature smart behavior which allows the systems to learn from and react to their surroundings [Bures et al., 2017].

The smart functionality of pervasive systems is mainly implemented through software, focusing on self-awareness, self-adaptation, and self-optimization capabilities in a context-

aware manner, where the information about the situation of users, the environment, and the state of the system itself are considered to adapt the behavior of the system accordingly [Dey, 2001]. At this respect, the learning and adaptation is to be implemented as a continuous cycle, since the information gathered from the surroundings is expected to change over time [Galushka et al., 2006]. Smart capabilities should include the capability of reasoning over the collected data from the sensors, including previous knowledge of the context, adaptability to changing situations in the environment and their users, and learning [Moreno et al., 2017].

Data analytics mechanisms based on Machine Learning (ML) techniques have been the base for the development of pervasive systems that can learn from and react to the dynamic characteristics of the users, the environment, the system itself, and other context factors [Cook and Das, 2004]. Furthermore, these mechanisms should allow to: extract accurate preference from the monitored contexts and behaviors in a constantly changing environment, keep the preference set up to date by responding rapidly to changes in users' behavior while being not sensitive to noise-like deviations in their behavior, and produce human-readable output when necessary [Bental et al., 2015]. In spite of the important progress made in this area, no holistic learning approach, which is applicable to all cases, have been achieved [Müller, 2004]. Therefore, the development of services that are able to deliver smart functionality to users requires to consider new ML-based mechanisms that take into account the needs of each particular environment [Aztiria et al., 2010].

The development of software aimed at delivering value requires to consider user-centred approaches to the design of technology intended to understand users as well as the context in which they live and operate [Mancini et al., 2017]. Furthermore, the use of various pervasive technologies, such as wearables, smart devices, and sensors should be implemented to allow learning about users' habits and behavior and carrying out intelligent automatic and adaptive tasks involving more convenient and personalized services [Kidd et al., 1999].

4.3 Engineering Model

In this section we introduce a model for engineering smart software services for pervasive computing systems, which is our answer to *RQ1*. We derive this model by framing the findings obtained in the systematic literature review in Section 3.3 (Chapter 3), using the engineering model for software-intensive systems presented in Section 2.4 (Chapter 2). That is, we use the model aimed at building software services and products for software-intensive systems as the basis to define the engineering model to build software for pervasive systems, which are a subset of software-intensive systems. In Figure 4.1 it is presented the model for engineering software for pervasive systems.

As it can be observed in Figure 4.1, most of the theory derived in Section 2.4 (Chapter 2) for software-intensive systems apply directly to the case of pervasive computing. However, we define some aspects that are particularly relevant for the case of pervasive systems.

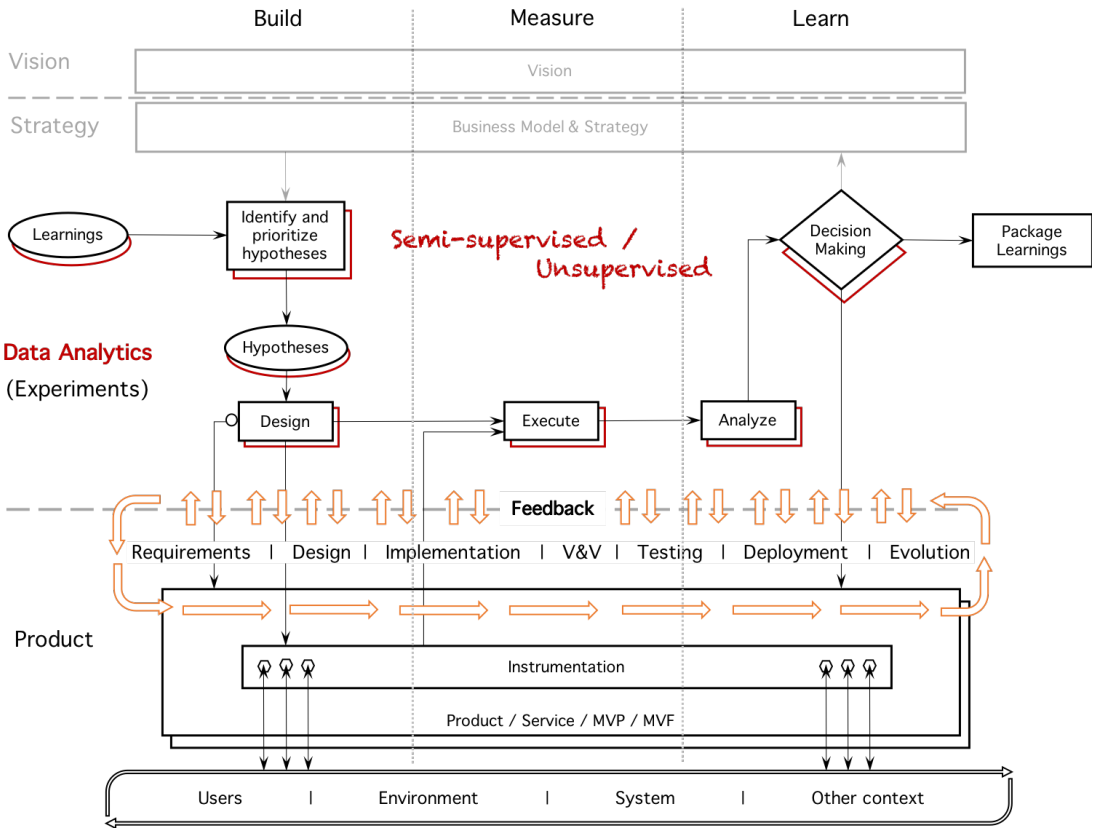


Figure 4.1: Engineering Model for Software for Ubiquitous Systems

1. The experiments, which are a relevant aspect for the case of software-intensive systems as a vehicle to deliver higher-value software products/services to users, take a different dimension in the case of pervasive systems. In this case, the experiments can be seen in a more specific sense as data analytics mechanisms which are essential for the whole process. That is, without data analytics it is not possible to build software that allows pervasive systems to be in the way they are envisioned (see Section 1.1 in Chapter 1).

2. While for software-intensive systems the experiments of the engineering process can be manual or semi-supervised, for pervasive systems (which are expected to feature self-adaptability) data analytics should be mostly semi-supervised and whenever appropriate and possible unsupervised. That is, Machine Learning (ML) techniques are fully integrated into the mechanisms, allowing to deal with complex tasks while reducing considerably the manual work needed [De Masi et al., 2016].

3. Also due to self-adaptability requirements, the data analytics mechanisms should be able to yield and use learning on different static and dynamic context factors, including features about the users (e.g., preferences, behavior, identity), characteristics of the environment, and the state of the logical and physical components of the system.

4. All the phases of the development life cycle, except for feedback, find fit in a non-

sequential cycle in which all or some phases might be present, depending on the specific case. In what respect to the feedback, it should be considered as a vital and continuous part of the development life cycle, which serves as the bridge between the data analytics processes and the other phases of the engineering process.

5. The infrastructure proposed for the development cycle of software for software-intensive systems can be used as basis for the case of pervasive systems. However, in pervasive systems the instrumentation is vital to allow the system to operate and including mobile network, devices, and sensors. The instrumentation is necessary to obtain the data for analysis from the users, the environment, the system itself, and other context factors.

4.4 Summary

In this chapter we answered *RQ1-c* to then provide solution to *RQ1* by combining our results from *RQ1-a*, *RQ1-b*, and *RQ1-c*. In addition, we gave our answer to *RQ2-a*. Specifically, we described based on the literature what are the main mechanisms necessary to provide smart functionality in pervasive systems. Then, we presented our proposed model for the systematical engineering of software for pervasive systems that feature smart functionality. Based on our engineering model and taking into account the outcome of the research work to answer *RQ1-a*, we give our answer to *RQ2-a* which leads the way from our model and into the answer we later give for *RQ2*.

Our proposed engineering model highlights the importance of data analytics mechanisms based on machine learning techniques as an essential element to support the whole development cycle. The model specified the typical phases of the development of software as part of a cycle where the only constant is the feedback component supported by the data analytics mechanisms. Furthermore, our model stresses the importance of the instrumentation, including smart devices and sensors, as a key part for the operation of the system, as well as to obtain and provide information to the users, the system, and the environment.

Part III

Mechanisms to Support the Development of Value-Added Software Services for Smart Environments

We present two essential mechanisms to support the development of value-added software services for smart environments. Furthermore, we present potential application services that can be built from the presented mechanisms.

The sections in this part are based on the work that has been presented in the following papers:

- *(Section 5) A. S. Guinea, A. Boytsov, L. Mouline, and Y. Le Traon. "Smart Discovery of Periodic-Frequent Human Routines for Home Automation," submitted to* IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2019.*
- *(Section 6) A. S. Guinea, A. Boytsov, L. Mouline, and Y. Le Traon. "Continuous Identification in Smart Environments Using Wrist-Worn Inertial Sensors," in* 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2018)*, November 2018.*

Chapter 5

Discovery of Users' Daily Routines at Home

5.1 Introduction

In this chapter we answer one of the research questions in which we have broken down *RQ2*, that is, *RQ2-b*: How to automatically learn users' routines at home, even when they are only frequent on specific periodicities?

The vision of our approach can be understood with the following scenario (depicted in Figure 5.1): Supposed a user enjoys watching a TV show every Monday at 17 hrs. While doing so the user likes to turn on the heating radiator and close the blinds in the room. This type of routines are indeed very common in people's behavior and thus relevant in learning their behavior patterns. As discussed previously in the dissertation, an important supporting aspect for a smart home system to provide value through intelligent and proactive assistance to users is to know the users' common behaviors and routines [Aztiria et al., 2012]. Thus, the goal is for the smart home system to infer routines as the one described from event data of smart devices and sensors deployed in the space where the user watches the TV. The challenge is, however, that existent approaches are only able to find routines that happen frequently in general and not for specific periodicity as in our example scenario, where the periodicity is 'on Mondays at 17 hrs.'

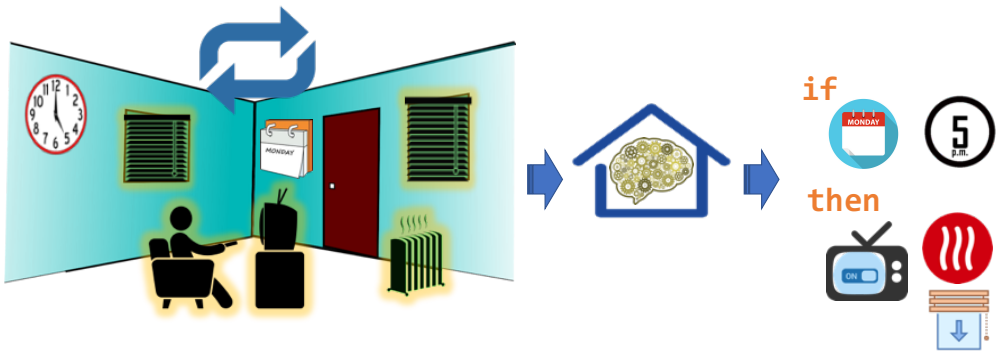


Figure 5.1: Discovery of users' periodic-frequent routines

In this chapter we present an unsupervised approach to discover human routines, with a specific time and periodicity of occurrence, in unlabeled event data from smart devices and sensors deployed at home. That is, our approach is able to discover both routines that are in general frequent and those that are only frequent for specific periodicities. Furthermore, our approach is built to be resilient to variability in the way in which people at home perform their routines, being able to recover periodic-frequent routines even when they are not performed with exactly the same actions or at exactly the same time.

To assess the performance of our approach we conduct an extensive evaluation that consists of an in the lab study, a study based on synthetic data, and an in-the-wild study. In all cases we assess the routines found by our approach against the routines actually performed by participants (for in the lab and in-the-wild studies) or simulated in the synthetic data. Our results indicate that overall the approach has a high recall-precision performance, being able to recover around 90% of the base routines in most cases. Furthermore, our evaluation shows that our approach is robust in terms of being able to maintain its performance level while tested under various heterogeneous scenarios and with widely different combination of parameters.

The rest of the chapter is organized as follows. In Section 5.2 we describe the related works, including state-of-the-art approaches for pattern mining, finding human routines in event data, and mining periodic-frequent patterns. We present our approach in Section 5.3, where we first briefly describe the pattern mining algorithm we use as a basis, to then go over the steps of our approach in detail. In Section 5.4 we present our evaluation methodology, with the details about each of the three studies we conduct to assess the performance of our approach. We present our results in Section 5.5, where we answer to the questions that guide our empirical evaluation. In Section 5.6 we discuss about the limitations of our approach, its practicality to be implemented in real scenarios, and the validity of our evaluation. To close the chapter, we present a summary in Section 5.7.

5.2 State-of-the-art approaches

5.2.1 Pattern mining

The discovery of patterns in event data has been extensively studied in the literature. The most basic approaches focus on unsupervised methods for obtaining frequent sequences of events in data [Agrawal and Srikant, 1995, Mannila et al., 1997]. Some works have explored the discovery of frequent patterns in relation to their time occurrence, either matching simple actions of an activity to a specific point in time [Aztiria et al., 2012], or considering the time as a context factor involved in the discovery process [Seiter et al., 2015].

In a somehow similar way as our approach, a number of previous works have focused on discovering periodic-frequent patterns in transactional databases [Tanbeer et al., 2009, Surana et al., 2011, Amphawan et al., 2009, Venkatesh et al., 2016]. The center of attention in some cases have been to improve the efficiency and overall performance of existing algorithms [Surana et al., 2011] and to deal with the increasingly widespread issue of big data databases [Kiran et al., 2016]. Some other works have recognized, just as

we have, the need to deal with partial patterns, which are more common to appear in real-world scenarios [Kiran et al., 2017]. For time series data various approaches have been proposed [Esling and Agon, 2012], however none of them consider the temporal information as part of the discovery process. Recently, a distinct model to find partial periodic-frequent patterns in a transactional database has been proposed [Kiran et al., 2017]. This model is similar to our approach in the importance it gives to periodicity for finding patterns and in its focus on partial periodic-frequent patterns. However, whereas our approach is targeted to event data from human routines, their approach focuses on web activity data. Overall, the approaches mentioned about periodic-frequent mining of patterns in transactional databases consider as input a set of transactions, which in our case is produced from event data.

For the specific case of smart environments, different previous works have explored approaches to identify regularly-occurring interactions between inhabitants and smart home, based on data mining approaches [Heierman and Cook, 2003, Aztiria et al., 2010]. However, these approaches do not focus on routines with a specific periodicity and frequency.

5.2.2 Finding human routines in event data

The most prominent approaches related to finding human routines in data have been typically around the recognition of routines, which requires a labeling and training phase. In this case, the problem is solved mostly through statistical (machine learning) classification methods [Van Kasteren et al., 2008, Kim et al., 2010, Wen et al., 2015, Hu et al., 2017]. In spite of the success shown by approaches for the recognition of human routines in home environments, their need for a labeling and training phase represents an issue, as the training process requires the labels to be both completely and accurately annotated beforehand. Recently, different approaches have been proposed in an attempt to alleviate this issue, allowing the use of uncertainty in labels [Hu et al., 2017] or the recognition based on a limited amount of labeled data [Wen and Wang, 2017]. However, labeling routines, which is an error-prone and laborious process [Nguyen et al., 2015], remains necessary.

To a lesser extent, different unsupervised approaches have been proposed to mine activities from data, without the need of data labeling and training [Kim et al., 2010, Rashidi et al., 2011, Cook et al., 2013]. These approaches, however, exhibit low performance in terms of the number of patterns discovered. As an answer to such performance issues, some methods have resorted to combine a small amount of labelled data with unlabelled data [Wen and Zhong, 2015, Wen and Wang, 2017]. Other approaches have considered the use of more context sources to achieve better recognition results [Wen et al., 2015, Wen et al., 2016]. All these approaches use activity discovery as an step in the process of activity recognition. Furthermore, they do not consider any type of temporal or periodical component.

Concerning the problem of discovering human routines from sensor data, parametric topic modeling has been used by most existing approaches to extract human activity patterns [Huynh et al., 2008], [Farrahi and Gatica-Perez, 2008a], [Farrahi and Gatica-

Perez, 2008b]. Other approaches, instead, have opted for nonparametric approaches in an attempt to avoid the issues related to model selection [Sun et al., 2014], [Seiter et al., 2015]. In terms of the type of sensor data considered by previous works, mobile phone sensor data [Farrahi and Gatica-Perez, 2008a], [Montoliu and Gatica-Perez, 2010] and ambient sensor data [Koreshoff et al., 2013, Dawadi et al., 2016] are among the most relevant. Most existing routine discovery approaches focus mainly on recognizing the high-level activities that form routines, while having the temporal component as a context factor or extra information [Huynh et al., 2008, Seiter et al., 2015, Castro et al., 2015]. In contrast, the main target of our approach is to discover human routines at home with specific periodicity and frequency, while considering low level actions instead of high-level activities. We have found one relevant work that does focus on the periodicity of human routines [Bamis et al., 2010]. However, this approach is limited to the identification of single event routines.

Different from the state of the art, the main target of our approach is to discover human routines at home with specific periodicity and frequency. Furthermore, our approach is unsupervised, without the need for a labeling and training phase, and allows the identification of multi-event routines which are not necessarily performed in an uniform manner.

5.2.3 Mining periodic-frequent patterns

The discovery of patterns in event data has been extensively studied in the literature. The most basic approaches focus on unsupervised methods for obtaining frequent sequences of events in data [Agrawal and Srikant, 1995, Mannila et al., 1997]. Some works have explored the discovery of frequent patterns in relation to their time occurrence, either matching simple actions of an activity to a specific point in time [Aztiria et al., 2012], or considering the time as a context factor involved in the discovery process [Seiter et al., 2015].

In a somehow similar way as our approach, a number of previous works have focused on discovering periodic-frequent patterns in transactional databases [Tanbeer et al., 2009, Surana et al., 2011, Amphawan et al., 2009, Venkatesh et al., 2016]. The center of attention in some cases have been to improve the efficiency and overall performance of existing algorithms [Surana et al., 2011], as well as to deal with the increasingly widespread issue of big data databases [Kiran et al., 2016]. Some other works have recognized, just as we have, the need to deal with partial patterns, which are more common to appear in real-world scenarios [Kiran et al., 2017]. For time series data various approaches have been proposed [Esling and Agon, 2012], however none of them consider the temporal information as part of the discovery process. Recently, a distinct model to find partial periodic-frequent patterns in a transactional database has been proposed [Kiran et al., 2017]. This model is similar to our approach in the importance it gives to periodicity for finding patterns and in its focus on partial periodic-frequent patterns. However, whereas our approach is targeted to event data from human routines, their approach focuses on web activity data. Overall, the approaches mentioned about periodic-frequent mining of patterns in transactional databases consider as input a set of

transactions, which in our case is produced from event data.

For the specific case of smart environments, different previous works have explored approaches to identify regularly-occurring interactions between inhabitants and smart home, based on data mining approaches [Heierman and Cook, 2003, Aztiria et al., 2010]. However, different from our approach, these methods do not focus on routines with a specific periodicity.

5.3 Approach Overview

5.3.1 Background

A Pattern mining

Pattern mining is devoted to discover interesting patterns in databases. In this paper we use as basis of our approach the Generalized Sequential Pattern (GSP) algorithm [Srikant and Agrawal, 1996], which is an extension of the Apriori algorithm [Agrawal et al., 1994, Agrawal and Srikant, 1995]. We first introduce some basic concepts of association analysis, to then describe the steps of the Apriori algorithm, all of which will help to understand the GSP algorithm.

The high-level idea of these algorithms is to be able to associate the occurrence of an item based on the occurrences of other items in a transaction. Let $I = \{i_1, i_2, \ldots, i_d\}$ be the set of all items under consideration (where items can refer to events, actions, or any type of objects) and $T = \{t_1, t_2, \ldots, t_N\}$ be the set of all transactions. Each transaction t_i contains a subset of items chosen from I. An *itemset* is a collection of one or more items. If an itemset contains k items, it is called a k-itemset. We define as *support* the fraction of transactions that contain an itemset and as *support count* the frequency of occurrence of an itemset in a given set of transactions. We say that an itemset is frequent if its support is greater than or equal to a minimum support *minsup* threshold. Furthermore, let *confidence* be the ratio between the number of times two disjoint itemsets X and Y appear together and the total number of occurrences of X in the entire set of transactions, i.e., the confidence determines how *frequently* items in Y appear in transactions containing X.

A.1 Apriori algorithm. The Apriori algorithm was proposed as a scalable method for mining frequent itemsets in a large transactional database [Han et al., 2007]. The algorithm is based on the Apriori principle, which establishes that among frequent k-itemsets, a k-itemset is frequent only if all its sub-itemsets are frequent [Agrawal et al., 1994].

The Apriori algorithm is as follows:

Stage 1. Set $k = 1$ and generate frequent itemsets of length k

Stage 2. Repeat the following steps until no more frequent itemsets are identified.

Step 2.1. Generate $k + 1$ candidate itemsets from length k frequent itemsets

Step 2.2. Discard candidate itemsets containing subsets of length k that are infrequent

Step 2.3. Scan the database of transactions to determine the support of each candidate itemset

Step 2.4. Eliminate candidates that are infrequent, i.e., those which support is not greater than or equal to *minsup*

A.2 Generalized Sequential Pattern (GSP) algorithm. This algorithm is similar to the Apriori algorithm but for mining sequential patterns. It adopts a multiple pass, candidate generate-and-test approach [Srikant and Agrawal, 1996]. The algorithm is as follows:

Stage 1. First pass over the database to yield all k-item frequent sequences for $k = 1$

Stage 2. Repeat the following steps until no more frequent sequences are found.

Step 2.1. Merge the pairs of frequent subsequences found in the $(k-1)$th pass to generate candidate sequences that contain k items

Step 2.2. Discard the candidate k-sequences that contain infrequent $k-1$ subsequences

Step 2.3. Make a new pass over the database to find the support for the candidate sequences obtained so far

Step 2.4. Eliminate the candidate k-sequences with support not greater than or equal to *minsup*

5.3.2 Terminology

Here, we introduce terminology that we use in our approach. Our intent is not to provide a definitive description, but to establish a common understanding about how we use these terms in this work.

Action. We call an action to any interaction that can be recorded between a person and a (controllable) object in a given space. Examples of what we consider an action are: turning On the light, turning Off the TV, opening the window, etc.

Activity session. A sequence of actions performed by a user during a "session" of activity. This definition has been inspired by web management concepts [Halfaker et al., 2015]. A particular activity session is thus described by a *time of occurrence*, a *duration*, and a sequence of actions that form an *activity*. Specifically, for the purpose of routine detection we consider in place of time of occurrence and duration, day of the week and time slot (i.e., start time and end time).

We consider that a "session" is active as long as a person is moving around the space or is interacting with (controllable) objects in that space. We define two ways to distinguish if a session has ended: halt-time threshold and all-counter actions criterion.

halt-time threshold: This threshold, denoted as (*ht*), corresponds to the maximum amount of time after which if no motion has been registered, we consider that the activity session has ended.

all-countered condition: We consider only actions that involve interaction with (controllable) objects at home. Furthermore, we say that counter actions are such that produce opposite effects, e.g., "turning On the light" is a counter action to "turning Off the light" and vice versa, increasing the temperature is a counter action to decreasing the temperature and vice versa. The all-countered condition is satisfied when for every action in an activity there is a counter action within the same activity.

Activity routine. It refers to an activity session defined with a specific periodicity (e.g., on Wednesdays) and frequency (e.g., every Wednesday or half of the total number of Wednesdays in consideration). In our approach we consider *weekly* routines as a basis.

5.3.3 Algorithm

The steps of our approach, including input and output, are as follows:

Input. Interaction and motion event data

Step 1. Detection of activity sessions

Step 2. Discovery of activity routines

Output. Activity routines

Next, we describe each part of the algorithm in detail.

Input – Event data. The event data considered as input for our approach concerns only events reported by sensors and devices that either hint interaction between occupants and (controllable) objects (e.g., appliances connected to a smart plug), or hint occupants' presence through motion.

Step 1 – Detection of activity sessions. This step receives as input interaction and motion event data and yields as output a set of activity sessions of the form

$\langle$`day of the week`, `exact start time`, `extact end time`, $\langle$`action`$_1$, `action`$_2$, $\cdots$, `action`$_n\rangle$ $\rangle$

Detecting activity sessions is divided into two steps:

Step 1.1. Based on the halt-time threshold (*ht*), *detect activity sessions.* The whole event data is processed from beginning to end sequentially. An activity session is considered started when the first event (at the beginning of the event data or after the end of another session) is encountered. From that point on, all events

are considered part of the same activity session until the time between a False and a True value reported by a motion sensor considered exceeds the value of the threshold *ht*.

Step 1.2. For all activity sessions detected, a *split step* is performed to avoid having sessions that contain other sessions. To this end, the all-countered condition is used. The actions of the activity are considered from beginning to end, and if at any point the all-countered condition is met (i.e., a counter action has been found for every action), the session is split at that point into two sessions. The process continues over the second session of the two obtained after the split, until all actions have been processed.

Step 2 – Discovery of activity routines. This step receives as input a set of activity sessions and produces as output a set of activity routines, each specifying its periodicity, frequency, and time slot (time of occurrence). An activity routine has the form

$$\langle \texttt{periodicity},\ \texttt{frequency},\ \texttt{time slot},\ \langle \texttt{action}_1, \texttt{action}_2, \cdots, \texttt{action}_n \rangle \ \rangle$$

The discovery of activity routines comprises two steps:

Step 2.1. Discover tentative activity routines in the set of activity sessions using an adapted version of the GSP algorithm. Specifically, each activity session is seen as a 'transaction' which preserves the order of the activity session. The *day of the week* and each $action_i$ of the activity session are considered items, and the *exact start time* and *exact end time* are combined into one item that corresponds to one of several predefined time slots. We consider predefined time slots of 30 minutes that uniformly divide a whole day. Then, GSP is applied (as explained in Section 2) over the set of activity sessions seen as transactions, producing sequences which support count is greater than or equal to the minimum support *minsup* threshold established. The resulting sequences correspond to tentative activity routines, where day of the week is the periodicity of the routine—since we consider weekly routines—(i.e., what day of the week the routine is repeated weekly) and the frequency of the routine is the number of activity sessions that support it.

Step 2.2 Select activity routines based on a two-stage confidence/weak-confidence process. We establish a minimum 'strong' confidence *minconf* threshold and a minimum 'weak' confidence *min-wconf* threshold, where the confidence is a measure of the ratio between the number of times the routine is repeated (i.e., the number of activity sessions that support it) and the total number of weeks covered by the entire event data. The threshold *min-wconf* can be less than or equal to the threshold *minconf*. The first stage accepts all activity routines which confidence value is greater than or equal to *min-wconf*. In the second stage, the count of support for the computation of confidence of a routine is made allowing one *discrepancy* in the sequence of actions of the activity sessions considered. A discrepancy can

be a *missing* action, an *extra* action, or a *transposition* between two actions. If the confidence value of an activity routine, computed in this manner (i.e., allowing one discrepancy), is greater than or equal to *minconf*, then the activity routine is selected.

Output – Activity routines. This output is the product of Step 2. It contains all the information of the found routines such as time of occurrence, periodicity, frequency, and sequence of activities. This output is used to verify the performance of the approach in terms of discovered routines with respect to base routines. Furthermore, this type of output can serve for further feedback to a learning module of a smart home system.

5.4 Evaluation Methodology

5.4.1 In the lab study

The goal of this study is to evaluate our approach under controlled conditions, while considering a dataset of real sensors/devices events, gathered from scripted routines performed by real people.

A Experimental setup

A.1 Environment. This study took place in the Internet of Things (IoT) laboratory of University of Luxembourg located at the Interdisciplinary Center for Security, Reliability and Trust (SnT). The IoT lab holds an approximate area of 44 m^2, where various experiments related to smart spaces and IoT systems are conducted. For this particular study 4 different spaces were arranged. Each of these spaces resemble a small version of spaces (or rooms) typically found in a dwelling: bedroom (BR), living room (LR), dining room (DR), and study (S). The floor plan of the IoT lab, as prepared for this study, is shown in Figure 5.2. The bedroom (BR) has one window with blinds and a heating radiator underneath, it is furnished with a single bed, a night table, and a small bookcase next to it. The living room (LR) has one window with blinds, a big bookcase near the window, a TV bench, and a two-seat sofa. The dining room (DR) has one window with blinds, a small table near the window, and a dining set with a table and four chairs. Both LR and DR share a heating radiator placed near their windows. The study (S) has no windows, and it is furnished with a small bookcase, a desk, and a chair.

Concerning home appliances, the rooms were equipped as follows: BR has 1 floor lamp (*a-1*) and 1 table lamp (*a-2*); LR has 1 flat-panel TV (*a-3*) and 1 floor lamp (*a-4*); S has 1 table lamp (*a-5*) and 1 laptop (*a-6*). The lab is equipped with an audio system that has speakers in BR, LR, and DR. There are blinds in rooms BR, LR, and DR which are used through remote control.

A.2 Participants. The participants of this study are 3 members of our team. All three participants are male with an average age of 30.

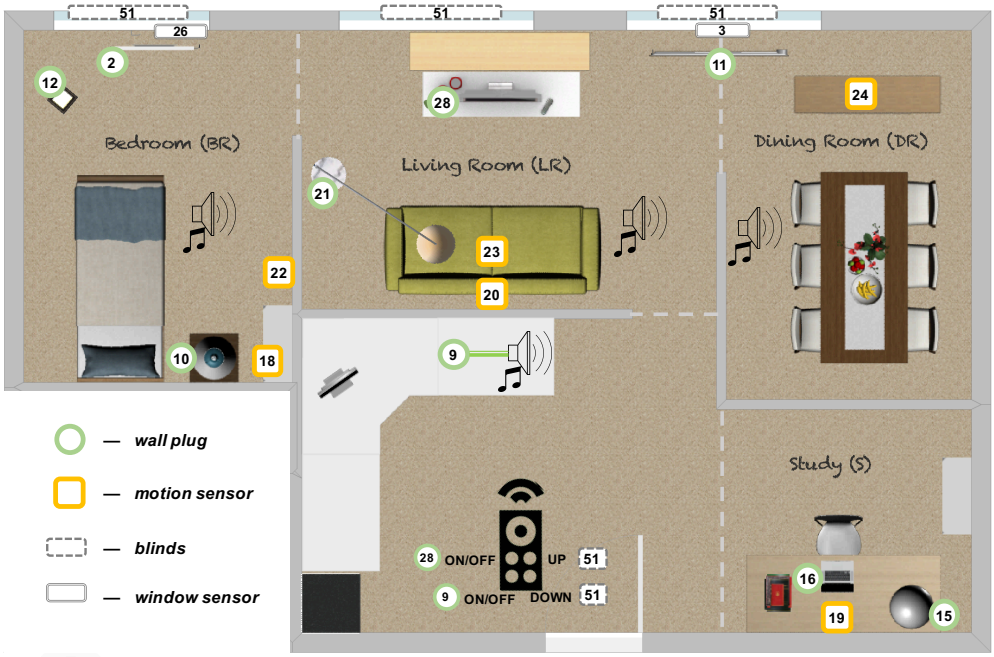


Figure 5.2: Floor plan of the IoT laboratory used as environment for the in the lab study

A.3 Instrumentation. In this study we used a number of smart devices and sensors to capture the interactions between the participants and the home-like environment in the different spaces of the laboratory. These devices and sensors are: wall plugs (Fibaro[1] smart plug), a remote control (Aeotec key fob remote[2]), window sensors (Fibaro[3]), a USB stick for creation and management of a Z-wave network (Aeotec Z-Stick[4]), and multi-sensors (Aeotec Gen5[5] and Gen6[6])—including motion, temperature, and humidity sensors. All the smart devices/sensors employed are Z-wave compatible, allowing manageable integration and interoperability. The Z-stick was paired to all sensors and devices and it was connected to a computer which run our implementation for data collection and management.

A total of 8 wall plugs (*wp-2,10,12,21,28,11,15,16*) were connected between electrical outlets and specific electrical appliances. In *BR* the floor lamp (*a-1*) was connected to *wp-12* and the table lamp (*a-2*) to *wp-10*. In *LR* the TV (*a-3*) was connected to *wp-28*, while the floor lamp (*a-4*) was connected to *wp-21*. In *S* the table lamp (*a-5*) was connected to *wp-15* and the laptop (*a-6*) to *wp-16*. The audio system, with speakers in *BR*, *LR*, and *DR*, was connected to *wp-9*. In addition, we used wall plugs *wp-2* in *BR* and *wp-11* in *LR* and *DR* to register "On" and "Off" states of the heating radiators[7] (every time the participant change the state in the radiator, he had to match the state of the wall plug to the state of

[1] http://manuals.fibaro.com/wall-plug/
[2] https://aeotec.freshdesk.com/support/solutions/articles/6000065292-key-fob-gen-5-user-manual-
[3] http://manuals.fibaro.com/door-window-sensor/
[4] https://aeotec.freshdesk.com/support/solutions/articles/6000056439-z-stick-gen-5-user-manual-
[5] https://aeotec.freshdesk.com/support/solutions/articles/6000056451-multisensor-gen5-user-guide-
[6] https://aeotec.freshdesk.com/support/solutions/articles/6000057073-multisensor-6-user-guide-
[7] we use this over a smart radiator knob for simplicity

Table 5.1: Sensors/devices by room used in the lab study

Room	Sensors/devices
Bedroom (*BR*)	wall plugs: *wp-9* (audio system), *wp-10* (night table lamp), *wp-2* (radiator)
	motion sensors: *m-18*, *m-22*
	outer blinds: *b-51*
Living room (*LR*)	wall plugs: *wp-9* (audio system), *wp-28* (TV), *wp-21* (floor table), *wp-11* (radiator)
	motion sensors: *m-23*, *m-20*
	outer blinds: *id-51*
Dining room (*DR*)	wall plugs: *wp-9* (audio system), *wp-11* (radiator)
	motion sensors: *m-24*
	blinds: *b-51*
Study (*S*)	wall plugs: *wp-15* (table lamp), *wp-16* (computer)
	motion sensors: *m-19*

the radiator).

We used 6 multi-sensors, 2 in *BR* and in *LR*, and 1 in *DR* and *S*, respectively. In *BR*, multi-sensor *m-18* was placed in one of the mid shelves of the bookcase pointing to capture motion on the bed, and multi-sensor *m-22* was set near one of the sides of the room to cover as much of the motion within the space of the room as possible. In *LR*, multi-sensor *m-20* was placed at a height of approximately one meter, on one of the sides of the room, to capture the motion happening in the center of the room, while multi-sensor *m-23* was placed on the top of the sofa to cover the motion of people sitting on the sofa. In *DR*, multi-sensor *m-24* was set on top of the small table near the window to capture as much motion happening around the dining table as possible. In *S*, multi-sensor *m-19* was placed right in front of where a person's upper body would be while sitting, in order to capture the motion of the person while reading or interacting with the laptop.

To allow participants to interact with the audio system and the blinds (in *BR*, *LR*, and *DR*), and the TV (in *LR*) we programmed the key fob remote control. The remote has 4 buttons, of which we programmed one to switch between the "On" and "Off" states of the audio system[8], and a second button to switch between the "On" and "Off" states of the TV. From the remaining two buttons, we use one to take the blinds up and the other one to take the blinds down. The "up" button also serves to stop the blinds if they are being taken down, and similarly the "down" button can be used also to stop the blinds if they are being taken up. Table 5.1 shows the relation between rooms and devices used for this study.

[8] the audio system was set to play continuously through a predefined playlist

A.4 Data collection and management. We collected the events reported by the wall plugs, multi-sensors, and (controllable) blinds deployed in the laboratory. No data was collected concerning the pressing of buttons on the remote. An *event* reported by the sensors and devices of this study includes the following information: *timestamp* of the event, *id* of the device or sensor, *variable* being reported (e.g., a wall plug may report on its "On"/"Off" and on the energy consumption of the appliance connected to it), the *value* of the reported variable, and the *type* of the reported value (e.g., numeric, binary—"On"/"Off" or True/False).

An event is reported by a sensor or device when a variable, which is expected to report changes, gets its value modified. The latency in this case depends largely on the report interval to which the device or sensor is set. For wall plugs and blinds this is not an issue, however, for the motion sensor of the multi-sensor the latency as well as the sensitivity has to be calibrated taking into consideration the kind of motion expected to be sensed, e.g., long motion of people passing by or short movements executed by people while typing or reading. We calibrated our sensors according to their placement in the laboratory.

To manage the data coming from our instrumentation, we use an existing framework based on model driven engineering which allows to store, manage, retrieve, and navigate through historical information of cyber-physical systems in a resource-efficient manner[Hartmann, 2016]. This approach works by having a fixed model of each device with a related timeline, where the modification of the variables of the model together with their timestamp are maintained [Hartmann et al., 2014]. The approach has been successfully applied in the domains of smart grids and IoT [Moawad et al., 2015, Hartmann et al., 2015]. In our case, the use of this framework allowed us to store all the historical information of the events of the devices and sensors, to later retrieve exactly the parameters we need for the specific period of time we were targeting.

B Study design

B.1 Questions. Guided by the goal of this study we target to answer the following questions:

ch5-lab-Q1. How many of the routines followed by people can the approach recover from the event log of smart sensors/devices under the controlled environment of the laboratory?

ch5-lab-Q2. How robust is the approach and what are the parameters that have a higher influence on its performance?

B.2 General description. To answer *ch5-lab-Q1* and *ch5-lab-Q2* we design an experiment that allows to measure the performance of our approach in a systematic way in the controlled environment of the laboratory.

Base patterns. Activity routines defined by design in the schedule followed by the participants.

Friday, December 23, 2016	
9:30am	LM: "DR-b&m", "BR-a&h", "S-b"
...	
Friday, January 20, 2017	
8:30am	ASG: "DR-b&m", "S-a", "LR-b&h"
...	
9:30am	LM: "DR-b&m", "BR-a&h", "S-b"
...	
2:00pm	LM: "DR-b&m", "S-b", "BR-b"
...	
4:00pm	AB: "LR-a&h", "BR-a&h", "S-b"
4:30pm	LM: "BR-c", "DR-b", "S-a"
...	

Figure 5.3: Excerpt of the schedule

Discovered patterns. Activity routines found by our approach from the event data gathered.

Rationale. We define a set of activity routines (i.e., *base patterns*) and arrange them in a schedule form to be followed by the participants of the study. The participants follow the schedule by performing the actions described, while the smart devices and sensors deployed report the events related to motion and interaction with various (controllable) objects around. The reported events are stored according to our data management process. The data collected during the days that the experiment is set to last is used as input to our approach. Finally, the set of activity routines discovered by our approach (i.e., *discovered patterns*) is compared against the base patterns.

B.3 Schedule of scripted activities. We break down scripted routines into activities with a specific time slot (i.e., starting time and end time), as well as a predefined periodicity and frequency. To do this, we first define a set of scripted activities, such as *'watching TV with the blinds half-way open and the heater On'* or *'having a meal with the blinds open while listening to music'*. The script of an activity is defined as to provide a set of steps or actions for the participant to follow, including a suggested minimum and a maximum duration. In addition to the steps, each activity script includes a short description, the room where it is to be performed, the participant to whom the activity is assigned, and an identifier. Figure 5.4 shows an excerpt of the script of activities used in this study.

The scripted activities were obtained from the combination of a simple base of actions,

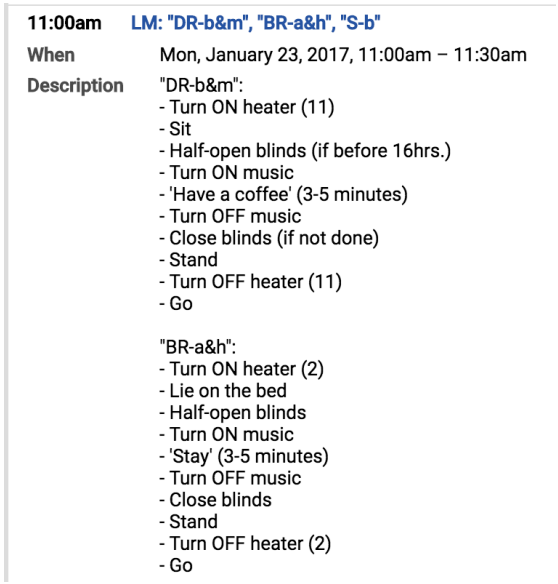


Figure 5.4: Excerpt of the activity scripts

depending on the room. Thus, in *BR* a participant may be scheduled to lie on the bed to read or relax, while possibly listening to music, which may involve one or more of the following actions: turn On/Off the music system, take Up/Down the blinds, and turn On/Off the table lamp. In *LR*, a participant may be assigned to sit on the sofa to read, watch TV, or listen to music, which may involve one or more of the following actions: turn On/Off the TV, turn On/Off the music system, and take Up/Down the blinds. In *DR*, a participant may be assigned to have a meal or a coffee, while possibly listening to music, which may involve one or more of the following actions: turn On/Off the music system and take Up/Down the blinds. In *S*, a participant may be asked to read or use the laptop, which may involve either turning On/Off the table lamp or turning On/Off the laptop.

Once the set of activities are established, we define each routine by taking one of the activities and setting the *time slot* in which the activity is expected to take place, its *periodicity*, concerning the days when the activity is to be repeated, and its *frequency*, in terms of the specific weeks in which the activity ought to occur out of the total number of weeks of the span of the experiment.

Finally, we arrange the routines on a *schedule* that details what activities are to be followed per calendar day, with their corresponding time slot. The suggested duration for an activity session is between 3 and 5 minutes, with 3 activities being typically scheduled within a 30 minutes time slot (i.e., activities are not supposed to have a duration that exceeds 10 minutes). Figure 5.3 shows an excerpt of the schedule followed by the participants.

Overall, the schedule included a minimum of 12 activities per participant per day. Activities were scheduled for mornings and afternoons, from around 8 a.m. to around 5 p.m. The schedule spanned a total of 22 days over a period of 5 weeks. The routines considered include periods (i.e., time between the repetition of the activity per week) such as daily, every other day, and only one, two, or three days per week. In terms of frequency

(i.e., how often the activity is repeated over the weeks that span the experiment), routines are considered to happen every week or all weeks except for one, two, or up to three weeks. The total number of routines included by design in the schedule is 85. This is the number of base patterns that our approach is expected to find.

B.4 Metrics. In order to answer questions *ch5-lab-Q1* and *ch5-lab-Q2* we assess our approach using four metrics. Three of these metrics are standard in the evaluation of pattern discovery approaches, namely, precision, recall, and F_1-measure. In addition, we evaluate the robustness of our approach.

Precision P, is defined as the ratio of correct patterns detected out of all detected patterns (including incorrectly detected patterns). *Recall R*, on the other hand, is defined as the number of correctly detected patterns divided by the total number of base (input) patterns. *F_1-measure* is the harmonic mean between precision and recall, giving them both equal weight into a single performance measure:

$$F_1 = \frac{2 * P * R}{P + R} \tag{5.1}$$

We evaluate the *robustness* of our approach based on an existing evaluation framework for pattern mining [Gupta et al., 2008]. Here, robustness assesses the sensitivity of the approach to its input parameters. In our case, we examine the behavior of F_1-measure for a parameter space of minimum support *minsup*, minimum confidence *minconf*, minimum weak confidence *min-wconf*, and halt-time *ht* threshold. Specifically, *robustness* according to [Gupta et al., 2008] is evaluated by obtaining the mean and variance of the F_1-measure for the top k% combination of parameters, where the mean denotes the performance of the algorithm in terms of quality of the patterns and the variance denotes how sensitive it is to the selection of parameters. The ideal case is for the mean to be high and the variance low. However, in real-world applications one may consider acceptable to have consistent low variance values at the cost of pattern quality.

Our evaluation of robustness is twofold: first, we consider the mean and variance for the top 10% (i.e., $k = 10$) of combinations of parameters for which the values of F_1-measure are the highest, and second, we consider the mean and the variance for all combinations of parameters.

C Execution of the experiment

The experiment spanned a period of 22 weekdays, from December 15 to December 23 of 2016 and from January 16 to February 3 of 2017. Activities were mainly concentrated between 8 a.m. and 5 p.m. More than 100 thousand events from sensors and devices were gathered during the whole experiment, with an average of 5,679 events per day.

C.1 Selected dataset. For the purpose of evaluating our approach, from the event data of multi-sensors, we only consider events from the motion sensors.

Out of the total event data collected we excluded for the evaluation of our approach data reported by wall plugs *wp-12* (in *BR*) and *wp-21* (in *LR*). The reason for this was

that these devices were implemented for the sole purpose of responding to the motion sensors in their respective rooms in order to provide feedback of the continuous correct operation of the sensors. This was achieved by mirroring the value, True or False, of motion from the sensor to the value, "On" or "Off", of the wall plug.

In addition, we excluded from the evaluation event data from both window sensors deployed (i.e., *w-3* and *w-26*). The decision at this respect was taken, after the collection had concluded, when we realized through data analysis that the recorded data from both sensors was inconsistent and not reliable.

The event data excluded from consideration for evaluation is still present in the datasets we are making publicly available (see Section 5.4.4).

5.4.2 Synthetic data study

The *goal* of this study is to evaluate our approach under different cases which either are expected to emerge in real life or they test specific capabilities or sensitive aspects of the approach.

A Synthetic data generator

We built a data generator for this work. The programming language used for the implementation is Python. We design the generator based our experience with the lab study. In fact, the generator itself is meant to simulate the event data that the activity of real people would produce, as it occurred in the laboratory. Specifically, the data generator takes as input a schedule, with the same format as the one used in the lab study, to generate event data similar to the one that the activity of real people would cause devices and sensors to report.

We incorporated some specific features in our data generator aimed at simulating different scenarios that may occur in real life or that test a particular aspect of our approach. Specifically, these features are:

- Definition of routines for a particular time of the day and periodicity, and with specific frequency.
- Definition of activities with different duration, according to a predefined range,
- Definition of routines with 'correct' activities and 'incorrect' activities, specifying their particular chance of occurrence.
- Configurable motion sensor reporting rate. The default rate is of 1 triggering every 30 seconds on average.

For the generation of synthetic data we consider the same number of devices and rooms used in the lab study.

B Study design

B.1 Questions. To help us achieve the goal of this study, we propose to answer the following questions:

ch5-syn-Q1. What is the behavior of our approach under ideal conditions?

ch5-syn-Q2. How is the performance of our approach affected if the duration of the activities that form a routine vary widely?

ch5-syn-Q3. What is the impact that different deviations on the actions of the activities that form a routine have on the performance of our approach?

ch5-syn-Q4. How does the responsiveness of motion sensors affect the performance of our approach?

B.2 General description. To answer the questions of this study we design an experiment that consists of a series of trials and sub-trials that evaluate the performance of our approach systematically under various scenarios.

Base patterns. Activity routines defined by design in the schedule that serves as basis for the data generation.

Discovered patterns. Activity routines found by our approach from the event data produced by our data generator.

Rationale. We define a set of activity routines (i.e., *base patterns*) and arrange them in a schedule form. Our synthetic data generator takes the defined schedule and produces event data according to predefined parameters (see Section A). The event data generated is used as input to our approach. Finally, the set of activity routines discovered by our approach (i.e., *discovered patterns*) is compared against the base patterns.

B.3 Experimental trials. To answer the questions of this study we perform 4 different experimental trials, each focusing on one particular question. For this study we have considered longer time span compared to the other studies of our evaluation, in an attempt of taking advantage of the data generator. Overall, trials span a period between 3 and 6 months.

Trial 1. This trial focuses in evaluating the approach under ideal conditions The trial considers a total of 331 base patterns. The time span of the simulated routines for this trial is 3 months. The total number of events generated for the dataset of this trial is 176,917.

Trial 2. In this trial we examine the approach under variations on the duration of activities within the same routine. That is, a routine repeats the same activity but with widely different durations every time. In addition, we incorporate activities that occur in parallel, but only in different rooms. This trial encompasses 3 subtrials (tr2-1, tr2-2, tr2-3), each with different range of possible activity duration for routines, different base patterns, and different time span. Table 5.7 details information about the datasets generated for each of these subtrials in terms of time span (in

Table 5.2: Subtrials of Trial 2

Subtrial	Time span (mos.)	Base patterns (#)	Activity duration (s)	Events (#)
tr2-1	3	68	(1500, 2400)	158,613
tr2-2	4	60	(360, 3600)	230,827
tr2-3	5	177	(2400,3300)	1,144,395

Table 5.3: Subtrials of Trial 3

Subtrial	Time span (mos.)	Base patterns (#)	Activity duration (s)	Original schedule	Events (#)
tr3-1 and tr3p2-1	6	331	(340, 380)	tr1	350,917 and 351,745
tr3-2 and tr3p2-2	6	68	(1500, 2400)	tr2-1	316,233 and 316,614
tr3-3 and tr3p2-3	6	60	(360, 3600)	tr2-2	349,727 and 348,839
tr3-4 and tr3p2-4	6	177	(2400,3300)	tr2-3	1,366,426 and 1,365,762

months), number of base patterns, range of activity duration (in seconds), and number of events in the generated dataset.

Trial 3. This trial is mainly focused on evaluating the ability of our approach to deal with discrepancies in the actions of the activities that form a routine. The first one is aimed at generating event data that considers routines with a 70% chance of being performed 'correctly' (according to the base activity of the routine) and 30% chance of being performed 'incorrectly', where this 30% chance is evenly distributed among errors that in theory our approach can handle under default configuration, i.e., at most 1 missing action, 1 extra action, or 1 swap between two actions. This first part encompasses 4 subtrials (tr3-1, tr3-2, tr3-3, tr3-4), each using as schedule for generation a modified version of previous schedules used for trials 1 and 2. The second part of trial 3, which encompasses 4 subtrials (tr3p2-1, tr3p2-2, tr3p2-3,tr3p2-4) as well, considers trials similar to the first part, with the main distinction being the inclusion of an additional 10% chance of having an 'incorrect' activity within a routine, but in this case the erroneous activity includes more than the approach can handle under default configuration. Specifically, the 'error' is about two swaps of actions within an activity. Our intent in this case is to evaluate the impact that an 'error', which in theory our approach cannot handle, may have on the performance of the approach. Table 5.3 shows the information related to the datasets generated for the different subtrials of trial 3, in terms of time span (in months), number of base patterns, range of activity duration (in seconds), and number of events in the generated dataset.

Trial 4. This trial is aimed at evaluating the impact that variations on the reporting

Table 5.4: Subtrials of Trial 4

Subtrial	Time span (mos.)	Base patterns (#)	Activity duration (s)	Events (#)
tr4-1	6	331	(340,380)	1216,544
tr4-2	6	331	(340,380)	162,710
tr4-3	6	331	(340,380)	270,080
tr4-4	6	331	(340,380)	257,222

Table 5.5: Summary of number of base patterns and number of events per experimental trial of the synthetic data study

Trial No.	# base patterns	# datasets	Avg. time span (months)	# events	Avg. # events per day
1	331	1	3	176,917	1,966
2	305	3	4	1,533,835	4,261
3	538	4	5	3,697,746	5,869
4	todo	16	6	14,345,637	4,981
5	todo	19	6	7,901,884	2,310
Total		*43*	*5*	*27,656,019*	*3,747*

(or response) rate of the motion sensors may have on the performance of our approach. To place the focus only on this issue, we use as a basis the schedule of trial 1, which refer to an ideal case, and set our data generator to produce motion sensor events at different rates from the default rate of 30 seconds. We have 4 different subtrials (tr4-1, tr4-2, tr4-3, tr4-4), each with a dataset generated with a specific reporting rate of the motion sensors. Subtrial tr4-1 considers an average rate of 60 seconds for all motion sensors, subtrial tr4-2 an average rate of 90 seconds for all motion sensors, subtrial tr4-3 an average rate of 60 seconds for the motion sensors in *BR* and the default rate (i.e., 30 seconds) for the other three rooms, and subtrial considers an average rate of 90 seconds for *BR*, an average rate of 60 seconds for *LR*, and the default rate for the other two rooms. Table 5.4 shows the information related to the datasets generated for the different subtrials of trial 4, in terms of time span (in months), number of base patterns, range of activity duration (in seconds), and number of events in the generated dataset.

Table 5.5 presents a summary of the base patterns and number of events per experimental trial of the synthetic data study.

B.4 Metrics. To answer the questions of this study we consider the same metrics used for the lab study, namely, *precision*, *recall*, F_1-measure, and *robustness*. See Section 4.1.2–Metrics for the definitions.

5.4.3 In-the-wild study

The *goal* of this study is to evaluate our approach under real-world conditions, including a real home, deployment of commercially available smart devices and sensors, as well as real people going about their daily routines without any constraints.

A Study setup

A.1 Environment. This study was conducted in an apartment of approximately 82 m^2, with 2 bedrooms, 1 living room, 1 kitchen, and a bathroom. No smart home system or smart devices or sensors had been ever deployed in this apartment previously. A simplified floor plan of the apartment is shown in Figure 5.5. The only rooms that are considered in the study are marked in the floor plan, that is, 'parents' room (*PR*), M. room (*MR*), living room (*LR*) and kitchen (*K*). The 'parents' room (*PR*) has one TV bench in front of the bed, two tables facing each other near the window, and near the door of the room there is a wardrobe next to the bed and facing a bookcase. The M. room (*MR*) has one window with a desk to its right and a chest of drawers to its left. Next to the chest there is a bookcase, and near the door of the room there is a round table facing a wardrobe which is next to the bed. In the living room (*LR*) there is a TV bench near a large window which has a fish tank underneath, followed by a desk on the corner and a three-seat sofa and an armchair next to it. On the opposite side there is a showcase and a bookcase, and a piano is between the two doors of the room, one to the hallway and one to the balcony. The kitchen (*K*) has a dining table set with a table and four chairs next to the refrigerator, which is opposite to the kitchen cabinetry with a corner countertop.

The home appliances found in the apartment and considered for this study are as follows: from *PR* we consider 1 TV (*a-1*) and 1 laptop (*a-2*); from *MR*, 1 laptop (*a-3*) and 1 table lamp (*a-4*); from *LR* we consider 1 TV (*a-5*), 1 laptop (*a-6*), and 1 table lamp (*a-7*); from *K* we consider an electric kettle (*a-8*).

A.2 Participants. This study considers 7 participants (4 female and 3 male). The participants are 2 older adults (average age= 60.5), 4 young adults (average age= 27.25) and a 3-year-old child. Of these, 3 are the inhabitants (a young adult and two older adults) of the apartment considered, 3 are frequent family visitors, and 1 is a short-term guest. The event data collected for this study was gathered from all 7 persons. However, the interviews that helped to assess the approach were conducted only with home inhabitants.

A.3 Instrumentation. Similarly to the lab study, we used in this study different smart devices and sensors to obtain event data of the interaction between the participants and the home environment. Some devices and sensors are the same as in the lab study. These are: the wall plugs (Fibaro smart plug), the USB stick for creation and management of a Z-wave network (Aeotec Z-stick), and the multi-sensors (Aeotec Gen5 and Gen6). The window sensors were chosen to be different from the lab study (Aeotec 6[9] instead of Fibaro) following the issues we had in the lab (see Section A–Selected dataset). In this study no remote control was used. In a similar way as we did in the lab study, the USB

[9] https://aeotec.freshdesk.com/support/solutions/articles/6000161816-door-window-sensor-6-user-guide-

Figure 5.5: Floor plan of the apartment considered for the in-the-wild study

Z-wave stick was paired to all sensors and devices and to a computer that was set to continuously run our implementation for data collection and management.

A total of 8 wall plugs (*wp-2,10,11,13,14,15,16,27*) were connected between electrical outlets and specific electrical appliances. In *PR* the TV (*a-1*) was connected to *wp-15* and the laptop (*a-2*) to *wp-16*. In *MR* the laptop (*a-3*) was connected to *wp-27* and the table lamp (*a-4*) to *wp-14*. In *LR* the TV (*a-5*) was connected to *wp-13*, the laptop (*a-6*) to *wp-2*, and the table lamp (*a-7*) to *wp-11*. The kettle (*a-8*) in *S* was connected to *wp-10*.

A total of 6 multi-sensors were used, 1 in *PR*, 2 in *MR*, 2 in *LR*, and 1 in *K*. In *PR* multi-sensor *m-24* was placed in front of the bed to cover the space where most of the motion in the room occurs typically, according to the inhabitants (i.e., around the bed and tables near the window). In *MR* two multi-sensors were placed on opposites sides of the room, also considering with the room's inhabitant where the motion happens usually in the room. Thus, one multi-sensor (*m-23*) was placed on top of the chest of drawers near the window, pointing to the desk. The second multi-sensor (*m-26*) was placed on top of the table near the door, pointing to the wardrobe and the bed. In *LR* we also used two multi-sensors in opposite sides of the room: on top of the TV bench we placed multi-sensor *m-19* pointing mainly to the desk and partly to the sofa, and on top of the piano we placed multi-sensor *m-20* trying to cover the center of the room and the

Table 5.6: Sensors/devices by room used in the in-the-wild study

Room	Sensors/devices
Parents room (PR)	wall plugs: id-15 (TV), id-16 (computer)
	motion sensors: id-24
	window sensors: id-51
M. room (MR)	wall plugs: id-14 (table lamp), id-27 (computer)
	motion sensors: id-23, id-26
	window sensors: id-6
Living room (LR)	wall plugs: id-2 (computer), id-11 (table lamp), id-13 (TV)
	motion sensors: id-19, id-20
Kitchen (K)	wall plugs: id-10 (kettle)
	motion sensors: id-21

sofa and arm chair. In *K* we placed multi-sensor *m-22* to capture the activity of people around the dining set. Table 5.6 shows the relation between rooms and devices used for this study.

A.4 Data collection and management. We collected the events reported by the wall plugs and multi-sensors deployed in the apartment considered. In general the data collection and management for the in-the-wild study is similar to that of the lab study (see Section A–Data collection and management).

B Study design

B.1 Questions. We target to answer the following questions in order to achieve the goal of this study:

ch5-wild-Q1. What is the performance of our approach in the wild?

ch5-wild-Q2. What compromises have to be made in the wild compared to controlled environments?

Question *ch5-wild-Q1* is to be answered in terms of the activity routines found by our approach and the patterns that the occupants participating in the study recall to have actually performed. Question *ch5-wild-Q2* is meant to be answered once the evidence from the other two is in place.

B.2 General description. To answer *ch5-wild-Q1* we design an experiment that allows to measure the performance of our approach systematically, based on what the inhabitants participating in the study recall.

Base patterns. Compilation of the activity routines that the inhabitants recall to typically perform and have actually performed during the days that the experiment takes place. This information is gathered through interviews conducted with the three inhabitants before and during the experiment. Their feedback is compiled altogether into the set of based activity routines.

Discovered patterns. Activity routines discovered by our approach from the event data gathered.

Rationale. The participants perform their daily routines without any type of constraints, while the smart devices and sensors deployed report on the events related to motion and interaction with the various selected (controllable) objects around. The data collected during the days that the experiment is set to last is used as input to our approach. As last step, the set of activity routines found by our approach is compared against the base patterns.

The answer of *ch5-wild-Q2* is expected to come as an aftermath of conducting the experiment used to answer *ch5-wild-Q1*.

B.3 Interviews. We conduct interviews with the inhabitants participating in this study to help answering the questions presented above.

Before. This phase of interviews is set to take place prior to the start of the experiment. The goal is to ask to each of the inhabitants what routines they recall to typically follow in their daily routines, which they believe will continue to appear during the time the experiment is expected to be active.

During. In this phase, interviews are to corroborate or otherwise correct the answers that the inhabitants provide in the 'before' phase. The main goal is to learn from the inhabitants if the daily routines recalled in the previous phase have actually been performed and if any unexpected routine have emerged during the time the experiment have been active.

After. This phase of interviews is dedicated to present to the inhabitants any routines found by our approach but not recalled by themselves in the previous phases, to learn if it seems plausible that such routine occurred and what might be the possible reasons for the original omission.

The responses gathered during the 'before' and 'during' interviews are to be compiled in order to serve as the *ground truth* for the evaluation of our approach. In other words, the routines mentioned by the inhabitants during these interviews are meant to be the base routines against which the routines found by our approach from the event data are to be compared. The interviews conducted only include inhabitants and are only aimed at learning about the inhabitants' routines. This is because we consider in this study that the activity of visitors correspond to noise in the event dataset.

C Execution of the experiment

The experiment lasted for 49 days. Specifically, event data from smart devices and sensors was collected, twenty-four hours a day, seven days a week from around 10 in the morning of 10 June 2017 until around 3 in the afternoon of 28 July 2017. The total number of events in the data is over 2 million, with an average of 46,329 events per day. Prior to the experiment we took 2 days, in a deployment and testing phase, to ensure that all devices and sensors were reporting data correctly. The inhabitants of the apartment were involved during this phase to try to guarantee that the placement of the devices and sensors was as optimal as possible for the experiment, while avoiding inconvenience for the people at home.

C.1 Interviews. The 'before' phase of our interviews took place one day prior to the start of the experiment, in an in-person session of 20-30 minutes, with each of the three inhabitants. The weekly interviews of the 'during' phase were conducted every Saturday (first on June 17 and last one on July 22), during video call sessions of 10-15 minutes with one of the inhabitants (mostly, the young adult). The 'after' phase of the interviews took place with each of the three inhabitants, one day after the end of the experiment (i.e., Saturday 29 July 2017), during in-person sessions of 20-30 minutes.

5.4.4 Publicly Available Datasets

We are making the *datasets*[10] collected (in the lab and in-the-wild) and produced (synthetic data) during our studies publicly available. The data is made available complete, without any type of amendment. The datasets correspond to 9,778,671 events from smart devices and sensors, with 124,944 events collected in the lab, synthetic datasets containing 7,383,574 generated events, and the in-the-wild study dataset containing 2,270,153 events. We hope that these datasets serve for further research in the domain of pattern discovery for smart environments.

5.5 Results

Our results correspond to the evaluation of the performance of our approach under a wide variety of combination of parameters. Specifically, the values we tested for each parameter were:

$$minsup = \{\, 1.00, 0.6, 0.3, 0.1, 0.06, 0.03, 0.01, 0.006, 0.003, 0.001, 6.00 \times 10^{-4}, 3.00 \times 10^{-4}, 1.00 \times 10^{-4}, 6.00 \times 10^{-5}, 3.00 \times 10^{-5}, 1.00 \times 10^{-5}, 6.00 \times 10^{-6}, 3.00 \times 10^{-6}, 1.00 \times 10^{-6} \,\}$$

$$ht\ (s) = \{\, 210, 240, 270, 300, 330, \cdots, 900 \,\}$$

$$(minconf,\ min\text{-}wconf) = \{\, (1.0, 1.0), (1.0, 0.9), (1.0, 0.8), (1.0, 0.7), (1.0, 0.6), (0.8, 0.8), (0.8, 0.7), (0.8, 0.6), (0.8, 0.5), (0.8, 0.4), (0.6, 0.5), (0.6, 0.4), (0.6, 0.3)\ , (0.6, 0.2) \,\}$$

Our results on precision, recall, and F_1 were obtained for each of the combination

[10] https://doi.org/10.6084/m9.figshare.5572558.v1

Table 5.7: Results on *precision*, *recall*, and F_1 for the in the lab study

minconf	*min-wconf*	*ht* (s)	*P*	*R*	F_1
1.0	0.9	780	0.96	0.53	0.68
	0.8	720	0.93	0.60	0.73
0.8	0.7	750	0.95	0.80	0.87
	0.6	540	0.93	0.80	0.86
0.6	0.5	600	1.00	0.85	0.92
	0.4	600	0.95	0.88	0.91
	Aggregate		0.95	0.74	0.83

All results with $minsup = 0.001$

of parameter values above (e.g., one combination of parameter values is $minsup = 0.1$, $ht = 210$, $(minconf, min\text{-}wconf) = (0.8, 0.6)$).

In terms of run-time performance our approach can be said to be practical for a small number of combination of parameters. For instance, considering only one combination of parameter values, for the dataset of trial 1 of the synthetic data study, which contains nearly 180 thousand events, our implementation of the approach takes around 12 minutes to produce the final output on a single machine. And, for the dataset of the in-the-wild study, which contains more than 2 million events, the implementation takes around 28 minutes on a single machine. However, when considering all possible combination of parameter values above, the process becomes unpractical to be computed on a single machine. To cope with this, we parallelized the computation using the high performance computing platform of our university. The parallelization is configured to have one combination of parameters per job running in the cluster, with up to 80 jobs scheduled to run in parallel.

5.5.1 Lab results

Answer to ch5-lab-Q1. This question is about evaluating the performance of our approach in terms of discovered routines against based routines. Table 5.7 shows the results on precision, recall, and F_1 for the combination of parameters that we consider to be more significant to illustrate the overall performance of our approach. The results show that most of the base routines performed by the participants of this study are recovered by our approach. We can see that the overall value of the F_1-measure is of 0.83, with an aggregate recall rate of 74% and a maximum of 88%. The high values reported for the halt-time threshold *ht* (between 9 and 13 minutes) reflect that no much motion was reported during the activities performed by the participants. This is understandable, since most of the scripted activities required the participant to move only at the beginning and at the end of the session, while in the middle the participant was asked to perform actions that do not require much motion, such as watching TV, listening to music, reading, etc.

The variations on the number of routines discovered by our approach with respect to

the values of *minconf* and *min-wconf* depend on two factors:

1. The schedule followed by the participants includes some routines that are to be performed all weeks of the total number of weeks of the experiment (i.e., by design a confidence of 1.0), and some other routines to be performed only in some of the weeks (i.e., by design a confidence below 1.0)

2. The participants not always adhered to the schedule. In some occasions they missed an entire activity session or they mistakenly performed a different activity from the one scheduled. Other times they made mistakes on the actions scripted for the activity. This last case is alleviated by the ability of our approach to handle discrepancies (i.e., missing, extra, or swapped actions). However, the approach was run with a default tolerance to one discrepancy. Thus, anything beyond that could not have been discovered.

In summary, after analyzing both the data and our results, we can say that the routines that our approach failed to discover (10 out of 85), for the combination of parameters *minconf*= 0.6 and *min-wconf*= 0.4, were all caused by human error (failing to adhere to the schedule).

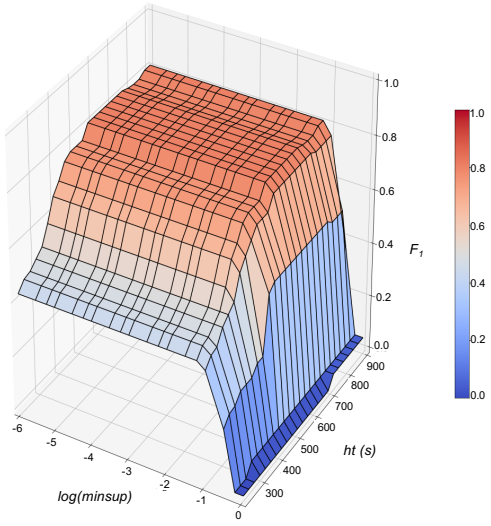
Answer to ch5-lab-Q2. This question is about assessing the robustness of our approach. Figure 5.6 shows three plots of the highest F_1-measure values for different values of *minconf*, specifically 1.0, 0.8, and 0.6. The variation on F_1 among the different plots reflects the fact that some routines were scheduled by design to happen all weeks, whereas some others were scheduled to occur on some of the weeks. Therefore, for the second case only a *minconf* equal to, or less than the confidence that those routines have by design would allow their discovery.

Additionally, all plots show that the value of *minsup* is not particularly relevant in obtaining a higher value of F_1, once we consider values below 0.1. On the other hand, the halt-time threshold *ht* seems to depend on the value chosen for *minconf*, particularly for lower values. This can be due to the fact that a lower *minconf* is more tolerant to errors.

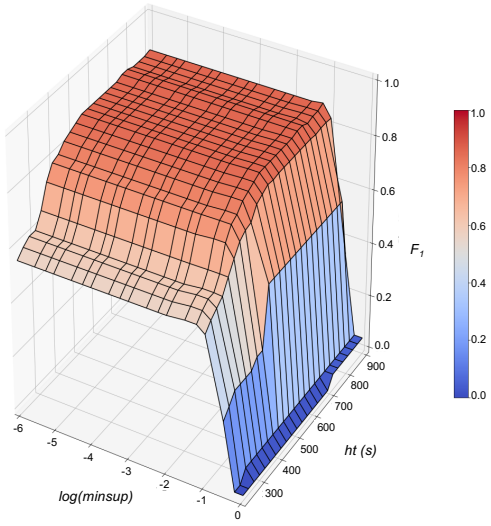
Overall our approach seems to be quite robust, based on the results obtained in this study. Table 5.8 shows the mean and variance for the top 10% value of F_1 and for all combination of parameters. Although the mean is considerably lower than the highest values, it still remains high, which means that the overall quality of the patterns discovered by our approach is good. Furthermore, the variance for both cases is quite low, which indicates that our approach is not markedly sensitive to different selection of parameters.

5.5.2 Synthetic-data results

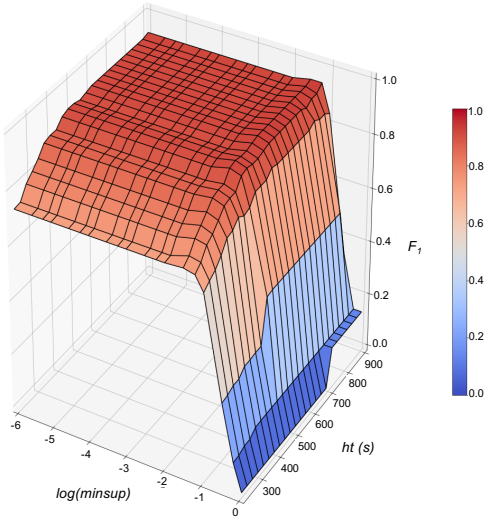
Answer to ch5-syn-Q1. This question is about evaluating the performance of our approach under ideal conditions, which is the goal of trial 1. The dataset generated for this trial was generated with a configuration that by design is expected to produce the highest possible recovery rates. Table 5.9 shows the results obtained for this trial,



(a) Top F_1-measure values for *minconf* = 1.0



(b) Top F_1-measure values for *minconf* = 0.8



(c) Top F_1-measure values for *minconf* = 0.6

Figure 5.6: Results on Robustness. Plots of the max. of F_1-measure for *minconf* = $\{0.6, 0.8, 1.0\}$

considering the most representative values for *minconf* and *min-wconf*. The table presents the value of *ht* for which the highest value of F_1 was obtained, for each of the combination of parameters considered. The two columns on the right are aimed at evaluating the robustness of the approach for this trial. As we can see, the aggregate of the highest values of F_1 is 1.00, which confirms that our approach has no issue in discovering all routines under ideal conditions. In terms of robustness we can see that when considering all possible combination of parameters, the mean of F_1 is considerably high and its variance maintains a low value, which means that under ideal conditions our approach is highly robust.

Answer to ch5-syn-Q2. This question, aligned with the goal of trial 2, is concerned with the evaluation of our approach when presented event data containing routines for

Table 5.8: Results on *robustness* for the in the lab study

		Top 10%		Overall	
conf	*w-conf*	*mean*(F_1)	*var*(F_1)	*mean*(F_1)	*var*(F_1)
1.0	0.9	0.67	4.76×10^{-5}	0.55	0.02
1.0	0.8	0.73	0.00	0.61	0.02
0.8	0.7	0.87	0.00	0.78	0.01
0.8	0.6	0.86	0.00	0.78	0.01
0.6	0.5	0.92	4.26×10^{-6}	0.88	4.07×10^{-3}
0.6	0.4	0.92	3.93×10^{-6}	0.88	3.37×10^{-3}
Aggregate		0.83	9.30×10^{-6}	0.75	0.01

Note: minsup = 0.001

Table 5.9: Results of Trial 1 on F_1, and *robustness*

minconf	*min-wconf*	*ht* (s)	F_1	mean(F_1)	var(F_1)
1.0	0.9	270	1.00	0.91	0.07
	0.8	270	1.00	0.91	0.06
0.8	0.7	210	1.00	0.93	0.06
	0.6	180	1.00	0.93	0.05
0.6	0.5	120	1.00	0.94	0.04
	0.4	120	1.00	0.94	0.04
	Aggregate		1.00	0.93	0.05

All results with *minsup* = 0.001

which the duration of activities vary widely.

Table 5.10 presents the results for the values of *minconf* and *min-wconf* that we consider more representative. When compared to the ideal case of trial 1, we can see that for the highest values of F_1 the drop is minimal, with an aggregate value of 0.99. The aspect that seems to be more affected by the variation on the duration of the activities of a routine is the robustness of the approach, where the mean of F_1 has decreased and the variance has increased. This is an indicator that under the case examined in this trial parameters have to be chosen more carefully than when we consider an ideal scenario.

Answer to syn-Q3. The question *syn-Q3* is about evaluating the impact that different deviations on the actions of the activities that form a routine have on the performance of our approach. The results of trial 3 help answering this question. Table 5.11 shows part of the results obtained for the first part of trial3, where only 'errors' that in theory our approach can handle are added by design.

The results for the first part of trial 3, show an aggregate value for F_1 of 0.80, which

Table 5.10: Results of Trial 2 on F_1 and *robustness* for each subtrial

Subtrial	*minconf*	*min-wconf*	*ht* (s)	F_1	all-mean(F_1)	all-var(F_1)
tr2-1	1.0	0.8	330	0.99	0.85	0.09
	0.8	0.6	210	0.98	0.86	0.08
	0.6	0.4	180	0.97	0.94	0.03
tr2-2	1.0	0.8	270	0.98	0.86	0.09
	0.8	0.6	210	1.00	0.90	0.07
	0.6	0.4	180	1.00	0.91	0.07
tr2-3	1.0	0.8	300	0.99	0.82	0.09
	0.8	0.6	210	0.99	0.88	0.09
	0.6	0.4	180	0.99	0.89	0.08
		Aggregate		0.99	0.88	0.08

All with *minsup* = 0.001, *robustness* considered all 30 values tested for *ht* (i.e., {30s ... 900s})

is considerably below the values obtained for trials 1 and 2. The main reason for this is not that our approach is unable to handle the 'errors' added by design, but rather that, as explained in Section B, we have designed this trial so that the 'correct' activity for the routine occurs around 70% of the times, whereas the 'incorrect' ones occur overall around 30% of the times. This means that even using a *min-wconf* threshold of 0.8 does not help in finding the 'correct' routines. If we focus only combinations where *minconf* and/or *min-wconf* are below 0.7, the values for F_1-measure are all greater than 0.9 for the best case. The aggregate variance shows that overall our approach is robust to the selection of parameters when errors that is able to handle are present in the data.

Table 5.12 shows the results for part 2 of trial 3, where an extra 'error', which our approach under default configuration is not supposed to be capable of handling, has been incorporated by design, with a 10% chance of occurrence. Leaving a 60% chance for the occurrence of a 'correct' activity and 40% for an 'error', manageable or not, to occur.

The drop in the values of F_1 between part 1 and part 2 of trial 3 is not entirely due to the incorporation of an error that in theory our approach is not able to handle. Instead, the slide happens because the extra 10% for an error translates into 10% less chance of finding the 'correct' activity, i.e., 60% as opposed to 70% in part 1. This is why the average difference between the highest value for F_1 in part 1 and part 2 with *minconf* equal to 1.0 or 0.8 and *min-wconf* equal to 0.8 or 0.6 is of around 0.10. In contrast, the average difference between the highest value for F_1 in part 1 and part 2 for *minconf* = 0.6 and *min-wconf* = 0.4 is of only 0.02. In other words, our results show that the impact of adding an extra discrepancy, beyond the ones that the approach is supposed to handle in theory, is not very pronounced. Furthermore, the mean and variance of F_1 are maintained around good levels (i.e., high and low, respectively), which reflects that the approach is robust for the case of trial 3, part 2.

Table 5.11: Results of Trial 3, part 1, on F_1 and *robustness* for each subtrial

Subtrial	*minconf*	*min-wconf*	*ht* (s)	F_1	all-mean(F_1)	all-var(F_1)
tr3-1	1.0	0.8	240	0.45	0.41	0.01
	0.8	0.6	210	0.96	0.91	0.03
	0.6	0.4	180	1.00	0.96	0.04
tr3-2	1.0	0.8	240	0.38	0.33	0.01
	0.8	0.6	270	0.91	0.79	0.08
	0.6	0.4	210	0.96	0.85	0.08
tr3-3	1.0	0.8	270	0.55	0.44	0.03
	0.8	0.6	210	0.96	0.86	0.07
	0.6	0.4	180	1.00	0.91	0.07
tr3-4	1.0	0.8	270	0.45	0.24	0.02
	0.8	0.6	270	0.95	0.80	0.08
	0.6	0.4	210	0.99	0.88	0.08
		Aggregate		0.80	0.70	0.05

All with *minsup* = 0.001, *robustness* considered all 30 values tested for *ht* (i.e., {30s ... 900s})

Table 5.12: Results of Trial 3, part 2, on F_1 and *robustness* for each subtrial

Subtrial	*minconf*	*min-wconf*	*ht* (s)	F_1	all-mean(F_1)	all-var(F_1)
tr3p2-1	1.0	0.8	210	0.36	0.34	0.00
	0.8	0.6	270	0.85	0.81	0.03
	0.6	0.4	180	0.99	0.95	0.04
tr3p2-2	1.0	0.8	330	0.28	0.24	0.01
	0.8	0.6	270	0.77	0.67	0.05
	0.6	0.4	210	0.92	0.81	0.08
tr3p2-3	1.0	0.8	270	0.48	0.37	0.02
	0.8	0.6	270	0.83	0.74	0.06
	0.6	0.4	300	0.98	0.89	0.07
tr3p2-4	1.0	0.8	300	0.37	0.18	0.01
	0.8	0.6	300	0.81	0.67	0.06
	0.6	0.4	270	0.99	0.86	0.08
		Aggregate		0.72	0.63	0.04

All with *minsup* = 0.001, *robustness* considered all 30 values tested for *ht* (i.e., {30s ... 900s})

Table 5.13: Results of Trial 4 on F_1 and *robustness* for each subtrial

Subtrial	*minconf*	*min-wconf*	*ht* (s)	F_1	all-mean(F_1)	all-var(F_1)
tr4-1	1.0	0.8	360	0.92	0.78	0.08
	0.8	0.6	240	1.00	0.89	0.08
	0.6	0.4	210	1.00	0.91	0.07
tr4-2	1.0	0.8	390	0.73	0.60	0.07
	0.8	0.6	330	0.99	0.85	0.10
	0.6	0.4	240	1.00	0.89	0.09
tr4-3	1.0	0.8	330	1.00	0.87	0.09
	0.8	0.6	210	1.00	0.91	0.08
	0.6	0.4	120	1.00	0.93	0.07
tr4-4	1.0	0.8	300	1.00	0.86	0.10
	0.8	0.6	180	1.00	0.90	0.08
	0.6	0.4	150	1.00	0.92	0.07
		Aggregate		0.97	0.86	0.08

All with *minsup* = 0.001, *robustness* considered all 30 values tested for *ht* (i.e., {30s ... 900s})

Answer to syn-Q4. This question, aligned with the goal of trial 4, is about evaluating the impact that the responsiveness or reporting rate of motion sensors may have on the performance of our approach. Table 5.13 shows the results obtained for this trial, considering the most representative values for *minconf* and *min-wconf*. The results show that the reporting rate of motion sensors has a clear impact on the performance of our approach. This is more apparent when looking at the mean of the values of F_1. As we can observe, subtrials *tr4-1* and *tr4-2*, which consider an average rate higher than the default for all rooms instead of for some specific ones, as in *tr4-3* and *tr4-4*, show lower values of F_1. Furthermore, the values of F_1 obtained for trial *tr4-2*, which considers an average motion reporting rate of 90 seconds, are lower that those obtained for trial *tr4-2*, which uses an average rate of 60 seconds.

5.5.3 In-the-wild results

For the case of this study we had to make an implementation modification to our approach in order to obtain an acceptable number of routines. The decision to do this was taken after we observed that by applying our approach directly to the event data collected, the output showed only routines for the kitchen. A careful inspection helped us to realize that while our approach was able to obtain routines in all rooms, they could only be considered frequent under the combinations of *minconf* and *min-wconf* used, if a broader definition of time slot was adopted. In other words, most routines could be said to be formed by activity sessions occurring in different nearby times (e.g., some activity sessions at 5 p.m. and others at 6 p.m., but all having the same exact sequence of actions). To cope

Table 5.14: Results of in-the-wild study, found routines compared to ground truth

Room	Found routines (#)	Matches (%)	Extra routines (#)
PR	24	73	2
MR	47	94	0
LR	13	59	2
K	18	95	3
Aggregate	102	80.25	7

All results with $minsup = 0.001$, $minconf = 0.6$, $min\text{-}wconf = 0.4$, and $ht = 510s$

with this issue we stopped considering time slot as part of the discovery phase of our approach, but we kept the information. Once the routines were discovered, we combined the information of time occurrence of the activity sessions of the routine to assign one of the following high-level times of the day accordingly: 'early morning' (from 5 a.m. to 8 a.m.), 'late morning' (from 8 a.m. to 12 p.m.), 'early afternoon' (from 12 p.m. to 3 p.m.), 'late afternoon' (from 4 p.m. to 5 p.m.), 'early evening' (from 5 p.m. to 7 p.m.), and 'late evening' (from 7 p.m. to 12 a.m.). The routines obtained using this implementation adjustment were the ones used for the evaluation of our approach in this study.

For this study we also run our approach considering all the combination of parameters mentioned at the beginning of Section 5.5. However, after analyzing the results obtained we chose the routines from only one of the combination of parameters for the evaluation against the responses gathered during our interviews. The combination of parameters chosen was decided based on what we considered to be the most representative results. Specifically, the combination of parameters chosen is $minconf = 0.6$, $min\text{-}wconf = 0.4$, $ht = 510s$, and $minsup = 0.001$.

Answer to ch5-wild-Q1. This question is about the assessment of the performance of our approach in terms of the number of base routines found. Table 5.14 shows the results of our approach in terms of the number of routines discovered per room, the percentage of the matches between the discovered routines and the base routines (established during the 'before' and 'during' interview phases), and the number of routines that our approach discovered, which were not part of the ground truth.

Our results are supported by what we could gather during the interviews about the overall activities happening in each room. The living room (*LR*) is mostly used by one of the older adults to watch TV, read, or use the laptop. These activities take place typically on early mornings and during evenings (early and late). This room rarely sees any activity between 9 a.m. and 5 p.m. All these factors explain the low performance observed for this room, since activities during the day are not many and typically involve not much motion from the inhabitant. The family always have meals in the kitchen (*K*), which is the room that sees the largest number of uniform routines around the apartment. This may explain the high percentage of matches between routines remembered by inhabitants and those found by our approach for this room. The room *MR* is only used by the young inhabitant.

This room sees a lot of activity at different times of the day, depending on a changing schedule imposed by activities the young adult performs outside home. This may explain the high number of discovered routines, which is a sign of a large number of routines happening in a not very homogenous manner. The room *PR* sees activity from both older adults, but predominantly from one of them (not the one that typically uses *LR*). This older adult stays at home all day some days of the week and other days goes to work at different times, depending on the day. We were able to confirm during the 'after' phase of the interviews that the extra routines found seem to correspond to typical times when the frequent visitors stop by.

Answer to ch5-wild-Q2. Question *ch5-wild-Q2* asks to recognized the compromises that have to be made to apply our approach in the wild, compared to controlled environments, such as the lab. The answer to this question has been hinted above, when we introduced the kind of modifications we had to make to our approach in order to obtain an acceptable number of the routines performed by the inhabitants. While the modification was minor and mostly related to an implementation detail, it helped us to observe that people's routines in real-life are discoverable, but mostly when broad times of the day are considered, instead of strict time slots.

5.6 Discussion

5.6.1 Limitations of the approach

One of the main limitations of our approach is the lack of a feedback mechanism that allows the discovery process and automatic production of control rules to evolve over time. This may include automatic tuning of parameters based on the quality of the patterns found, and the use of a reinforcement learning model based on the reaction of inhabitants to the suggested rules.

The detection of activity sessions of our approach is in general accurate but it is not able to handle activities overlapping in the same room. We believe that the use of overlapping time slots could alleviate this issue. On the other hand, while our approach is robust to routines for which activity sessions occur not at the exact same time, it allows only variations within a predefined time slot. A possible solution for this issue is, as we did during the in-the-wild study, to remove predefined time slots during the discovery of activity sessions and then combine the resulting sessions into routines based on higher-level, wider time slots (e.g., 'early morning', 'early evening').

5.6.2 Approach practicality

The robustness that we observed in our approach based on our evaluation is a good sign towards its applicability in real-world scenarios. However, several challenges can be foreseen before achieving end-user adoption of this technology. First, the performance of our approach highly depends on reliable reporting of smart devices and sensors. This, as we could observed during our lab and in-the-wild studies, requires proper installation, testing, and continuous monitoring, which is not something that one can

expect regular users to be willing or prepared to do. Our experience confirms previous works that have pointed out that there is still much room for improvement in terms of how much struggle users have to go through in order to understand and set up simple smart devices [Takayama, 2017]. One possible solution at this respect can be that service providers offer smart devices, sensors, and smart home systems capable of performing semi-supervised testing and monitoring with minimal user intervention, by sending data to be analyzed to the service provider, which can then react through, for instance, software updates.

Another issue towards the practicality of an approach like ours is the need for more people using behavior learning approaches in smart spaces against the lack of users' motivation to allow devices and sensors at home learning their behavior patterns. Although according to recent studies there is a growing interest in smart home technologies[11], it is still challenging to find people willing to have sensors deployed at home to learn their behavior patterns. We can say this, since out of at least 8 households to which we asked to participate in our study, only one decided to accept. Those that refuse to participate argue as main reasons privacy concerns and the inconvenience that the deployment would cause in their regular lives. Furthermore, the household that agreed to participate was reluctant in having a large amount of devices and sensors deployed at home, which made us to keep the number of devices deployed to a minimum that we considered useful for the purpose of the evaluation.

5.6.3 Ecological validity

Although in this work we have assessed our approach through an extensive evaluation, some issues can still be questioned in trying to generalize the findings to real-life scenarios.

In the lab study, the sequence of actions used for the scripted activities can be considered too simple. Additionally, overlapping activity sessions, occurring in the same room by different people, are not taken into account. Furthermore, the number of devices is small compared to the range of devices commercially available today. While the issue about scripted activities can be compensated to some extent through in-the-wild studies such as the one we conducted, the issue about the number and type of devices used deserves special attention in the lab, since it is an environment especially suited to test a wider variety of devices.

While the synthetic data study allowed us to test our approach under a wider range of cases, all of them were designed by hand with some random variations. In addition to this, it would be ideal to consider cases obtained from real human behavior data, from which synthetic event data could be generated.

In spite of the valuable results and insights we obtained from our in-the-wild study, it is clear that a study covering a large number of heterogeneous households would need to be conducted before being able to consider the approach as fully validated under real-life scenarios.

[11] http://www.bosch-presse.de/pressportal/de/media/dam_images/pi9349/survey_smart_home_overview_of_results_bosch_twitter_160830.pdf

5.7 Summary

In this chapter we provide our solution to *RQ2-b*. That is, we present a novel approach to discover periodic-frequent routines of people from event data collected from smart devices and sensors deployed at home.

The proposed approach is an extension of the existent sequence mining algorithm Generalized Sequence Pattern (GSP). Our algorithm is able to discover routines that are frequent in general and those that are only frequent for specific periodicities. Furthermore, it can find routines that were not always repeated by the inhabitants in exactly the same manner.

To assess our approach we conducted an extensive evaluation that we presented in this chapter, including an in the lab study, an in-the-wild study, and a study based on synthetic data. Our results show that our approach is capable of achieving a high recall-precision performance (~ 0.9) for the different scenarios considered.

Chapter 6

Continuous Identification of Users in Indoor Spaces

6.1 Introduction

In this chapter we give solution to one of the questions in which we have broken down our research question *RQ2*, that is, *RQ2-c*: How to automatically learn to recognize the identities of users in a continuous manner in smart environments?

The vision behind the approach we propose in this chapter is shown in Figure 6.1. In a preliminary stage, a learning model is build from the behavioral biometrics of users. Then, our approach expects as input behavioral biometrics in the form of time series from different unidentified users, which will be processed by the smart engine to ultimately assign each user to its corresponding identity.

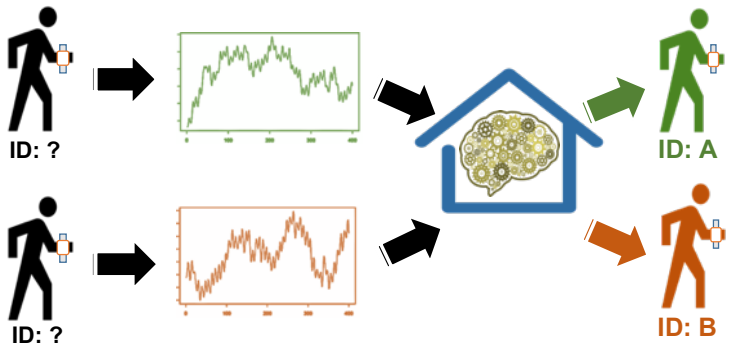


Figure 6.1: Continuous Identification Overview

In this chapter we present an approach that allows to identify people in a continuous manner during their daily routines based on hand and arm motion patterns. Specifically, we consider the frequency of the motion patterns over time by applying wavelet transform over the time series obtained from a wrist-worn inertial measurement unit (IMU). Different from state-of-the-art methods, our approach is not constrained to particular types of movements, gestures, or activities, thus allowing users to perform freely and unconstrained their daily routines while the identification takes place. Our approach is

targeted to home and office environments, where the number of people sharing the same space is not large.

To evaluate our approach, we conduct one in the lab study and two in-the-wild studies, one in home environment and one in office environment. Our evaluation involved a total of 29 different participants and the data collected corresponds to approximately 256 hours. The results obtained in the studies indicate that our approach is able to perform continuous user identification with a maximum accuracy of 0.88 for office environments and of 0.71 for the average size of a household.

The rest of the chapter is organized as we describe next. In Section 6.2 we present relevant related works on behavioral biometrics, continuous identification, and the specific tasks of user identification for smart environments. In Section 6.3 we introduce our approach, including some relevant background, the description of the feature extraction and selection processes of our approach, as well as its machine-learning classification stage. Section 6.4 provides details about the studies that constitute our evaluation methodology. In Section 6.5 we present our results in accordance with the questions we formulate to guide our evaluation. Then, in Section 6.6 we discuss on the validity of our evaluation, the limitations of our approach, and how practical would be for implementation in a real scenario. In Section 6.7 we provide a summary of the chapter.

6.2 State-of-the-art Approaches

6.2.1 Behavioral biometrics

Biometric technology has been extensively studied and applied for the authentication and identification of users. Biometric authentication and identification approaches have been classified into two groups [Sultana et al., 2014]: physical biometrics (e.g., fingerprint [Cappelli et al., 2006], iris scan [Sanderson and Erbetta, 2000], face recognition [Chia et al., 2015]) and behavioral biometrics. Behavioral biometrics are based on building feature profiles of users based on behavioral traits that appear over time [Bromme, 2003].

Behavioral biometrics approaches have been classified into five categories [Yampolskiy and Govindaraju, 2008], according to the kind of user's information employed. These are: based on authorship, based on direct human computer interaction (HCI), based on indirect HCI, based on skills and knowledge, and based on motion. Authorship-based biometrics focus on examining a piece of text or drawing produced by a person [Meng, 2012]. Approaches based on direct HCI focus on traits that arise during user's interaction with computers (e.g., interaction with keyboards [Rahman et al., 2013], with computer mice [Jorgensen and Yu, 2011], and haptics [Sae-Bae et al., 2012]). Indirect HCI-based approaches make use of events that can be obtained by monitoring user's interaction with computers through software [Yampolskiy, 2007a] (e.g., based on program execution behavior [Inoue, 2006], location-based authentication [Agadakos et al., 2016], and based on system calls [Toch et al., 2018]). Biometrics based on skills and knowledge challenge users with mentally demanding tasks to identify distinctive traits in their response behavior (e.g., based on memories [Das et al., 2013] and based on their own history knowledge [Ciria et al., 2014]).

Behavior biometric approaches based on motor-skills or motion focus on innate, unique, and stable body actions of users while performing a particular tasks [Yampolskiy, 2007b]. This last category includes approaches from other categories, such as direct HCI-based approaches based on, for instance, keystroke and mouse dynamics [Mondal and Bours, 2016], as well as different uses of haptics (e.g., button pressing [Pohl et al., 2015] or a thimble to interact with a virtual screen [Kanneh and Sakr, 2008]). The advent of wearable technologies have widened the spectrum of possible approaches in this direction, using motion from different parts of the body as distinctive trait (e.g., the motion of the head while listening music [Li et al., 2016], foot motion [Gafurov et al., 2011], and hips and legs motion [Frank et al., 2010]). One of the most studied approaches among motion-based biometrics focus on the gait of users, which has been typically captured by one of three methods: using video-processing techniques [Matovski et al., 2012], pressure-sensing floor [Bränzel et al., 2013], or wearable sensors (e.g., VR/AR headsets [Shen et al., 2018], wrist-worn sensors [Xu et al., 2017], and shoe-integrated sensors [Bamberg et al., 2008]). Recently, some approaches have proposed to use the radio frequency of Wi-Fi signals to capture gait patterns (e.g., [Shi et al., 2017], [Wang et al., 2016]).

Our approach is based mainly on arm and hand motion, captured through wrist-worn inertial sensors. However, it is aimed at working in a continuous manner, without equiring users to perform specific gestures, actions, or activities. Therefore, it handles altogether, without explicit cues, episodes that are typically considered separately by other approaches, such as walking, interacting with a keyboard or a mouse, or any other action or activity performed by the users as part of their daily routines.

6.2.2 Continuous user identification

Although some works in the literature have used these terms interchangeably [Guillén-Gámez et al., 2017], user authentication and user identification are widely seen as two different concepts [Marcel and Millán, 2007]. User authentication aims to accept or to reject a person claiming an identity, that is, comparing a biometric data to a pre-built template. On the other hand, the goal of user identification is to match the biometric data against all the records in a database.

A small number of approaches have been proposed claiming to perform continuous identification of users. Among these, we found an approach based on a multi-camera setup to capture the shape of people's bodies while they perform specific everyday actions or activities, such walking, running, jumping, waving one hand, eating, and drinking, all of which are labelled in a preliminary phase [Iosifidis et al., 2012]. Another method we found is based on images that are obtained from surveillance video streaming, where the newly processed images are matched against an existent database of images [Das et al., 2017]. This approach makes use of a manual labeling phase in which human annotators give identities (labels) to the images by comparing them with an existent gallery. A third approach claiming to address the problem of continuous person identification is a multi-modal method that combines speech and face recognition to provide humanoid robots with the ability to identify people during mutual interaction [Martinson et al., 2010]. According to the authors, the combination of the two biometrics is aimed at

reducing the moments in which the robot is unable to identify the speaker, either for being out of sight or remaining quiet.

Other relevant approaches also claiming to address the task of continuous identification are: one that identifies a user through a personal device that the user places on a tabletop surface [Ackad et al., 2012], one that is based on the monitoring of computer system properties and characteristics such as memory, CPU, and network data for the identification [Malatras et al., 2017], one that combines keystroke and mouse dynamics for user authentication and identification [Mondal and Bours, 2016], one in which the identification is based on capturing user-unique features through a wrist-worn sensor which are then transmitted through the user's body to a touch screen [Holz and Knaust, 2015], and an approach targeting identification for smart environments, which use a Kinect sensor to acquire video sequences from which face and anthropometric measurements (e.g., height, length of arms) are obtained as the features to identify users [Ferrara et al., 2014].

The related task of continuous authentication has received much more attention, with many approaches focusing on the authentication for smartphone devices (e.g., [Sim et al., 2007, Jakobsson et al., 2009, Shi et al., 2011b, Sitová et al., 2016]) mainly proposing multi-modal methods that combine different users' traits captured by the technology onboard (e.g., touchscreen, inertial sensors, localization). Other relevant approaches claiming to addressed the task of continuous authentication are based on keystroke characteristics [Roth et al., 2015], response to stimulation generated by the keyboard [Martinovic et al., 2017], gait patterns (e.g., [Lu et al., 2014, Schürmann et al., 2017]), ECG signals patterns [Louis et al., 2016], gestures [Burgbacher and Hinrichs, 2014], and breathing patterns [Chauhan et al., 2017].

In spite of their claims, the approaches mentioned above fail to be continuous, as they either require that the person performs a specific action or activity, which is not constantly performed by people throughout the day but only in specific moments, or they require a specific type of interaction with the system which would be impossible to maintain in a real-world setting without imposing restrictions on the activities or movements of the users. In contrast, the approach we propose in this paper does not require users to perform any type of specific gesture, actions, or activities in particular, but it works in a continuous manner regardless of the activities that users perform throughout the day in indoor environments such as home and office spaces.

6.2.3 User identification for smart environments

For the specific case of smart environments such as home or office, not many approaches have been proposed focusing on the identification of users. In fact, we found only one previous work aimed at addressing this task in a continuous manner: a multi-modal approach that combines face characteristics and anthropometric measurements, captured through a Kinect sensor, as the features used to identify the people at home [Ferrara et al., 2014]. In spite of the promising results presented by its authors, that approach was only evaluated while users were performing specific actions, instead of performing freely their daily routines at home.

Among other approaches that have focused on identification for home environments, although not in a continuous manner, some relevant works are: an approach that identifies users interacting with domestic networked devices from the traffic they generate [Brown et al., 2014], approaches that identify users based on the patterns they exhibit when interacting with electrical appliances [Garnier-Moiroux et al., 2013] or other objects found at home [Ranjan, 2015], identification methods that use the floor to capture step patterns of home inhabitants [Bränzel et al., 2013], a method for user identification based on smartphones [Alhamoud et al., 2014], and an approach based on WiFi signals to identify users while performing specific walking and stationary activities [Shi et al., 2017]. For the case of office environments, some relevant approaches that address user identification, however not continuous, are: an identification approach that uses WiFi signals to capture gait patterns from users [Wang et al., 2016], an approach that builds user profiles for identification based on the motions of opening and closing a refrigerator door and the pressure distribution of gripping the door-handle [Ishida et al., 2017], an approach that identifies users at the office using features related to button pressing in a coffee machine [Pohl et al., 2015], and a method that, using a ceiling-mounted depth camera on top of a table, provides person identification based on shoulder length, shape of the head, and posture of the back [Maekawa et al., 2016].

6.3 Approach Overview

Following our hypothesis, our approach is based on hand and arm motion patterns over time to build a profile of each user. To this end, the user is asked to wear a wristband IMU, from which accelerometer, gyroscope, and magnetometer data is collected in the form of time series of quaternions. We apply wavelet transform over the time series to obtain the observations that are used for the machine-learning classification.

6.3.1 Background

A IMU & Quaternions

Our approach is based on the use of a wrist-worn Inertial Measurement Unit (IMU). An IMU is an electronic device with an onboard processor that computes its own orientation by fusing the output of a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis magnetometer [Yang et al., 2015]. The orientation of the device is defined with respect to a 3D-world absolute coordinate system, where the x-axis points to the geographical east, the y-axis to the magnetic north, and the z-axis is perpendicular to the ground in the direction opposite to the earth's gravity. In this work we use for evaluation purposes the commercially available IMU Shimmer3 [shimmer, 2017], which cal yield its output in the form of quaternions.

Originally introduced by W.R. Hamilton [Hamilton, 1844], quaternions are mathematical objects that are convenient for representing orientations and rotations in three-dimensional space. A quaternion can be defined by a rotation axis and a rotation angle

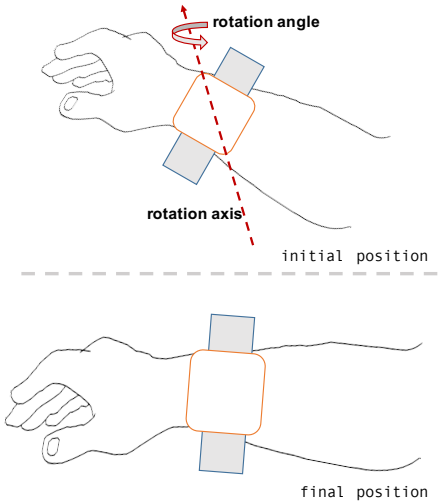


Figure 6.2: Arm with a wrist-worn IMU moving from an initial position into an end position, based on a quaternion (rotation axis, rotation angle).

as follows:

$$q = \cos\frac{\theta}{2} - r_x \sin\frac{\theta}{2} - r_y \sin\frac{\theta}{2} - r_z \sin\frac{\theta}{2} \tag{6.1}$$

where $[r_x, r_y, r_z]$ is the rotation axis vector and θ is the rotation angle. Figure 6.2 shows a representation of an arm wearing a wrist-worn IMU moving from an initial position (top) to an end position (bottom) based on a quaternion.

B Wavelet transform

Recalling our hypothesis (i.e., 'frequency of movements over time is a distinctive trait for identification'), two relevant approaches that are widely used for analyzing the frequency content of time series data are: Fourier transform [Barralon et al., 2006] and Wavelet Transform [Brajdic and Harle, 2013]. Fourier Transform, however, is known for introducing resolution issues, particularly when analyzing changes of frequency throughout time [Lester et al., 2009]. For a continuous analysis of a time series Wavelet Transform is more appropriate, as it can capture sudden changes in time [Nyan et al., 2006, Wang et al., 2012].

Wavelet Transform provides a time-frequency representation of a signal, offering optimal resolution both in the time and the frequency domains, as well as fine granularity in a multi-scale analysis [Abdelnasser et al., 2015]. For this work we have opted for the Discrete Wavelet Transform (DWT) over the Continuous Wavelet Transform (CWT). The main reason is that DWT is able to capture in a compressed manner the most relevant features of many natural signals, while CWT produces a redundant output which requires much larger computational resources [Rioul and Duhamel, 1992]. Previous works on identification and authentication have used DWT (e.g., [Hestbek et al., 2012]) and CWT

(e.g., ([Juefei-Xu et al., 2012]), focusing only, however, on the users' gait patterns.

The generic step of DWT splits the signal into two parts: high-frequency or detail coefficients and low-frequency or approximation coefficients. The splitting can be applied recursively a number of steps or levels to the approximation coefficients to obtain finer details from the signal [Mallat, 2008]. For a single level, which is what we use in this work, the DWT approximation coefficients α and detail coefficients β are computed using the following equations:

$$\alpha_k = \langle x_n, g_{n-2k} \rangle_n = \sum_{n \in \mathbb{Z}} x_n \; g_{n-2k} \tag{6.2}$$

$$\beta_k = \langle x_n, h_{n-2k} \rangle_n = \sum_{n \in \mathbb{Z}} x_n \; h_{n-2k} \tag{6.3}$$

6.3.2 Sensing arm and hand movements

In order to obtain the hand and arm movements data in a continuous manner, we use an Inertial Measurement Unit (IMU) placed on the wrist. For our case, the IMU is expected to operate with a 3-axis ultra low-noise accelerometer in combination with the gyroscope and the magnetometer. An ultra low-noise accelerometer is known to improve the clarity and resolution of the signal, allowing to sense very small vibrations [Zabit et al., 2011]. This is important in our case, since the continuous nature of our approach requires to operate even during people's idling episodes, when movements could be barely distinguishable.

6.3.3 Feature extraction

The data from the IMU is transformed into nine degrees of freedom (9DOF) quaternions, yielding four separate channels of time series which correspond to the components of the quaternions (x, y, z, w). For each of these channels, we reformulate the corresponding time series into 'examples', which are a particular type of encoding that span multiple events within individual segments of predefined size [Weiss and Hirsh, 1998, Lockhart and Weiss, 2014]. The use of 'examples' allows the application of conventional classification algorithms (e.g., k-NN, random forest (RF), decision trees), which cannot be used directly over time series data [Kwapisz et al., 2010].

Our reformulation into examples follows a two-step process. First we divide the time series data into segments of predefined size. We refer to this size as the *example size*. Then, we apply discrete wavelet transform (DWT) to the segment and form the 'example' from the coefficients of the DWT of all the channels.

The specific type of wavelet transform we use in this work is Haar transform, as it is easy to use and we found it appropriate for our type of data. To support our decision in favor or Haar wavelet transform we tested it against Fast Fourier Transform. Using 3 hours of data collected from 3 volunteers performing their daily routine at the office, we applied Fast Fourier Transform and Haar Wavelet Transform and compared the transformed time

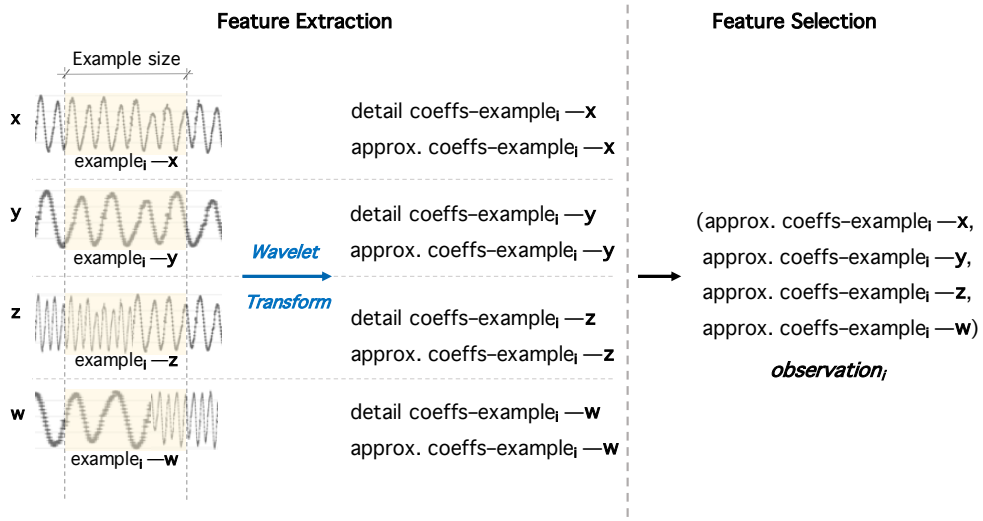


Figure 6.3: Feature extraction and selection. The extraction is made in two steps: dividing each time series into segments of predefined size and obtaining the wavelet transform coefficients from it. The selection leaves only approximation coefficients.

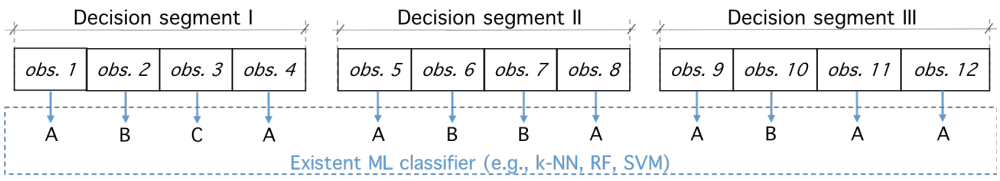
series across individuals. The results provided preliminary evidence that the frequency of movements could effectively be a distinctive trait among different people. Wavelet transform, however, yielded considerably better results as it was expected, due to its capability of capturing frequency changes throughout time [Brajdic and Harle, 2013]. The feature extraction process can be seen in the left-hand side of the schematic of Figure 6.3.

6.3.4 Feature selection

For feature selection we applied DWT over data from 12 volunteers performing their daily activity throughout 2 sessions of 3 hours each. The first session was used for training and the second session was used for validation. Using a k-NN method over different values of k, from 1 to 20, we compared the accuracy between considering all coefficients of the transform, only approximation coefficients, or only detail coefficients. We found that by considering only the approximation coefficients we attained the best accuracy. Thus, each of the observations, which are later used for classification, are formed by concatenating the approximation coefficients of the wavelet transform of the channels (x, y, z, w) of the quaternions output by the IMU (see right-hand side of Figure 6.3).

6.3.5 Classification

The final stage of our approach is to apply machine-learning classification on the observations previously obtained. We use one of the following existent classifiers as basis: k-nearest neighbors (k-NN) or Random Forest (RF). On top of the standard classification, we use a voting mechanism, which places particular importance on contiguity in terms of time of occurrence, with the aim of improving the overall performance of the multiclass classification [Tax and Duin, 2002]. For the voting mechanism, we first define constant-time length decision segments of consecutive observations. Then, all



Decision segment	Plurality-voting class	Majority-voting class
I	A	undecided
II	undecided	undecided
III	A	A

Figure 6.4: Machine-learning classification used to ultimately decide on the class of a set of observations. There are three options: *plurality* (any class with more votes wins), *majority* (the class with more than 50% of the votes wins), or *undecided* (no class satisfies any of the other decision rules).

observations within a decision segment are considered a vote for the class they have been previously assigned to by the classified. In the end, the final class with which all observations within a decision are labeled corresponds to the winner class of the voting process. If the voting yields a draw, we label the decision segment as 'undecided'. The use of 'undecided' is meant to favor a delayed decision based on a supporting identification method, instead of committing to a potential erroneous decision. Figure 6.4 illustrates the classification stage of our approach.

In this work we explore the performance of our approach under two voting strategies [Kuncheva, 2004]: plurality and majority. Majority voting is such where the final decision is taken following the class with 50% + 1 votes. Plurality voting is such in which the class with the most votes, regardless of having more or less than 50%, becomes the final decision.

To build our learning model we consider 3 sessions of data collected from users: one session is used for training, another one for cross-validation, and the last one for testing.

6.4 Evaluation Methodology

Instrumentation. To sense the hand and arm movements we used the commercially available inertial measurement unit (IMU) Shimmer3 [shimmer, 2017]. The Shimmer3 unit includes integrated nine-degrees-of-freedom (9DOF) inertial sensing via accelerometer, gyroscope, and magnetometer. The Shimmer3 unit offers two options for handling the data registered by its sensors: streaming of data via bluetooth to a host computer, or recording the data to its local storage, consisting of a micro SD card of 32 GB. In our studies we opted for the second alternative (i.e., local storage), after performing preliminary trial runs, in which we realized that the data streaming via bluetooth was not reliable, since it suffered from constant connectivity issues when moving to different rooms from where the host computer was placed, or if many objects were on the way between the unit and the host computer. Participants of the three studies were asked to wear the IMU on their wrist. In all cases the IMU device was placed on the dominant hand of the

participants, since we consider it to be the one that could experience more activity. The IMU device was configured with a sampling rate of 51.20 Hz and it was set to compute six-degrees-of-freedom (6DOF) and 9DOF quaternions from its integrated low-noise and wide-range accelerometers.

6.4.1 In-the-wild study at home

A Experimental setup

A.1 Environment. For this study the environment was the home of the participants. Overall, of the 8 participants, 2 are single persons living each in a studio apartment, while the rest are couples, each sharing a dwelling: a two-story house and two apartments.

A.2 Participants. For this study we had a total of 8 volunteers, 5 men and 3 women, with an average age of 44.5.

A.3 Data collection. For this study we collected an average of 9 hours of data per participant in three sessions of approximately 3 hours each. As in the rest of the studies, the datasets include timestamp and quaternions both for 6DOF and 9DOF, each from the wide-range accelerometer and the low-noise accelerometer of the IMU. No labels were added to the data, other than the headers indicating which feature is provided on each column.

B Study design

The objective of this study is to asses the performance of our approach on people performing their daily routine at home.

B.1 Questions. We use the following questions as the basis for this study:

ch6-home-Q1. What is the accuracy of the approach in identifying people who perform their daily routine at home without any constraints?

ch6-home-Q2. What is the impact that input parameters and configurations have on the performance of the approach when people perform their daily routines at home without restrictions?

B.2 General description. A key aspect of the study is to allow participants to move and act freely without any constraints. The data collection took place at the home of the participants, and we suggested to them to divide their participation into 3 sessions of 3 hours each, scheduled at their own convenience. For most participants data collection occurred in early evenings and, with few exceptions, it was evenly divided into three sessions. The Shimmer3 device was placed on the wrist of the participants and set on recording mode right before the start of the session and stopped after the scheduled session has finished. At the end of each session we asked the participant to describe roughly his/her activities. In particular, we were interested in knowing if the person

spent time idling or in activities involving minimal or no movement at all, since from such moments it is harder to obtain motion patterns. We found that a good part of the activities involved minimal movement, such as watching TV, listening to music, among others.

6.4.2 In-the-wild study at the office

A Experimental setup

A.1 Environment. The environment for this study corresponds to the office space where the participants work. Specifically, two office buildings are considered, with each participant performing her activities only in one of the buildings.

A.2 Participants. For this in-the-wild study, we counted with the participation of 20 volunteers, 17 men and 3 women, where the average age is 28.4.

A.3 Data collection. For this study, we collected an average of 9 hours of data per participant. The collection was roughly divided into 3 sessions of approximately 3 hours each. The datasets include timestamp and the components of quaternions, both for 6DOF and 9DOF, each from the wide-range accelerometer and the low-noise accelerometer of the IMU. No additional labels are added to the data.

B Study design

The objective of this study is to evaluate the performance of our approach in an office environment while people follow their regular activities.

B.1 Questions.

ch6-office-Q1. What is the accuracy of the approach in identifying people who perform their daily routine at their office without any constraints?

ch6-office-Q2. What is the impact that input parameters and configurations have on the performance of the approach when people perform their daily routines at their office without restrictions?

B.2 General description. No restrictions in terms of type of movements, actions, or activities were imposed over the participants and they were encouraged to simply act freely. The Shimmer3 device was set on recording mode once placed on the wrist of the participants at the beginning of each session, and stopped and removed once the session had concluded. As in the home environment, we also inquired participants about their general activities, to have a rough idea mainly concerning idling episodes. In this case, participants indicated that reading documents was the activity that could involve the smallest amount of movement, and mentioned writing, highlighting, typing, and interacting with their computer as the most common activities.

6.4.3 In the lab study

A Experimental setup

A.1 Environment. The environment for this study corresponds to the Internet of Things (IoT) laboratory of University of Luxembourg located at the Interdisciplinary Center for Security, Reliability and Trust (SnT). The IoT lab lab holds an approximate area of 44 m^2, arranged as to have four different spaces corresponding to a bedroom, a living room, a dining room, and a study. The prearranged activities of the lab study took place interacting with the bed at the bedroom, on top of the table in the dining room, and using the bookshelf at the study.

A.2 Participants. The participants of this study are 7 volunteers, 6 men and 1 woman, with 30.1 as the average age.

A.3 Data collection. From the in the lab study we collected an average of 30 minutes of data per participant. The dataset gathered includes timestamp and quaternions both for 6DOF and 9DOF, each from the wide-range accelerometer and the low-noise accelerometer of the IMU. No additional labels, other than the headers indicating which feature is provided on each column, are added to the data.

B Study design

The main objective of lab study is to evaluate the performance of the approach when people follow the same set activities, and when the training data is considerably short

B.1 Questions. We target to answer the following questions in order to achieve the goal of this study:

ch6-lab-Q1. What is the accuracy of the approach in identifying people who follow all the same set of short activities?

ch6-lab-Q2. What is the impact that input parameters and configurations have on the performance of the approach when people perform the same set of activities and the amount of available data is small?

B.2 General description. Each participant was involved in 3 separate sessions in which he/she performed the same sequence of 4 activities. The 4 activities were designed to be completed all in about 10 minutes. The sessions were scheduled at the convenience of the participants, with 4 participants having all sessions performed on the same day, 2 having them each on a different day, and the rest having two sessions on the same day and one session on a different day. All objects related to each activity were set on the corresponding initial state prior to the start of the session. The Shimmer3 device was placed on the wrist of the participants and set on recording mode right before the start of the session and stopped only once they have completed all 4 activities. The activities are described below and illustrated in Figure 6.5.

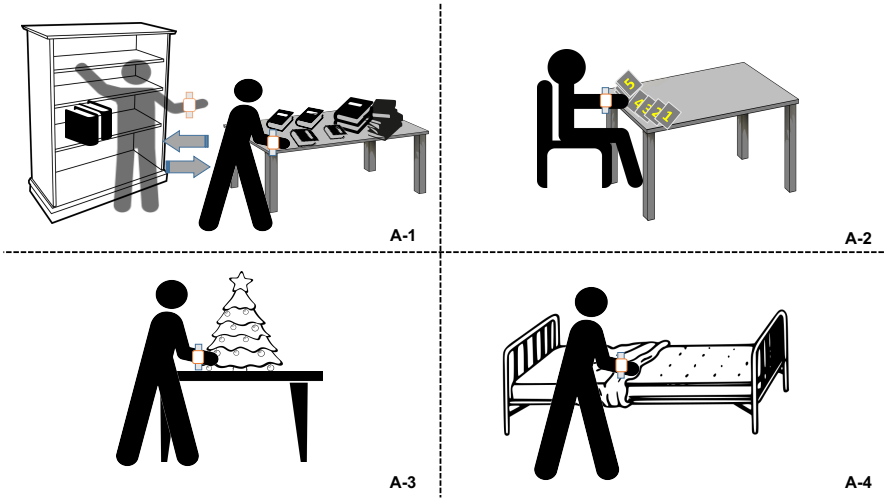


Figure 6.5: Activities for the in the lab study (User Identification). Act-1: Arrange books in a bookshelf. Act-2: Ordered a deck of numbered cards. Act-3: Decorate a tabletop Christmas tree. Act-4: Make up a bed.

Act-1. *Arrange books on a bookshelf.* The participant found 20 books disperse on the top of a desk. From this initial state, the participant had to arrange the books in a bookshelf next to the desk in the way he/she deemed reasonable.

Act-2. *Order a deck of numbered cards.* The participant found on a table a deck of 50 cards which had been previously shuffled thoroughly. From the initial state, the participant had to put the cards in order (ascending or descending), as he/she deemed convenient.

Act-3. *Decorate a tabletop Christmas tree.* The participant found on a table a tabletop Christmas tree, and a box containing decorations such as baubles, tinsels, and strings of lights. From the initial state, the participant had to decorate the tree as he/she deemed reasonable, using all the decorations found in the box.

Act-4. *Make up a bed.* The participant found a single bed unmade, with one bed sheet, one quilt, and one pillowcase folded out on top of a pillow placed on the foot of the bed. From the initial state, the participant had to make up the bed using the bedclothes found on the bed.

The activities described above were chosen as to be different in nature and diverse in the movements involved. Thus, some activities involve being standing (e.g., Act-1,4) and some being sitting (e.g., Act-2), some involve long arm movements (e.g., Act-4) and some short and fast movements (e.g., Act-2). The activities Act-2 and Act-3 took place in the same room within the IoT lab, however we observe that all participants deemed more convenient to sit while performing Act-2 and stand during Act-3. The activities Act-1 and Act-4 happened in a different room with respect to each other and Act-2 and Act-3. Therefore, the participants had to walk from one room to the other as part of each

session, which means that the data collected in this case includes also the motion related to the displacement of the participant between rooms.

6.4.4 Publicly Available Datasets

We are making the *datasets*[1] gathered during our studies publicly available through the online digital repository Figshare. The data is made available complete, as we collected it and used it, with the exception of data from 6 participants, who expressly requested not to share their data. Thus, the datasets made publicly available correspond to approximately 205 hours of data. We hope that these datasets serve for further research in the domain of continuous identification and authentication based on wrist-worn inertial sensing.

6.5 Results

6.5.1 In-the-wild at home results

A Answer to ch6-home-Q1

Based on the total number of participants of the in-the-wild study at home ($n = 8$), the combinations $\binom{n}{k}$ considered in this case are 28, 50, 50, 50, 28, 8, and 1 combinations, respectively, for each of the population sizes k , i.e., $\{2, 3, \ldots, 8\}$.

Figure 6.6 shows the results obtained, in the best case (red line) and on average (rest), for the population sizes for which the accuracy is not less than 0.5. We can observe that for the range of average size of a household, i.e., 2.3 to 4.0 members[8], the accuracy of our approach is, for the best case, between 0.79 and 0.63, with a mean of 0.71. Specifically, the best case is obtained with 100 points as example size, plurality as voting mechanism, and 90 seconds as decision time.

B Answer to ch6-home-Q2

B.1 Example sizes & voting mechanisms. We tested our approach against the four different cases of the combinations between example sizes (100 and 500 points) and voting mechanisms (plurality and majority). The results obtained on average for all decision times considered (i.e., 30, 60, and 90 seconds), for each of the four cases are shown in Figure 6.6.

The average results show 100 points with plurality voting as the best case, followed by 500 points with majority and 500 points with plurality, leaving 100 points with majority voting as the case with the lowest average accuracy. However, the difference between the accuracy values of any of the cases considered is not statistically significant[9] with respect to the rest of the cases.

[1] https://doi.org/10.6084/m9.figshare.c.4058669.v1

[8] The average household size for the 28 countries of the European Union is of 2.3 members [European Commission (Eurostat), 2017], for the 32 countries of the OECD is of 2.63 members [Organisation for Economic Co-operation and Development (OECD), 2016], and for 202 countries worldwide is of 4.0 members [United Nations (UN), 2017].

[9] *t-test* ($p < 0.05$), used herein for all cases when we test the statistical significance of the difference between two sets of values, unless we specified otherwise.

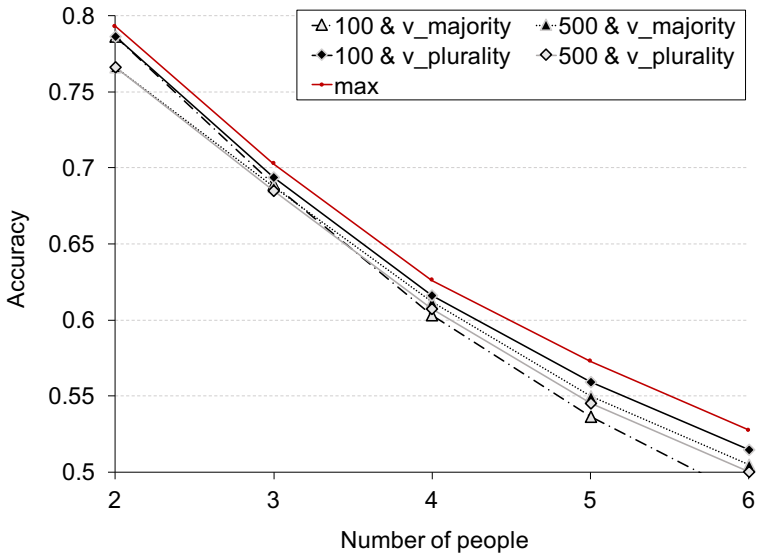


Figure 6.6: Accuracy — Home environment (User Identification).

B.2 Decision times. Overall longer decision times yielded higher accuracy values (see Figure 6.7). However, the difference between the accuracy values obtained with any two distinct decision times is not statistically significant. In addition to the main decision times considered, we also ran additional trials with decision time as short as 10 s and as long as 300 s (only for the case of 100 points as example size, plurality voting, and up to 10 combinations per population size). The results in the case of 10 s for decision time showed a loss in accuracy of 0.21 on average for the home environment and of 0.20 for the office environment, which in both cases is statistically significant. On the other hand, for the case of 300 s compared to the best case found for 90 s, the improvement was of 0.03 for the home environment and of 0.04 for the office environment, not statistically significant in either case. Therefore, we can say that, while a considerable decrease in the decision time with respect to the main range considered (i.e., 30 s to 90 s) may impact significantly the accuracy of our approach, an increase beyond 90 s will not yield significantly better results.

B.3 Undecided. we analyze results on two aspects: 1) the ratio between undecided observations and the total number of observations not classified to the correct identity, and 2) the total error (including undecided), i.e., all misclassified observations, against the misclassified observations minus the undecided. As mentioned in Section 6.3, considering an undecided class may allow to avoid misclassification in some cases, resorting to supporting methods to reach a final decision.

Figure 6.8 shows the results for the home environment on the ratio between undecided and total number of observations not classified to the correct identity. The values presented correspond to the example size that yielded the highest ratio, considering the average obtained from the results of all decision time values (i.e., 30, 60, and 90 seconds). For plurality the best ratio was obtained with 500 points as example size, whereas for

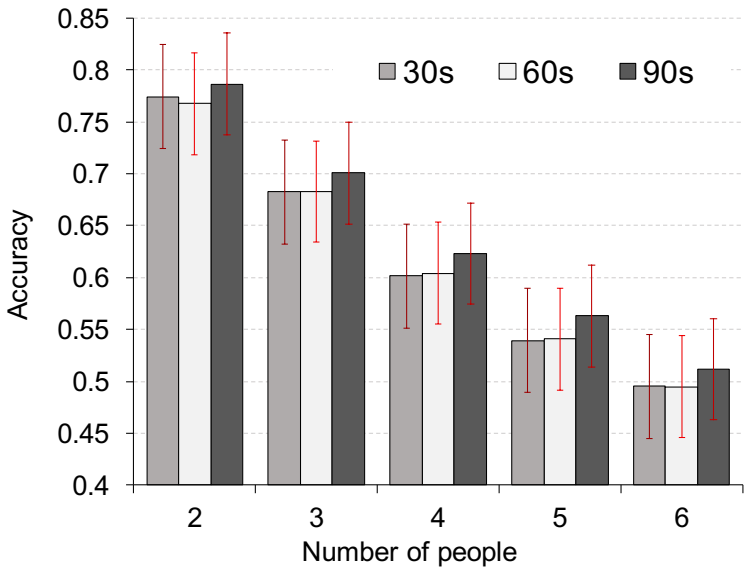


Figure 6.7: Decision times — Home environment (User Identification).

majority the best case was obtained considering 100 points. As we can see, the proportion of undecided observations grows as more people is considered for the identification, reaching nearly 18 % with respect to the total number of observations not correctly identified. Overall the ratio is higher for majority voting than for plurality. For majority the value of the ratio grows from 1 % at a population size of 2, to 17.9 % at a population size of 6, while for plurality the value of the ratio grows from 6 % for 2 people to 12 % for 6 people. The difference between these two cases, however, is not statistically significant.

In Figure 6.9 we contrast, for the home environment and each of the two voting mechanisms considered, the average number of misclassified for all decision time values on the best case overall (i.e., example size and voting mechanism that yield the lowest number of misclassifications), against the average number of misclassified for all decision time values minus the undecided, for the best case in terms of undecided. This can help to visualize the possibility of avoiding misclassification by reaching the decision on the undecided using some extra supporting method. Specifically, the average overall case is obtained, as we have seen, using 100 points and plurality, while the average best case for plurality is 500 points and for majority is 100 points. As we can see, although for a population size of 2 there is no positive difference in considering undecided for plurality and very little (0.002) for majority voting, the larger the population, the larger the amount of undecided, reaching a reduction on the total error of up to 0.09 for the best case (majority) at a population size of 6. For this case, the difference between the results using majority voting and using plurality is not statistically significant.

6.5.2 In-the-wild at the office results

A Answer to ch6-office-Q1

This question is about the accuracy of our approach in identifying users in an office environment while they perform their regular activities without restrictions.

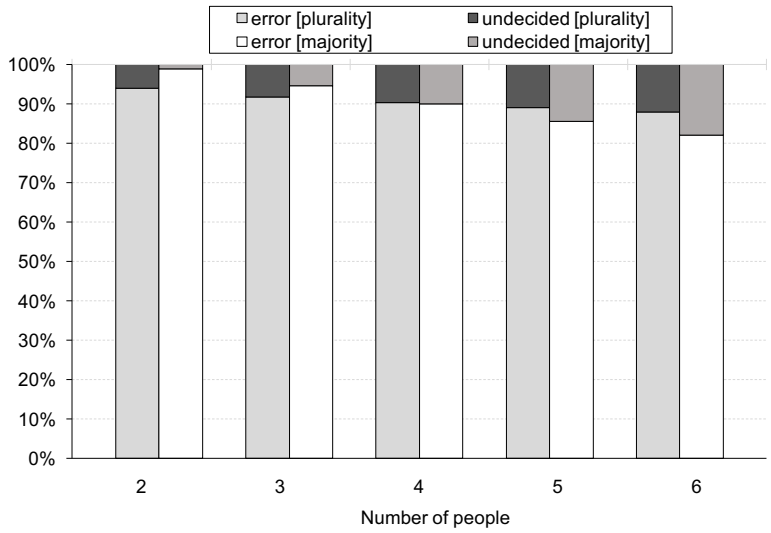


Figure 6.8: Ratio between undecided and total misses — Home environment (User Identification).

Based on the number of participants in this study (i.e., $n = 20$) and having a limit of up to 50 combinations per population size (as we mentioned at the beginning of this section), we have considered 50 combinations of people for each of the population sizes, except for 19 and 20 for which we considered the total number of combinations, i.e., 20 and 1, respectively.

Figure 6.10 shows the accuracy obtained in the best case (red line) and other average specific cases, for the population sizes for which the accuracy is not less than 0.5. The maximum accuracy reached is of nearly 0.88 for the two-people combinations tested, using 100 points as example size, plurality as voting mechanism, and 90 seconds as decision time. The average accuracy for the combinations of all population sizes depicted in the figure (i.e., 2 to 15) is of 0.68, holding an accuracy over 0.6 even when considering a population size of 12.

B Answer to ch6-office-Q2

B.1 Example sizes & voting mechanisms. The results obtained on average for all decision times considered (i.e., 30, 60, and 90 seconds), for each of the four cases are shown in Figure 6.10.

The average results show 100 points with plurality voting as the best case, followed by 500 points with majority and 500 points with plurality, leaving 100 points with majority voting as the case with the lowest average accuracy. However, the difference between the accuracy values of any of the cases considered is not statistically significant[9] with respect to the rest of the cases.

[9] *t-test* ($p < 0.05$), used herein for all cases when we test the statistical significance of the difference between two sets of values, unless we specified otherwise.

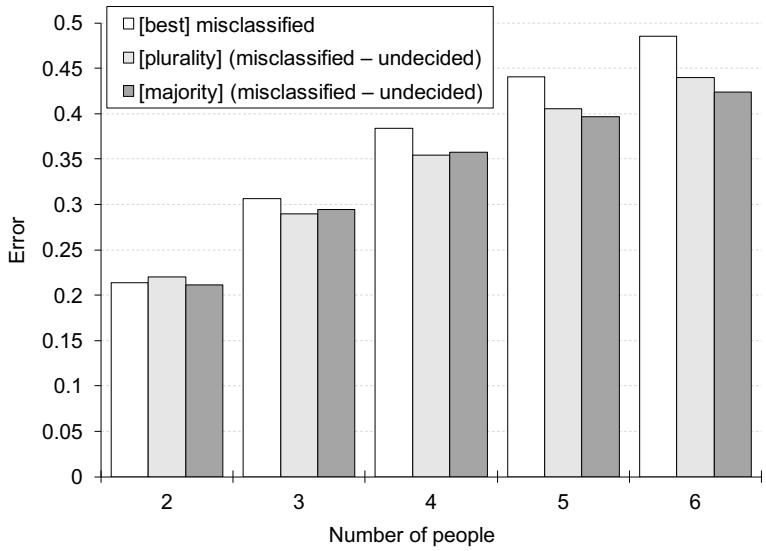


Figure 6.9: Error considering undecided in favor of accuracy — Home environment (User Identification).

B.2 Decision times. The results at this respect are shown in Figure 6.11. It can be seen that longer decision times yielded higher accuracy values. However, the difference between the accuracy values obtained with any two distinct decision times is not statistically significant.

In addition to the main decision times considered, we also ran additional trials with decision time as short as 10 s and as long as 300 s (only for the case of 100 points as example size, plurality voting, and up to 10 combinations per population size). The results in the case of 10 s for decision time showed a loss in accuracy of 0.21 on average for the home environment and of 0.20 for the office environment, which in both cases is statistically significant. On the other hand, for the case of 300 s compared to the best case found for 90 s, the improvement was of 0.04, not statistically significant however. Therefore, we can say that, while a considerable decrease in the decision time with respect to the main range considered (i.e., 30 s to 90 s) may impact significantly the accuracy of our approach, an increase beyond 90 s will not yield significantly better results.

B.3 Undecided. The results for the office environment on the ratio between undecided and total number of observations misclassified are shown in Figure 6.12. The proportion of undecided increases as the population size grows, similar to what we observed in the home environment, but in a more pronounced manner. Results considering plurality start at 8 % for 2 people, to then increase steadily, registering 14 % for 6 people and continue until reaching 22.6 % for a population size of 15 people. Majority voting shows a more drastic growth than plurality. Thus, although for a population size of 2 the proportion of undecided is of only 2 %, it increases importantly until reaching 20 % for 6 people and 45 % at a population size of 15. In this case, the difference between plurality and majority results is statistically significant.

Also on the office environment, Figure 6.13 presents a comparison between the

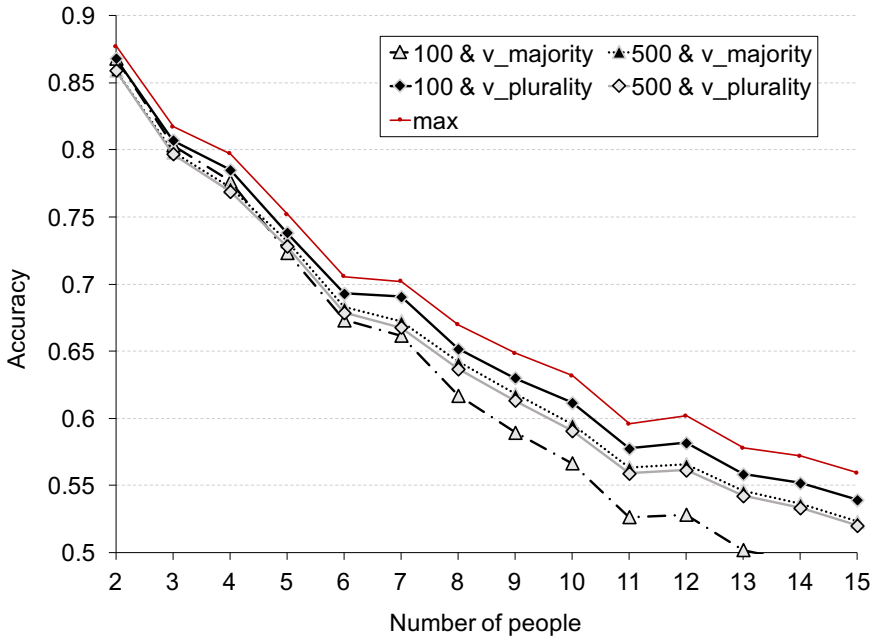


Figure 6.10: Accuracy — Office environment (User Identification).

average number of misclassified for all decision time values in the best case overall (i.e., 100 points and plurality), and the average number of misclassified for all decision time values minus the undecided for the best case in terms of undecided, considering plurality (i.e., 500 points) and majority (i.e., 100 points). As we can observe, for both plurality and majority the error is reduced considerably by subtracting undecided, as the population size increases. Thus, initially for 2 people the error is barely reduced by 0.003 for plurality and 0.002 for majority, going as far as to reach a reduction of 0.09 for plurality and of 0.17 for majority at a population size of 15. Overall, majority voting yields more undecided than plurality, with the difference between the two sets of values being statistically significant. Furthermore, for majority voting the set of values obtained by subtracting the undecided from the total number of observations misclassified is statistically significantly lower with respect to the case that considers all the misclassified. Situation that is not true for plurality voting.

6.5.3 Lab results

A Answer to ch6-lab-Q1

This question is related to the accuracy of our approach for prearranged activities taking place in the laboratory. Given the number of participants of the study (i.e., $n = 7$), the combinations $\binom{n}{k}$ considered for each population size value k (i.e., $\{2, 3, \ldots, 7\}$) are 21, 35, 35, 21, 7, and 1 combinations. The accuracy results obtained for the combinations of each population size are shown in Figure 6.14. The red line depicts the best case, in terms of decision time, example size, and voting mechanism, while the rest of the lines correspond to the average results for each of the possible configurations by combining either 100 or 500 points as example size with plurality or majority as voting mechanism.

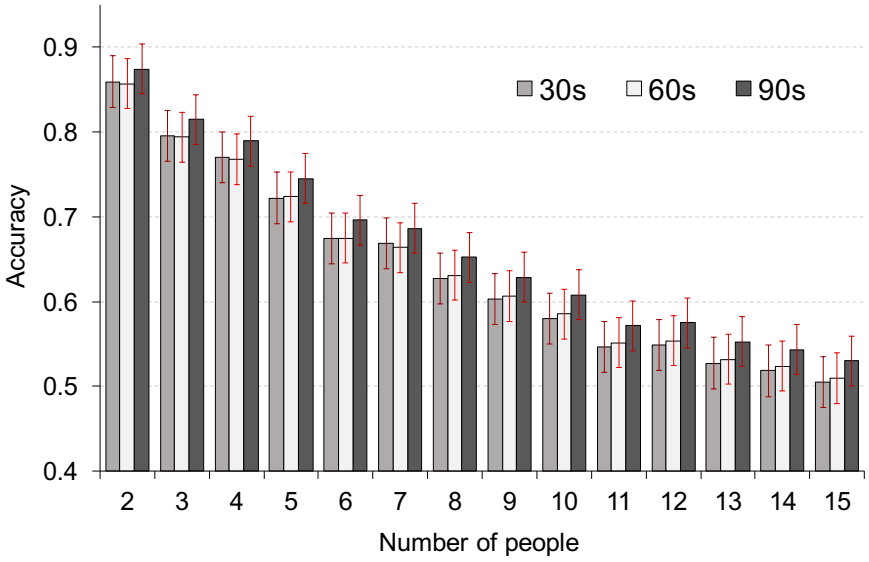


Figure 6.11: Decision times — Office environment (User Identification).

B Answer to ch6-lab-Q2

B.1 Example size & voting mechanism. The maximum accuracy value obtained is of 0.85 for a population size of 2, considering 100 points as example size, plurality voting, and 90 s as decision time. Furthermore, the average maximum accuracy for the combinations considered of all population size values is of 0.68. As we consider for the lab study home-like (prearranged) activities, we compare these results to the ones obtained in the in-the-wild study at home. We can see that for the average size of a household, i.e., between 2.3 and 4.0 members (see Section 6.5.1), the accuracy for the best case is between 0.85 and nearly 0.70, with an average improvement of 0.06 over the results obtained in the home environment. However, the difference between the two sets of results is not statistically significant.

In general the longer the decision time the higher accuracy obtained (see Figure 6.15). However, for the decision times considered (30, 60 and 90 s), the difference between the accuracy values obtained with any two distinct decision times is not statistically significant.

B.2 Undecided. The results for the lab study on the ratio between undecided and the total number of observations not classified correctly are shown in Figure 6.16. Similar to what was presented for the other environments, the values that are depicted in the figure correspond to the example size that produced the highest ratio, taking the average of the results of all the decision time values of our evaluation (i.e., 30, 60, and 90 seconds). The highest ratio of undecided for plurality is obtained with an example size of 500 points, while for majority voting the highest ratio is obtained with 100 points. Overall, the ratio is higher for majority than for plurality. The ratio for plurality goes from 10 % at a population size of 2 to 15 % at a population size of 7. On the other hand, the ratio when using majority grows more drastically from only 2 % at a population size of 2 to 31 % at a

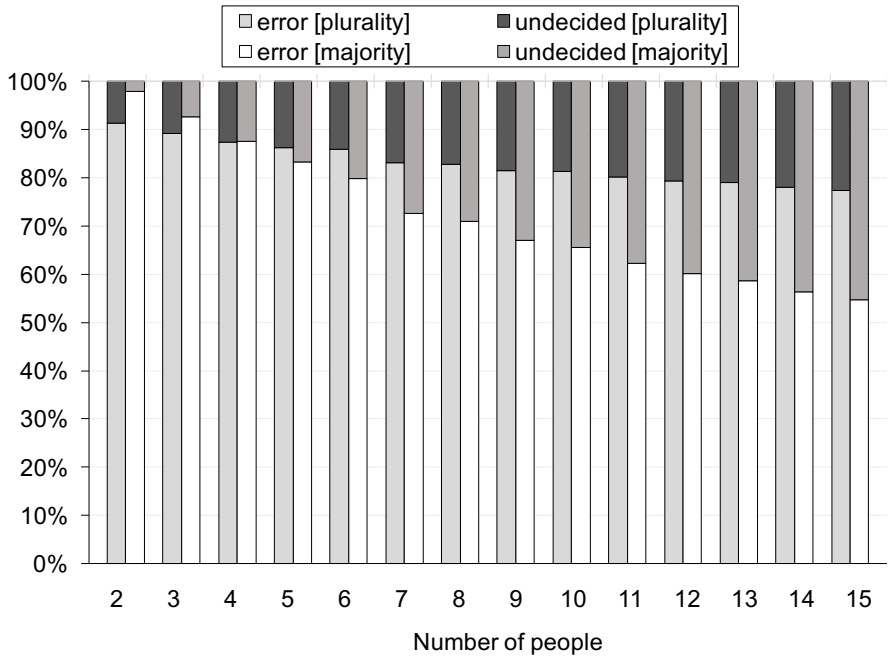


Figure 6.12: Ratio between undecided and total misses — Office environment (User Identification).

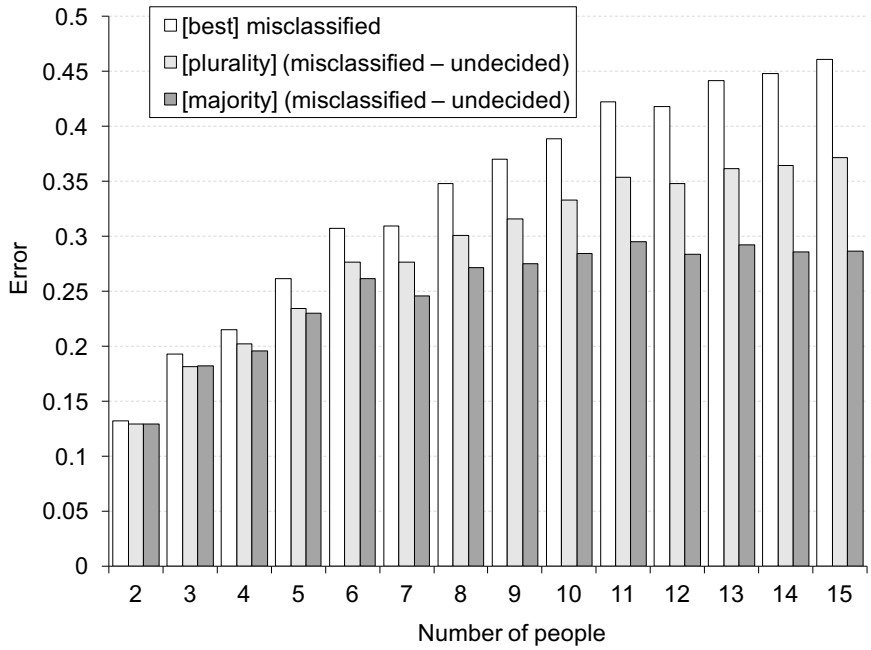


Figure 6.13: Error considering undecided in favor of accuracy — Office environment (User Identification).

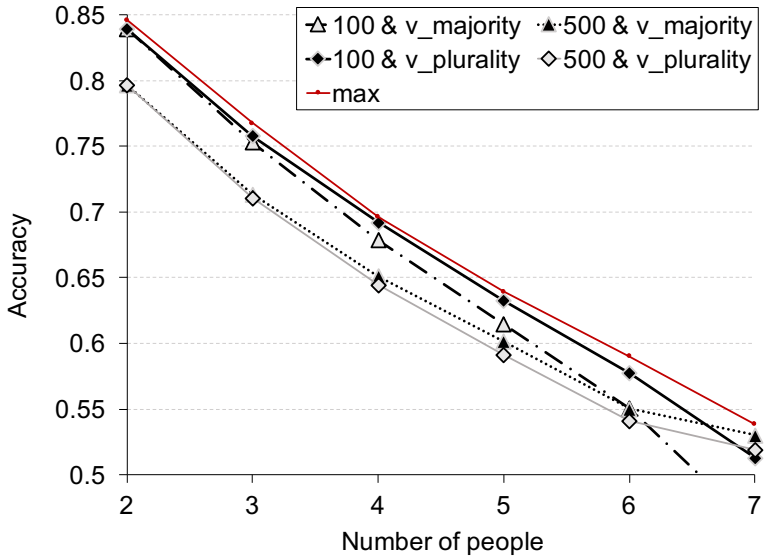


Figure 6.14: Accuracy — Lab environment.

population size of 7 people. The difference between the two sets of values, however, is not statistically significant.

In Figure 6.17 is shown the contrast between the average number of misclassified for all decision time values on the best case overall (i.e., 100 points and plurality) and the misclassified minus undecided on the best case in terms of undecided for plurality (i.e., 500 points) and for majority (i.e., 100 points). We can see that for a population size of 2 or 3 the difference with the best total is either too small as in the case of majority (0.004 and 0.02 for 2 and 3 number of people, respectively), or not even positive for plurality voting. As the population considered increases, the number of undecided also increases, until reaching 0.08 for plurality and 0.12 for majority, at a population size of 7. However, the overall difference with respect to the best total in either case is not statistically significant.

6.5.4 Analysis across studies

The answer to question *RQ3* corresponds to the analysis of the results obtained on the three studies we conducted (i.e., lab and in-the-wild at home and at the office). In Table 6.1 we present, from 2 to 7 people of population size (i.e., the population values shared by all studies), the results for the case which yielded the 'maximum' accuracy for the corresponding environment and the best case on accuracy for the shortest decision time (i.e., 30 s). As we have seen previously, the case with the maximum accuracy for all studies is obtained using 100 points as example size, plurality as voting mechanism, and 90 s as decision time. We can observe that the highest accuracy is obtained for the office environment, followed by the lab, and the home environment is in last place. Between the set of maximum values and the ones for the shortest decision time (i.e., 'fastest'), the difference is not statistically significant for any of the three environments. The average accuracy in both, maximum and fastest cases, for each of the environments is greater than or equal to 0.60, with an aggregate for all studies of 0.67 in the 'fastest' case and

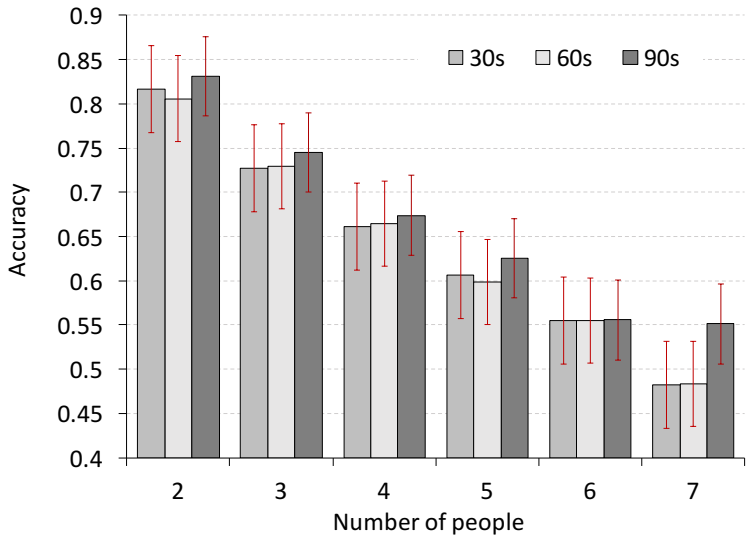


Figure 6.15: Decision times — Lab environment (User Identification).

0.69 in the best case.

Table 6.1: Accuracy for each experiment using Random Forest as classification method.

Number of people	Office		Lab		Home		Avg.	Avg.
	Max.	*Fastest*	*Max.*	*Fastest*	*Max.*	*Fastest*	Max.	Fastest
2	0.88	0.86	0.85	0.84	0.79	0.78	0.84	0.83
3	0.82	0.80	0.77	0.75	0.71	0.68	0.77	0.74
4	0.80	0.77	0.70	0.68	0.63	0.61	0.71	0.69
5	0.75	0.73	0.64	0.62	0.57	0.55	0.65	0.63
6	0.71	0.68	0.59	0.57	0.53	0.51	0.61	0.59
7	0.70	0.68	0.54	0.50	0.49	0.47	0.58	0.55
Aggregate	0.78	0.75	0.68	0.66	0.62	0.60	***0.69***	***0.67***

Figure 6.18 shows a box-plot representation of the average accuracy (considering all decision times, example sizes, and voting mechanisms) of our approach in each of the environments (i.e., lab, home, and office). The figure allows to visualize the results obtained with k-NN and RF as classification methods. As we can see, the results obtained using k-NN show slightly higher accuracy than those that use RF. However, the difference between these two cases is not statistically significant. In the figure we can observe clearly the difference in accuracy across the three environments. The highest accuracy is obtained for the office environment, with an edge over the other two environments which is statistically significant even for a *t-test* with $p < 0.01$. Also noticeable in the figure is that the accuracy obtained in the lab is higher than the one in the home environment; however, in this case the difference is only statistically significant for a *t-test* with $p < 0.1$.

In terms of error, the results from the office environment are statistically significantly lower than those obtained for the home environment (even for a *t-test* with $p < 0.01$) and for the lab (only for a *t-test* with $p < 0.1$). In what respect to proportion of undecided

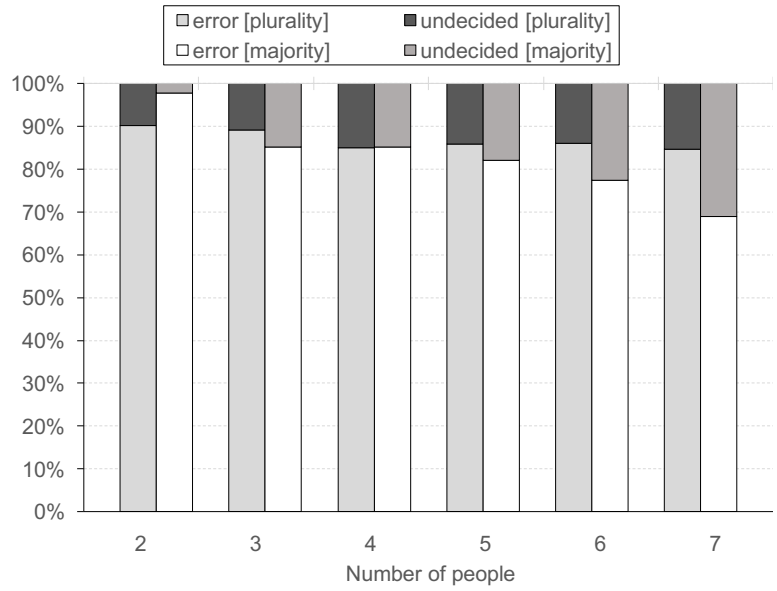


Figure 6.16: Ratio between undecided and total misses — Lab environment (User Identification).

among the total misclassified, the results from the lab environment are statistically significantly higher than those from home and office environments. Furthermore, the undecided for the office is statistically significantly higher than in the home environment.

Overall, the results from the home environment are the ones with the lowest accuracy, the highest error, and the lowest proportion of undecided. This type of outcome may be in part due to the large amount of idle moments that people experience at home, which may have an important impact in the performance of an approach like ours that depends on arm and hand motion patterns. Furthermore, the higher level of heterogeneity and sloppiness with which many people tend to follow their daily routines in this environment may be another issue, since it makes it difficult to build a model that cover most, if not all, of the behavior patterns of individuals, given just a fragment of their daily activity. A potential solution for these issues is to increase the amount of training data, especially focusing on gathering data at different times of the day and on different days of the week.

Concerning the results from the lab, it is relevant to notice that while in terms of accuracy and error the values are midway with respect to office and home environment, in what respect to the proportion of undecided among the amount of misclassified, the results are the highest of the three environments, with a statistically significant difference with respect to the others. This fact may be due to the considerably smaller size of the training data, which although evaluate our approach under what could be considered a simpler problem (i.e., same set of prearranged activities for training and for validation), it seems to affect the formation of a more comprehensive model that avoids such higher number of undecided in validation.

For the office environment, although the results in accuracy and error have an edge over the ones at home and in the lab, the drop in accuracy in function of the increase

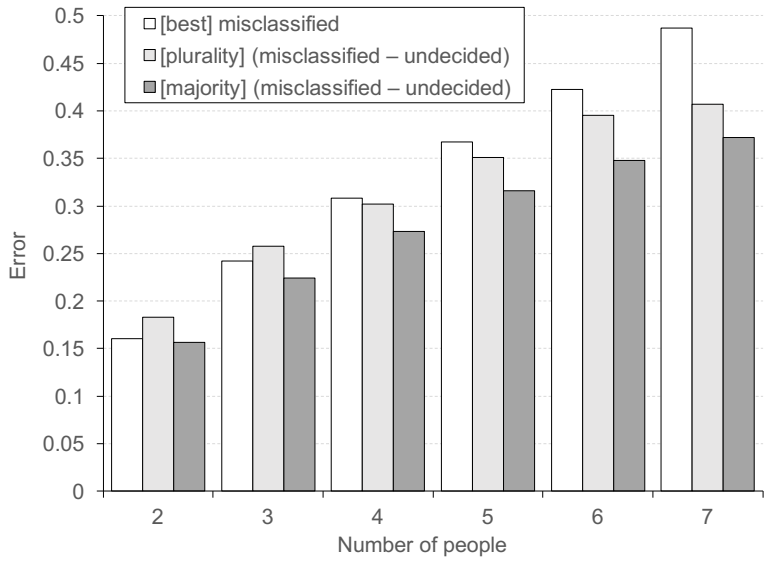


Figure 6.17: Error considering undecided in favor of accuracy — Lab environment (User Identification).

of people considered may be diminished by a similar strategy to what we propose for the home environment, i.e., considering a larger size of training data, with higher heterogeneity, from different moments of the day on different days.

6.6 Discussion

6.6.1 Ecological validity

Although our evaluation has been extensive, a number of issues remain to be able to generalize the findings to real-world scenarios. An important aspect that could be improved in this direction is to conduct studies with a larger number of participants, and consequently increase the number of combinations per population size considered.

The length of both training and validation sessions, as well as the time when they take place represent another relevant aspect for the ecological validity of the evaluation. For our current studies, each session took place in an uninterrupted manner and with no particular attention to the time of the day in which it happened. One step towards improving the validity of the studies could be to widen the duration of each session as well as to have more sessions for training and validation, purposely considering different times of the day and different days of the week.

Furthermore, our evaluation assessed the performance of our approach applied to data previously gathered, instead of considering live data. To improve on this, a complete system that works with live data should be built as a prototype, allowing to examine issues that relate to implementation and deployment of the identification system.

In spite of the valuable results and insights we obtained from our in-situ studies, it is clear that studies covering a larger number of heterogenous households and office spaces would need to be conducted before being able to consider the approach fully validated

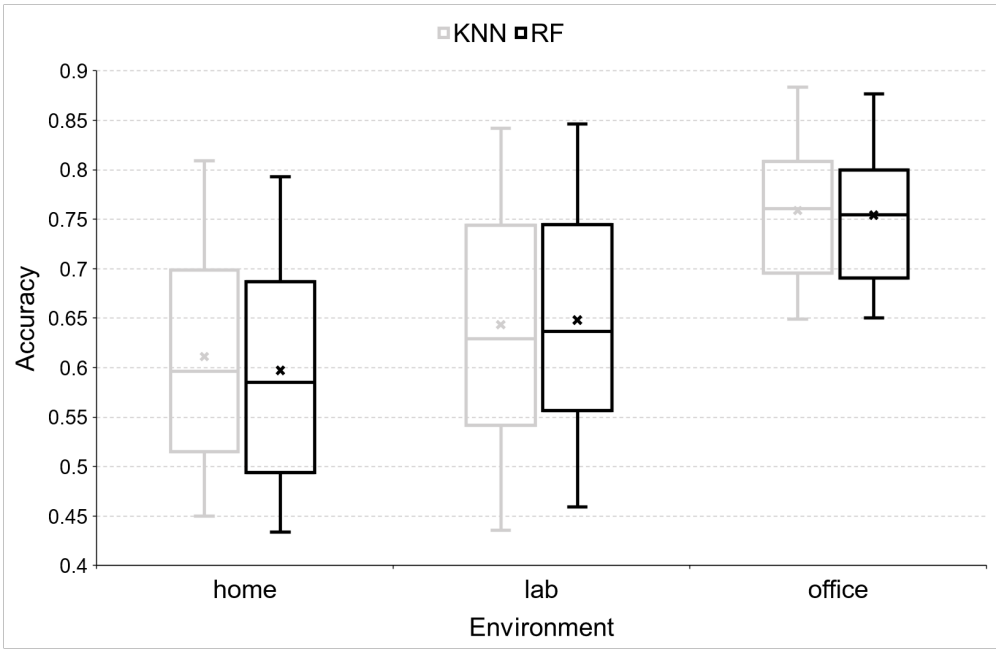


Figure 6.18: Overall results for each of the experiments considered: home, lab, and office (User Identification).

under real-life scenarios.

6.6.2 Approach limitations

The two main limitations of our approach are related to the number of people that it is able to handle at once, and the accuracy it exhibits, which although in a number of cases is acceptable, it is in general not enough to guarantee a seamless and effective continuous identification.

The issue on the population size may be addressed by delimiting predetermined spaces using indoor localization, thus allowing the approach to focus only on a small number of people at any given point in time.

On the other hand, to increase the accuracy of the approach, there are different options that could be considered: *a)* increase the amount of training data, especially obtaining it from different times of the day and on different days; *b)* consider supporting methods, either incorporating them with our approach as part of a multi-modal method, or using other approaches to refine the identification or helping in reaching a final decision; and *c)* use additional context information from other sources to build a more accurate model of each individual.

6.6.3 Approach practicality

Based on our results, we can say that for not very large office spaces and for the average size of a household in countries that are part of the OECD (2.6 inhabitants) our approach provides results which seem promising towards taking it into practice. However, to guarantee seamless and continuous identification we would need to use supporting

methods, which on the one hand should allow to improve the accuracy without the need of user's intervention, and on the other, benefit from the feedback or input of users to improve the learning model over time, in a reinforcement learning configuration.

Another issue on the practicality of our approach is the need for users to have a wearable all the time with them, especially at home in their idle moments. In recent years the use of wearables have grown considerably [Berglund et al., 2016], but it is not possible say that they have become as mainstream as, say, smartphones. A possible solution for this issue could be to build a system able to create a learning model from the behavior of the user based on different wearables and instrumentation (sensors and smart devices) deployed in the space around where the user moves, both for the identification itself and for the initial creation of the learning model and further improvement. In this way, if one wearable is not in use at certain moment, the system can still operate based on, either other wearable in use, or data coming from other sensors or smart devices deployed around the user.

6.6.4 Privacy concerns

As we mentioned in Section 6.4.4, not all data gathered and used as part of our studies is being made publicly available, since some participants (6 in total) agreed to be part of the study as long as their data was not made public. Furthermore, originally we had planned to have a larger number of participants for our studies, but a number of potential participants declined to be part of the study, most of them arguing that they felt uncomfortable by having their personal motion data collected. This happened in spite that we made clear to them that no location information was to be gathered and that the data collected was to be completely anonymized. As many as 5 people declined their participation for the in-situ study at the office, 7 for the in-situ study at home, and 3 for the lab study. In real-world cases this issue has to be considered. As it has been discussed in previous works [Liebling and Preibusch, 2014], we believe that in a real-world implementation of a continuous identification system it is important to provide the user with access to the data that is being collected and processed, as well as to mechanisms to start/stop the automatic learning process at will. In the case of our approach, visualizations of the motion data gathered could help the user to gain trust towards the system at this respect.

6.7 Summary

In this chapter we gave our answer to *RQ2-c*. The answer is in the form of a novel approach for the continuous identification of users in indoor environments based on behavioral biometrics obtained from wrist-worn Inertial Measurement Unit (IMU). The main contribution of this approach to the state of the art is that is able to perform identification without requiring any specific gesture, action, or activity from the users, but allowing them to perform their daily routines freely.

To recognize the identities of users we use the frequency over time of their motion, in particular of their arm and hand motion patterns. Specifically, users are asked to wear

on their wrist IMU's, which contain an accelerometer, a gyroscope, and a magnetometer, and are capable of producing time series of quaternions. We apply wavelet transform on these time series to obtain the observations that are used in the machine-learning classification stage that ultimately decided the identity of each user.

We conducted an empirical evaluation of our approach both in the lab and in the wild. In total we counted with the participation of 29 volunteers for one in the lab study and two studies in real-world settings, one at home and one at the office. Our results show that our approach is able to identify users with an accuracy of 0.88 for office environments and of 0.71 for the average size of a household.

Chapter 7

Application Services

7.1 Introduction

The focus of this chapter is on answering the last of the questions that we formulated around the research question *RQ2*, namely *RQ2-d*: What are examples of application services that can be built from the mechanisms of question *RQ2-a*?

Specifically, we explore smart services that can be built having as support the mechanisms examined in Chapter 5 and Chapter 6. We first provide an overview of such smart services to then present a concrete example of a smart home service that we implemented on top of the approach proposed in Chapter 5. On this last point, we provide the description and results of an in-the-wild qualitative evaluation we conducted on the approach.

7.2 Smart Services

Mechanisms to learn the preferences and frequent habits of users as well as mechanisms to continuously recognize the identity of users are essential in order to provide personalized and adapted services [Aztiria et al., 2012].

Many electric appliances with limited input and output capabilities could benefit from personalization based on user identification [Hayashi et al., 2014]. For instance, an air conditioner can be turned on to a personalized temperature based on the identity of the recognized user. Other examples might include being able to access game consoles, parental control, personalization for electronic appliances like DVRs, or showing some personalized information on wall displays (such as today's schedule, feeds from social networks, emails, and local weather). For most of these applications the user identification is to be designed in favor of usability over security, since the underlying system will typically be deployed in a secure space such as homes and offices.

Some examples of smart services for pervasive systems at home are: smart home health care [Katsivelis et al., 2017], health monitoring for elderly [Pal et al., 2018], energy consumption management [Kibria et al., 2017], smart water tank control [Iqbal et al., 2018], robotic assistance on housework and companionship [Denning et al., 2009], recreation and entertainment [Park et al., 2018], user comfort services (e.g., grocery replenishment) [Wojciechowski and Xiong, 2008], and home automation [Poghosyan

et al., 2017].

Other smart services for different smart environments include [Bello and Zeadally, 2017]: smart city with water, energy, and disaster management services, smart industry with smart grid and smart transportation services, smart office with air quality monitoring, light sensing, and automated access control services.

7.3 Example Implementation — Smart Home Automation

The study of methods to produce rules for the automatic control of a smart environment based on users' behavior patterns is a relevant area that has not received much attention compared to the existent literature in activity and routine discovery. Existing approaches include mining undesired situations to generate reactive rules for smart spaces [Degeler et al., 2014], and the discovery of spatio-temporal interactions between users and intelligent environments to allow the automation of simple actions [Aztiria et al., 2012]. Other methods have focused on extracting temporal association rules that group several devices, based on their interactions, to potentially produce automation functions for smart offices and buildings [Gonzalez and Amft, 2015, Gonzalez and Amft, 2016].

7.3.1 Approach

Our approach to obtain automation routines is built on top of the output produced by the approach we presented in Section 5.3 (Chapter 5).

Step 1 – Production of automation routines from discovered patterns. The input for this step is a set of activity routines yielded by our approach presented in Section 5.3 (Chapter 5). The output is a set of human-understandable routine rules for automation (or *automation routines*) of the form

> `where`: `room`
>
> `when`: `periodicity`, `time slot`
>
> **If** *presence sensed*:
>
> $\langle \texttt{action}_1, \texttt{action}_2, \cdots, \texttt{action}_k \rangle$
>
> **Otherwise**:
>
> $\langle \texttt{action}_{k+1}, \texttt{action}_{k+2}, \cdots, \texttt{action}_n \rangle$

An automation routine is assumed to be valid always (all weeks) whenever the place, day of the week, and time slot correspond to the description.

Step 2 – Construction of automation routines. These are human-understandable automation rules that refer to routines. Similarly to activity routines, automation routines are described per room. Since the detection of activity sessions is based on events that

hint motion or interaction between occupants and (controllable) objects as a clue for 'presence', we use precisely presence as the condition that triggers the beginning and end of the execution of actions of the automation rule. To this end, the sequence of actions of the activity routine is split into two parts, one leading sequence that appears near the start time and the middle and one closing sequence that appears near the end time. The split point is obtained considering the exact start and end times of all activity sessions that support the corresponding activity routine. All actions that appear between the start and the split point are considered as the consequent of the 'presence sensed' condition. Correspondingly, the actions that appear after the split point are considered as the consequent for the 'presence not sensed' (i.e., 'Otherwise') condition.

Output – Automation routines. The output provides a set of simple home automation rules referred to routines, each valid for specific time slot and periodicity. The actions of these automation rules are conditioned on presence, as inferred by means of event data from motion sensors.

7.3.2 Evaluation & Results

The evaluation of our approach in this case is conducted on top of the in-the-wild study we presented in Section 5.4.3 (Chapter 5).

To evaluate our approach on producing automation rules from automatically discovered routine patterns we focus on answering the following question.

ch7-eval-Q1. How does home inhabitants perceive the automation routines suggested by our approach, in terms of their usefulness?

To answer *ch7-eval-Q1* we present the automation routines produced by our approach from the event data to the inhabitants to learn about their impressions and evaluate how many of the rules each inhabitant would be willing to accept to be applied at home. This step takes place as part of the *After* phase of the interviews conducted to the participants of the in-the-wild study in Section 5.4.3 (Chapter 5)

Figure 7.1 shows an excerpt of the automation routines presented to the inhabitants.

All inhabitants were highly interested in the rules (15 out of 18 rules) for the kitchen. For the other rooms, we decided to only enquire the inhabitant that typically spends more time there. The young adult showed interest on nearly half of the rules (20 out of 47 rules) for *MR*, but about the rest of the rules argued that most of them might not be useful in general, since her schedule is constantly changing. The older adults were interested in only a small portion of the rules for *LR* (3 out of 13 rules) and *PR* (4 out of 24 rules). They argued that while it would be interesting to see some things being automated, they would be concerned about malfunctions that could interfere with the regular flow of their daily routines.

```
where: LR
when: Tu Th “late evening”
if presence sensed:
        ('LR_TV_ON')
else:
        ('LR_TV_OFF')

where: MR
when: Tu Su “early evening”
if presence sensed:
        (‘MR_table-lamp-On’, ‘MR_laptop-On’)
else:
        (‘MR_laptop-Off’, ‘MR_table-lamp-Off’)
```

Figure 7.1: Excerpt of automation rules produced by our approach

7.4 Summary

In this chapter we first provided answer to *RQ2-d*. Then we examined an implementation of we built of a smart service: a smart home service that suggest automation rules to users based on the periodic-frequent routines that our approach in Chapter 5 can infer.

For answering the question we provided some examples of potential services that have been proposed in the literature, which deliver value to users through smart functionality.

Concerning the implementation of the smart home service, we presented the steps of the approach we propose to work on top of the learning method in Chapter 5. In addition we provided the description and results of an in-the-wild qualitative evaluation we conducted of the approach.

Part IV

Ending

Chapter 8

Conclusion

8.1 Overall Summary

In this dissertation we have focused on the investigation of the systematic development of value-added smart software services for pervasive systems.

The central body of this dissertation, just as the research efforts involved, has been divided into two parts: the first one (Part II) devoted to introducing our model for the systematic engineering of smart software services for pervasive computing, and the second one (Part III) dedicated to presenting novel approaches to deal with mechanisms that provide support for the development of value-added services for smart environments. The order of these two main parts is meaningful, with the second of the two (i.e., Part III) digging deep into a particular aspect of the engineering model proposed in Part II, namely the one related to data analytics.

Table 8.1 helps us to summarize the research corresponding to Part III with respect to that of Part II. The mechanisms presented in Part III find fit in our proposed engineering model as it is shown in the table. In both cases the 'build' stage of the data analytics requires to set up initial parameters, of which the performance of the approach will depend (even though we have found in our evaluations that the approaches are in general robust to changes on the parameter values). The 'measure' stage can be seen to correspond to the sensing/event data collection and the feature extraction/selection process. The 'learn' phase of the data analytics is directly related to the application of data-mining or machine-learning techniques to produce the learnings of the process. At the bottom of the table we have the product/service. This, in the case of our engineering model, is expected to be a smart software service such as smart home automation, which development is based on the previous establishment of the appropriate data analytics mechanism(s).

To conduct the work for this dissertation we defined research questions around our main objectives given in Section 1.3 (Chapter 1). These questions in turn were broken down into further questions that guided the research process towards achieving the objectives of the dissertation. We have provided answers to these questions throughout the text and in summary in Section 8.3. Our research process within the main parts of the dissertation followed a sound research methodology. Thus, to derive our proposed model for engineering software for pervasive systems we followed analytical, evidence-based,

Table 8.1: Summary of Data Analytics mechanisms presented in the context of the Engineering Model proposed

	Build	**Measure**	**Learn**	
	`Design`	`Execute`	`Analyze`	*Learnings*
Data Analytics	Set up parameters	Obtain activity routines	Discover periodic frequent routines	*Users' periodic frequent routines*
	Set up parameters	Obtain observations from time series using wavelet transform	Perform ML classification	*Users' identities*
Product	Smart Software Service E.g., Smart home automation, Personalization, Access			

and empirical methodologies, which we describe thoroughly in the text. Furthermore, for the approaches proposed in Part III we conducted extensive empirical evaluations to assess their performance under different configurations and settings.

The main contributions of this dissertation to the state of the art are: *i*) a systematic literature review on the phases of the engineering cycle to build software for pervasive systems; *ii*) an engineering model for the systematic development of value-added smart software services for pervasive systems; *iii*) an approach for discovering users' periodic-frequent routines from smart devices and sensors deployed at home; and *iv*) an approach for the continuous identification of users based on arm and hand motion patterns over time, with data captured via a wrist-worn IMU.

Additionally, we aim at contributing to further research in the domains of pattern discovery for smart environments and behavioral biometrics identification by making publicly available the data we have collected and used as part of our empirical evaluations (see Section 5.4.4 (Chapter 5) and Section 6.4.4 (Chapter 6)).

8.2 Future Research Directions

The work presented in this dissertation may be the basis for relevant future research directions, both considering its components separately and as a whole.

For the case of our proposed engineering model in Chapter 4, relevant prospective research paths may be around developing further the model both in breadth and in depth. That is, on the one hand further investigation is to be done in extending the model to make it as comprehensive as possible within the realm of the engineering of software for pervasive systems. This requires the collaboration of industry and academia in order to promote numerous and heterogeneous research efforts that encompass the whole process with several iterations of the engineering cycle.

On the other hand, exploring further the different components and mechanisms that

are part of the proposed engineering process is particularly relevant in aspects related to intelligent behavior, self-adaptability, runtime support, etc. Artificial Intelligence (AI) and Machine Learning (ML) have seen in recent years an accelerated progress, especially with the advent of big data technologies [Kawamoto et al., 2017], the successful revival of neural networks methods in deep learning approaches [Yao et al., 2018], and the dramatic surge of data analytics in terms of accessibility and business value [Turcu and Turcu, 2018]. In spite of this trends, there is a gap between the advancements in these areas and their application in the context of pervasive systems. Therefore, more research efforts have to be made at this respect. In particular, a larger number of extensive empirical studies have to be conducted to advance the state of the art and achieve the intelligent behavior with which pervasive computing systems are envisioned.

Future research work can also be expected concerning the mechanisms that we have explained in this dissertation to provide support for the development of value-added software services for smart environments. Regarding the learning of users' routines at home, a relevant research direction is to extend our proposed approach to include a larger number of context factors to improve the overall performance, as well as to allow more routines, with higher heterogeneity levels to be handled. Furthermore, it would be an important path for future research to explore the use of reinforcement learning mechanisms that allow to improve the performance of the approach over time, based on the learning obtained from previous iterations in combination with continuous users' feedback and active monitoring of the environment.

Regarding the continuous identification of users, a relevant prospective work is to combine our approach with other identification technologies as part of a multi-modal method that uses context factors and feedback from the user to improve progressively the identification accuracy. Thus, the process of identification can use the information at hand depending on the particular situation of the user. For instance, if wearable devices are not in use in a specific moment, the system can switch to rely on smart object, sensors, or signals found in the surroundings.

For our approaches or for any other of a similar kind, a relevant direction for future work is related to empirical research. Studies covering a large number of heterogeneous environments are necessary to further validate the approaches towards their successful applicability in real-life scenarios with end users.

8.3 Answers to Research Questions

Answer to RQ1. The solution to RQ1, fully given in Chapter 4, is obtained by combining the answers to *RQ1-a*, *RQ1-b*, and *RQ1-c*.

Answer to RQ1-a. The solution to this question on how to engineer value-added software for software-intensive systems in a systematic way is given in full in Chapter 2. We presented a process model and an infrastructure model which are aimed at guiding the systematic development of value-added software products/services for software-intensive systems. The key idea of the proposed engineering model is to place what is relevant to users in the center of attention. This is done through

experiment-based software development in which the decision-making process is based on the analysis of data collected from the direct interaction between the user and the software.

Answer to RQ1-b. We provide the full answer to this question in Chapter 3. Our results from the literature review indicate that the stages of the development cycle for software for pervasive systems that have been considered are: design, implementation, deployment, validation and verification, testing, feedback, and evolution and maintenance. From these, the ones that have received more attention are in descending order: implementation, evolution and maintenance, and feedback. It is noticeable the importance that the feedback is given, as pervasive systems require to establish a continuous feedback cycle between the system and the user. The consideration of research challenges obtained from the literature helped us to further our understanding about the engineering process. At this respect, a number of works highlighted the need for further developments in self-adaptability based on machine learning methods, context related issues, and runtime support, as well as on empirical research that promote more sound and reproducible results.

Answer to RQ1-c. The answer to this question is given in full in Chapter 4. Based on the results and analysis of our systematic review of the literature presented in Chapter 3, we say that the key mechanisms to support the development of smart software services for pervasive systems are such that allow, in a semi-supervised or unsupervised manner, to learn about the dynamic characteristics of the users, the environment, the system itself, and other relevant context factors.

To sum up, the resulting engineering model to build systematically value-added smart software services for pervasive systems is mainly characterized by: *i*) centering the attention on the user to produce value-added services; *ii*) featuring data analytics mechanisms aimed at learning about dynamic characteristics of the users, the environment, the system itself, and other relevant context factors, where the operation is expected to be semi-supervised in general and unsupervised whenever appropriate; and *iii*) the use of smart devices and sensors that make up the instrumentation of the system, which is deployed to collect the necessary data for the data analytics mechanisms from the users, the environment, the system itself, and other context factors, as well as to provide functionality.

Answer to RQ2. We give answer to *RQ2* through the solutions to the questions we formulated around it.

Answer to RQ2-a. The solution we provide for this question is that the most relevant mechanisms for providing value-added services through smart functionality in pervasive systems are such that allow to automatically learn various dynamic aspects of users (e.g., behavior patterns). Our answer to this question is presented in full length in Chapter 4. There we use our systematic review of the literature, defined in Chapter 3, as a basis to delineate, for the answer to *RQ1-c*, what are the

main mechanisms to support smart functionality, namely mechanisms to learn the dynamic characteristics of users, the environment, and the system itself. Based on this and the findings in Chapter 2, we put forward as answer that the mechanisms to learn the dynamic characteristics of users (e.g., behavior patterns) are the ones that are more relevant in delivering higher value to users.

Answer to RQ2-b. For this question on how to learn users' routines at home, we proposed a novel approach to discover periodic-frequent routines in data collected from smart devices and sensors deployed at home. The approach is presented in full in Chapter 5, where we also provide details and results of an extensive evaluation we conducted in the lab, in the wild, and based on synthetic data. The approach is the first of its class that is able to find multi-event human routines in home environment that are only frequent for specific periodicities, even when the routines are not repeated in exactly the same way.

Answer to RQ2-c. We provided answer to this question about how to learn to recognize the identities of users in a continuous manner by proposing a novel approach for the continuous identification of users based on hand/arm motion patterns over time. In Chapter 6 we presented the full description of the approach together with its empirical evaluation in the lab and in real home and office environments. We have obtained results that are promising towards achieving a truly continuous identification approach, without the need for specific gestures, actions, or activities from the user. However, further improvements through the use of multimodality (i.e., different biometrics) and more advance learning techniques are necessary to support the practicality of the approach in real-world scenarios.

Answer to RQ2-d. Our answer to this question is presented fully in Chapter 7. From the literature we have found different examples of application services which are supported to a great extent by the mechanisms of *RQ2-a*. Some of such application services are home automation, personalization, access, health monitoring, and energy control. From these, we have investigated the implementation of a home automation service on top of our proposed approach to discover periodic frequent routines of users. Our results allow to have a glimpse on some of the different challenges associated to the construction of smart services that deliver value to users, such as the need for in-the-wild studies that may require a considerable effort in terms of the instrumentation and deployment, as well as the organizational issues related to the participation of people in the studies

In sum, for *RQ2* we say that in order to provide support to the development of value-added services for smart environments it is necessary to establish mechanisms that are able to learn, in a semi-supervised and whenever possible unsupervised way, the dynamic characteristics of users, such as their behavior patterns. Among the most essential mechanisms at this respect, we have focused on approaches for discovering periodic-frequent routines of people at home, and for recognizing the identities of users in a continuous manner based on motion patterns over time.

8.4 Validity of the Research

The research work of this dissertation has been conducted with the intent of being generalizable and applicable to real-life scenarios.

In spite of our efforts, some issues still remain in terms of external and ecological validity. Concerning our proposed engineering model, it is of positive value the fact that we derived our model based on sound evidence from the literature and observations supported by empirical research. However, in order to ensure the generalizability of the model, it is necessary to assess the implementation of the model empirically, via extensive studies that evaluate the process as a whole through numerous iterations in various settings, and under different environments. In the case of the applicability of our proposed engineering model to real-life scenarios, it is also required to promote a larger number of empirical research efforts, especially through studies in the wild.

Regarding the approaches we proposed on the mechanisms that are meant to support the engineering of value-added services for smart environments, we have tried to ensure generalizability through extensive empirical evaluation under different configurations and settings. Also, we have sought to ensure ecological validity by conducting in-the-wild studies. However, in terms of generalizability a wider and larger set of plausible scenarios should be considered, e.g., more complex activities, larger amount of people with widely different profiles, and more heterogeneous environments. For ecological validity as well there are improvement to make, such as consider longer period of time (e.g., months instead of days), wider areas for the study, and larger amount of spaces (e.g., 50 or 100 homes instead of just a few).

Appendix A

Appendix – Approaches of the Systematic Review

Table A.1: Systematic Review — Paper - approach - QA

Paper id	Year	Source	Approach(phase)	QA score
P1 [Cassou et al., 2012]	2012	IEEE Transactions on Software Engineering	A1(des), A2(all), A3(impl)	5
P2 [Mamei and Zambonelli, 2009]	2009	ACM Transactions on Software Engineering and Methodology	A4(fdbk, evol)	5
P3 [Cassou et al., 2011]	2011	Proceedings of the International Conference on Software Engineering	A5(des, impl, v&v)	4
P4 [Xu et al., 2015]	2015	IEEE Transactions on Software Engineering	A6(fdbk)	6
P5 [Zachariadis et al., 2006]	2006	IEEE Transactions on Software Engineering	A7(impl)	5
P6 [Julien and Roman, 2006]	2006	IEEE Transactions on Software Engineering	A8(impl)	5
P7 [Coronato and De Pietro, 2012]	2012	IEEE Transactions on Software Engineering	A9(v&v)	5
P8 [Schreiber et al., 2012]	2012	IEEE Transactions on Software Engineering	A10(evol)	5
P9 [Sama et al., 2010]	2010	IEEE Transactions on Software Engineering	A11(v&v)	5
P10 [Forte et al., 2008]	2008	Journal of Systems and Software	A12(evol)	4

continued on next page

Table A.1 – continued from previous page

Paper id	Year	Source	Approach(phase)	QA score
P11 [Poladian et al., 2004]	2004	Proceedings of the International Conference on Software Engineering	A13(evol)	5
P12 [Xu et al., 2006]	2006	Proceedings of the International Conference on Software Engineering	A14(v&v)	4
P13 [Hallsteinsen et al., 2012]	2012	Journal of Systems and Software	A15(all)	6
P14 [Salifu et al., 2012]	2012	Journal of Systems and Software	A16(re)	4
P15 [Xu et al., 2012a]	2012	Journal of Systems and Software	A17(evol)	5
P16 [Boix et al., 2014]	2014	Journal of Systems and Software	A18(fdbk)	4
P17 [Kiani et al., 2013]	2013	Journal of Systems and Software	A19(impl)	5
P18 [Wang et al., 2007]	2007	Proceedings of the International Conference on Software Engineering	A20(test)	4
P19 [Lu et al., 2006]	2006	Proceedings of the International symposium on Foundations of software engineering	A21(test)	6
P20 [Sama et al., 2008]	2008	Proceedings of the International symposium on Foundations of software engineering	A22(v&v)	5
P21 [Xu et al., 2010]	2010	ACM Transactions on Software Engineering and Methodology	A23(v&v)	6
P22 [Lu et al., 2008]	2008	Proceedings of the International Conference on Software Engineering	A24(test)	6
P23 [Malek et al., 2010]	2010	Journal of Systems and Software	A25(all)	5
P24 [Riva and Toivonen, 2007]	2007	Journal of Systems and Software	A26(fdbk)	4
P25 [Liu et al., 2013]	2013	Journal of Systems and Software	A27(v&v)	6

continued on next page

Table A.1 – continued from previous page

Paper id	Year	Source	Approach(phase)	QA score
P26 [Kulkarni et al., 2012]	2012	ACM Transactions on Software Engineering and Methodology	A28(des, impl)	5
P27 [Liu and Cheng, 2011]	2011	Journal of Systems and Software	A29(evol)	5
P28 [Morin et al., 2009]	2009	Proceedings of the International Conference on Software Engineering	A30(evol)	3
P29 [Ballesteros et al., 2012]	2012	Journal of Systems and Software	A31(impl)	3
P30 [Murukannaiah and Singh, 2015]	2015	ACM Transactions on Software Engineering and Methodology	A32(impl, fdbk)	5
P31 [Meier and Cahil, 2010]	2010	IEEE Transactions on Software Engineering	A33(impl)	5
P32 [Verbelen et al., 2011]	2011	Journal of Systems and Software	A34(deploy)	5
P33 [Esfahani et al., 2011]	2011	Proceedings of the European conference on Foundations of software engineering	A35(evol)	5
P34 [Malek et al., 2005]	2005	IEEE Transactions on Software Engineering	A36(impl)	5
P35 [Xu and Cheung, 2005]	2005	Proceedings of the symposium on the Foundations of Software Engineering	A37(fdbk, evol)	3
P36 [Morris et al., 2015]	2015	Information and Software Technology	A38(impl)	6
P37 [Payton et al., 2010]	2010	ACM Transactions on Software Engineering and Methodology	A39(evol)	5
P38 [Becker et al., 2004]	2004	Proceedings of the Conference on Pervasive Computing and Communications	A40(evol)	4
P39 [Mamei and Zambonelli, 2004]	2004	Proceedings of the Conference on Pervasive Computing and Communications	A41(impl)	4

continued on next page

Table A.1 – continued from previous page

Paper id	Year	Source	Approach(phase)	QA score
P40 [Becker et al., 2003]	2003	Proceedings of the Conference on Pervasive Computing and Communications	A42(impl)	3
P41 [Chen et al., 2011]	2011	Proceedings of the Conference on Pervasive Computing and Communications	A43(fdbk,impl)	5
P42 [Hu et al., 2008]	2008	Proceedings of the Conference on Pervasive Computing and Communications	A44(evol, impl)	3
P43 [Ballesteros et al., 2006]	2006	Proceedings of the Conference on Pervasive Computing and Communications	A45(impl)	3
P44 [Munnelly et al., 2007]	2007	Proceedings of the Conference on Pervasive Computing and Communications	A46(des)	5
P45 [Henricksen and Indulska, 2006]	2006	Pervasive and mobile computing	A47(all)	4
P46 [Aitenbichler et al., 2007]	2007	Pervasive and mobile computing	A48(impl)	4
P47 [Wei and Chan, 2013]	2013	Pervasive and mobile computing	A49(des, evol)	5
P48 [Serral et al., 2010]	2010	Pervasive and mobile computing	A50(des, impl, evol)	5
P49 [McGlinn et al., 2014]	2014	Pervasive and mobile computing	A51(test)	5
P50 [O'Neill et al., 2013]	2013	Pervasive and mobile computing	A52(test)	5
P51 [Achilleos et al., 2010]	2010	Pervasive and mobile computing	A53(impl, des)	3
P52 [Pallapa et al., 2014]	2014	Pervasive and mobile computing	A54(impl), A55(fdbk)	4
P53 [Wei et al., 2012]	2012	Proceedings of the Conference on Pervasive Computing and Communications	A56(fdbk, evol)	3
P54 [Arnaboldi et al., 2014]	2014	Pervasive and mobile computing	A57(impl)	5

continued on next page

Table A.1 – continued from previous page

Paper id	Year	Source	Approach(phase)	QA score
P55 [Xue et al., 2013]	2013	Pervasive and mobile computing	A58(fdbk)	4
P56 [Gopalan and Znati, 2010]	2010	Pervasive and mobile computing	A59(deploy)	5
P57 [VanSyckel et al., 2014]	2014	Pervasive and mobile computing	A60(impl, evol)	5
P58 [Pham et al., 2009]	2009	Pervasive and mobile computing	A61(impl)	3
P59 [Ou et al., 2007]	2007	Pervasive and mobile computing	A62(evol)	5
P60 [Driver and Clarke, 2008]	2008	Pervasive and mobile computing	A63 (fdbk)	4
P61 [Paluska et al., 2008]	2008	Pervasive and mobile computing	A64(impl)	4
P62 [Athanasopoulos et al., 2008]	2008	Pervasive and mobile computing	A65(fdbk)	5
P63 [Nishikawa et al., 2006]	2006	Proceedings of the international conference on Ubiquitous computing	A66 (test)	3
P64 [Kawsar et al., 2008]	2008	Proceedings of the international conference on Ubiquitous computing	A67(deploy)	5
P65 [Weis et al., 2007]	2007	IEEE Pervasive Computing	A68(des)	4
P66 [Bannach et al., 2008]	2008	IEEE Pervasive Computing	A69(impl)	4
P67 [Gu et al., 2004]	2004	IEEE Pervasive Computing	A70(evol)	5
P68 [Hoareau and Mahéo, 2008]	2008	Personal and ubiquitous computing	A71(deploy)	3
P69 [Lézoray et al., 2011]	2011	Personal and ubiquitous computing	A72(des)	3
P70 [Guo et al., 2011]	2011	Personal and ubiquitous computing	A73(des)	3
P71 [Paspallis and Papadopoulos, 2014]	2014	Personal and ubiquitous computing	A74(des, impl, deploy)	3
P72 [Soldatos et al., 2007]	2007	Personal and ubiquitous computing	A75(deploy, impl)	3
P73 [Bruneau and Consel, 2013]	2013	Software: Practice and Experience	A76(test)	5

continued on next page

Table A.1 – continued from previous page

Paper id	Year	Source	Approach(phase)	QA score
P74 [Geihs et al., 2009]	2009	Software: Practice and Experience	A77(all)	4
P75 [Yau et al., 2006]	2006	Software: Practice and Experience	A78(all)	3
P76 [Romero et al., 2013]	2013	Software: Practice and Experience	A79(impl)	3
P77 [Bak et al., 2011]	2011	Proceedings of the international conference on Cyber-Physical Systems	A80(v&v)	3
P78 [Wang et al., 2013]	2013	Proceedings of the international conference on Cyber-Physical Systems	A81(des)	3
P79 [Castelli et al., 2015]	2015	ACM Transactions on Autonomous and Adaptive Systems	A82(impl)	5
P80 [Wang et al., 2014]	2014	ACM Transactions on Autonomous and Adaptive Systems	A83(test)	6
P81 [Gui et al., 2011]	2011	Journal of Systems and Software	A84(evol)	5
P82 [Payton et al., 2012]	2012	Pervasive and Mobile Computing	A85(fdbk)	4
P83 [Cooray et al., 2013]	2013	IEEE Transactions on Software Engineering	A86(evol)	6
P84 [Gehrke and Madden, 2004]	2004	IEEE Pervasive Computing	A87(fdbk)	5
P85 [Blackstock et al., 2008]	2008	Ubicomp*	A88(impl)	5
P86 [Acharya et al., 2009]	2009	Proceedings of the Conference on Pervasive Computing and Communications	A89(impl)	5
P87 [Meier et al., 2009]	2009	Pervasive and Mobile Computing	A90(fdbk)	5
P88 [Jaroucheh et al., 2012]	2012	Personal and Ubiquitous Computing	A91(impl)	2
P89 [Schuhmann et al., 2010]	2010	Ubicomp*	A92(impl)	5
P90 [Rashid et al., 2012]	2012	Pervasive and Mobile Computing	A93(impl)	5

continued on next page

Table A.1 – continued from previous page

Paper id	Year	Source	Approach(phase)	QA score
P91 [Holzmann and Ferscha, 2010]	2010	Pervasive and Mobile Computing	A94(fdbk)	4
P92 [Nahrstedt et al., 2005]	2005	Pervasive and Mobile Computing	A95(fdbk)	5
P93 [Hess and Campbell, 2003]	2003	Personal and Ubiquitous Computing	A96(impl)	4
P94 [Malandrino et al., 2010]	2010	Personal and ubiquitous computing	A97(fdbk, evol)	5
P95 [Salvaneschi et al., 2012]	2012	Proceedings of the international conference on Aspect-oriented Software Development	A98(des)	5
P96 [Schuhmann et al., 2013]	2013	ACM Transactions on Autonomous and Adaptive Systems	A99(evol)	5
P97 [Seinturier et al., 2012]	2012	Software: Practice and Experience	A100(evol)	4
P98 [Roussaki et al., 2010]	2010	Pervasive and Mobile Computing	A101(fdbk, impl)	4
P99 [Bettini et al., 2008]	2008	Pervasive and Mobile Computing	A102(impl)	4
P100 [Chan and Chuang, 2003]	2003	IEEE Transactions on Software Engineering	A103(fdbk, evol, deploy)	4
P101 [Xu et al., 2012b]	2012	Proceedings of the conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services	A104(fdbk)	2
P102 [Ali et al., 2013]	2013	Information and Software Technology	A105(re, fdbk)	5
P103 [Ali et al., 2010]	2010	Requirements Engineering	A106(re)	4
P104 [Füller et al., 2012]	2012	Pervasive and Mobile Computing	A107(fdbk), A118(fdbk)	5
P105 [Harrington and Cahill, 2011]	2011	Pervasive and Mobile Computing	A109(impl, evol)	5
P106 [Robinson et al., 2008]	2008	Pervasive and Mobile Computing	A110(impl)	5
P107 [Morla and Davies, 2004]	2004	IEEE Pervasive computing	A111(test)	4

continued on next page

Table A.1 – continued from previous page

Paper id	Year	Source	Approach(phase)	QA score
P108 [Gui et al., 2013]	2013	Software: Practice and Experience	A112(evol)	4
P109 [Chuang and Chan, 2008]	2008	IEEE Transactions on Software Engineering	A113(evol)	5
P110 [Capra et al., 2003]	2003	IEEE Transactions on Software Engineering	A114(evol)	4
P111 [Hoffmann and Söllner, 2014]	2014	Personal and ubiquitous computing	A115(fdbk)	2
P112 [Gámez and Fuentes, 2011]	2011	Personal and Ubiquitous Computing	A116(evol)	3
P113 [Cetina et al., 2009]	2009	IEEE Computer	A117(des, evol)	5
P114 [Liogkas et al., 2004]	2004	IEEE Pervasive Computing	A118(impl)	5
P115 [Zisman et al., 2013]	2013	IEEE Transactions on Software Engineering	A119(impl)	5
P116 [Sethi et al., 2014]	2014	Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing	A120(evol)	5
P117 [Spínola and Travassos, 2012]	2012	Information and Software Technology	A121(all)	6
P118 [Thomas et al., 2014]	2014	Proceedings of the International Conference on Software Engineering	A122(re)	4
P119 [Inverardi and Tivoli, 2013]	2013	Proceedings of the International Conference on Software Engineering	A123(evol)	5
P120 [Dalpiaz et al., 2013]	2013	Requirements engineering	A124(re)	5
P121 [Guo et al., 2010]	2010	Computer Networks	A125(fdbk, evol)	5
P122 [Baresi et al., 2012]	2012	IEEE Computer	A126(evol)	5
P123 [Chuang et al., 2011]	2011	Proceedings of the Symposium on Research in Applied Computation	A127(test)	3
P124 [Basanta-Val et al., 2013]	2013	Software: Practice and Experience	A128(impl)	5

continued on next page

Table A.1 – continued from previous page

Paper id	Year	Source	Approach(phase)	QA score
P125 [Huang et al., 2010]	2010	Proceedings of the International Conference on Cyber-Physical Systems	A129(test)	4
P126 [Moros et al., 2013]	2013	Information and Software Technology	A130(re)	6
P127 [Gamez and Fuentes, 2013]	2013	Information and Software Technology	A131(evol)	5
P128 [Ruiz-López et al., 2013]	2013	Science of Computer Programming	A132(re)	5

Bibliography

[Abdelnasser et al., 2015] Abdelnasser, H., Youssef, M., and Harras, K. A. (2015). Wigest: A ubiquitous wifi-based gesture recognition system. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 1472–1480. IEEE.

[Abowd, 1999] Abowd, G. D. (1999). Software engineering issues for ubiquitous computing. In *Software Engineering, 1999. Proceedings of the 1999 International Conference on*, pages 75–84. IEEE.

[Abowd, 2012] Abowd, G. D. (2012). What next, ubicomp?: celebrating an intellectual disappearing act. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 31–40. ACM.

[Acharya et al., 2009] Acharya, A., Banerjee, N., Chakraborty, D., Dasgupta, K., Misra, A., Sharma, S., Wang, X., and Wright, C. P. (2009). Programmable presence virtualization for next-generation context-based applications. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1–10. IEEE.

[Achilleos et al., 2010] Achilleos, A., Yang, K., and Georgalas, N. (2010). Context modelling and a context-aware framework for pervasive service creation: A model-driven approach. *Pervasive and Mobile Computing*, 6(2):281–296.

[Ackad et al., 2012] Ackad, C., Clayphan, A., Maldonado, R. M., and Kay, J. (2012). Seamless and continuous user identification for interactive tabletops using personal device handshaking and body tracking. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, pages 1775–1780. ACM.

[Adams et al., 2013] Adams, R. J., Evans, B., and Brandt, J. (2013). Creating small products at a big company: adobe's pipeline innovation process. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 2331–2332. ACM.

[Agadakos et al., 2016] Agadakos, I., Hallgren, P., Damopoulos, D., Sabelfeld, A., and Portokalidis, G. (2016). Location-enhanced authentication using the iot: because you cannot be in two places at once. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 251–264. ACM.

[Agrawal and Srikant, 1995] Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE.

[Agrawal et al., 1994] Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.

[Aitenbichler et al., 2007] Aitenbichler, E., Kangasharju, J., and Mühlhäuser, M. (2007). Mundocore: A light-weight infrastructure for pervasive computing. *Pervasive and Mobile Computing*, 3(4):332–361.

[Aken, 2004] Aken, J. E. v. (2004). Management research based on the paradigm of the design sciences: the quest for field-tested and grounded technological rules. *Journal of management studies*, 41(2):219–246.

[Al-Muhtadi et al., 2003] Al-Muhtadi, J., Ranganathan, A., Campbell, R., and Mickunas, M. D. (2003). Cerberus: a context-aware security scheme for smart spaces. In *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on*, pages 489–496. IEEE.

[Alam et al., 2012] Alam, M. R., Reaz, M. B. I., and Ali, M. A. M. (2012). A review of smart homes—past, present, and future. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(6):1190–1203.

[Alhamoud et al., 2014] Alhamoud, A., Nair, A. A., Gottron, C., Böhnstedt, D., and Steinmetz, R. (2014). Presence detection, identification and tracking in smart homes utilizing bluetooth enabled smartphones. In *Local Computer Networks Workshops (LCN Workshops), 2014 IEEE 39th Conference on*, pages 784–789. IEEE.

[Ali et al., 2010] Ali, R., Dalpiaz, F., and Giorgini, P. (2010). A goal-based framework for contextual requirements modeling and analysis. *Requirements Engineering*, 15(4):439–458.

[Ali et al., 2013] Ali, R., Dalpiaz, F., and Giorgini, P. (2013). Reasoning with contextual requirements: Detecting inconsistency and conflicts. *Information and Software Technology*, 55(1):35–57.

[Amphawan et al., 2009] Amphawan, K., Lenca, P., and Surarerks, A. (2009). Mining top-k periodic-frequent pattern from transactional databases without support threshold. *Advances in Information Technology*, pages 18–29.

[Arnaboldi et al., 2014] Arnaboldi, V., Conti, M., and Delmastro, F. (2014). Cameo: A novel context-aware middleware for opportunistic mobile social networks. *Pervasive and Mobile Computing*, 11:148–167.

[Athanasopoulos et al., 2008] Athanasopoulos, D., Zarras, A. V., Issarny, V., Pitoura, E., and Vassiliadis, P. (2008). Cowsami: Interface-aware context gathering in ambient intelligence environments. *Pervasive and Mobile Computing*, 4(3):360–389.

[Aurum and Wohlin, 2003] Aurum, A. and Wohlin, C. (2003). The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14):945–954.

[Aztiria et al., 2012] Aztiria, A., Augusto, J. C., Basagoiti, R., Izaguirre, A., and Cook, D. J. (2012). Discovering frequent user–environment interactions in intelligent environments. *Personal and Ubiquitous Computing*, 16(1):91–103.

[Aztiria et al., 2010] Aztiria, A., Izaguirre, A., and Augusto, J. C. (2010). Learning patterns in ambient intelligence environments: a survey. *Artificial Intelligence Review*, 34(1):35–51.

[Bak et al., 2011] Bak, S., Manamcheri, K., Mitra, S., and Caccamo, M. (2011). Sandboxing controllers for cyber-physical systems. In *Cyber-Physical Systems (ICCPS), 2011 IEEE/ACM International Conference on*, pages 3–12. IEEE.

[Ballesteros et al., 2012] Ballesteros, F. J., Soriano, E., and Guardiola, G. (2012). Octopus: An upperware based system for building personal pervasive environments. *Journal of Systems and Software*, 85(7):1637–1649.

[Ballesteros et al., 2006] Ballesteros, F. J., Soriano, E., Leal, K., and Guardiola, G. (2006). Plan b: An operating system for ubiquitous computing environments. In *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, pages 10–pp. IEEE.

[Bamberg et al., 2008] Bamberg, S. J. M., Benbasat, A. Y., Scarborough, D. M., Krebs, D. E., and Paradiso, J. A. (2008). Gait analysis using a shoe-integrated wireless sensor system. *IEEE transactions on information technology in biomedicine*, 12(4):413–423.

[Bamis et al., 2010] Bamis, A., Fang, J., and Savvides, A. (2010). A method for discovering components of human rituals from streams of sensor data. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 779–788. ACM.

[Bannach et al., 2008] Bannach, D., Lukowicz, P., and Amft, O. (2008). Rapid prototyping of activity recognition applications. *Pervasive Computing, IEEE*, 7(2):22–31.

[Baresi et al., 2012] Baresi, L., Guinea, S., and Pasquale, L. (2012). Service-oriented dynamic software product lines. *Computer*, (10):42–48.

[Barralon et al., 2006] Barralon, P., Vuillerme, N., and Noury, N. (2006). Walk detection with a kinematic sensor: Frequency and wavelet comparison. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 1711–1714. IEEE.

[Basanta-Val et al., 2013] Basanta-Val, P., García-Valls, M., and Estévez-Ayres, I. (2013). Enhancing osgi with real-time java support. *Software: Practice and Experience*, 43(1):33–65.

[Basili et al., 2007] Basili, V., Heidrich, J., Lindvall, M., Münch, J., Regardie, M., Rombach, D., Seaman, C., and Trendowicz, A. (2007). Gqm+strategies: A comprehensive methodology for aligning business strategies with software measurement. In *DASMA Software Metric Congress (MetriKon 2007), At Kaiserslautern, Germany*.

[Basili et al., 1986] Basili, V. R., Selby, R. W., and Hutchens, D. H. (1986). Experimentation in software engineering. *IEEE Transactions on software engineering*, (7):733–743.

[Becker et al., 2004] Becker, C., Handte, M., Schiele, G., and Rothermel, K. (2004). Pcom-a component system for pervasive computing. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 67–76. IEEE.

[Becker et al., 2003] Becker, C., Schiele, G., Gubbels, H., and Rothermel, K. (2003). Base-a micro-broker-based middleware for pervasive computing. In *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on*, pages 443–451. IEEE.

[Bello and Zeadally, 2017] Bello, O. and Zeadally, S. (2017). Toward efficient smartification of the internet of things (iot) services. *Future Generation Computer Systems*.

[Bental et al., 2015] Bental, D. S., Papadopoulou, E., Taylor, N. K., Williams, M. H., Blackmun, F. R., Ibrahim, I. S., Lim, M. Y., Mimtsoudis, I., Whyte, S. W., and Jennings, E. (2015). Smartening up the student learning experience with ubiquitous media. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 12(1s):23.

[Berglund et al., 2016] Berglund, M. E., Duvall, J., and Dunne, L. E. (2016). A survey of the historical scope and current trends of wearable technology applications. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, pages 40–43. ACM.

[Bettini et al., 2008] Bettini, C., Pareschi, L., and Riboni, D. (2008). Efficient profile aggregation and policy evaluation in a middleware for adaptive mobile applications. *Pervasive and Mobile Computing*, 4(5):697–718.

[Blackstock et al., 2008] Blackstock, M., Lea, R., and Krasic, C. (2008). Evaluation and analysis of a common model for ubiquitous systems interoperability. In *Pervasive Computing*, pages 180–196. Springer.

[Blank, 2013] Blank, S. (2013). *The four steps to the epiphany: successful strategies for products that win*. BookBaby.

[Boix et al., 2014] Boix, E. G., Scholliers, C., De Meuter, W., and D'Hondt, T. (2014). Programming mobile context-aware applications with totam. *Journal of Systems and Software*, 92:3–19.

[Bosch, 2012] Bosch, J. (2012). Building products as innovation experiment systems. In *International Conference of Software Business*, pages 27–39. Springer.

[Bosch et al., 2013] Bosch, J., Olsson, H. H., Björk, J., and Ljungblad, J. (2013). The early stage software startup development model: a framework for operationalizing lean principles in software startups. In *Lean Enterprise Software and Systems*, pages 1–15. Springer.

[Brajdic and Harle, 2013] Brajdic, A. and Harle, R. (2013). Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 225–234. ACM.

[Bränzel et al., 2013] Bränzel, A., Holz, C., Hoffmann, D., Schmidt, D., Knaust, M., Lühne, P., Meusel, R., Richter, S., and Baudisch, P. (2013). Gravityspace: tracking users and their poses in a smart room using a pressure-sensing floor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 725–734. ACM.

[Braun and Clarke, 2006] Braun, V. and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101.

[Bromme, 2003] Bromme, A. (2003). A classification of biometric signatures. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 3, pages III–17. IEEE.

[Brown et al., 2014] Brown, A., Mortier, R., and Rodden, T. (2014). An exploration of user recognition on domestic networks using netflow records. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 903–910. ACM.

[Brown et al., 2011] Brown, B., Reeves, S., and Sherwood, S. (2011). Into the wild: challenges and opportunities for field trial methods. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1657–1666. ACM.

[Brun et al., 2009] Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., and Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*, pages 48–70. Springer.

[Bruneau and Consel, 2013] Bruneau, J. and Consel, C. (2013). Diasim: a simulator for pervasive computing applications. *Software: Practice and Experience*, 43(8):885–909.

[Brush et al., 2011] Brush, A., Lee, B., Mahajan, R., Agarwal, S., Saroiu, S., and Dixon, C. (2011). Home automation in the wild: challenges and opportunities. In *proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2115–2124. ACM.

[Bures et al., 2017] Bures, T., Weyns, D., Schmer, B., Tovar, E., Boden, E., Gabor, T., Gerostathopoulos, I., Gupta, P., Kang, E., Knauss, A., et al. (2017). Software engineering for smart cyber-physical systems: challenges and promising solutions. *ACM SIGSOFT Software Engineering Notes*, 42(2):19–24.

[Burgbacher and Hinrichs, 2014] Burgbacher, U. and Hinrichs, K. (2014). An implicit author verification system for text messages based on gesture typing biometrics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2951–2954. ACM.

[Cappelli et al., 2006] Cappelli, R., Maio, D., Maltoni, D., Wayman, J. L., and Jain, A. K. (2006). Performance evaluation of fingerprint verification systems. *IEEE transactions on pattern analysis and machine intelligence*, 28(1):3–18.

[Capra et al., 2003] Capra, L., Emmerich, W., and Mascolo, C. (2003). Carisma: Context-aware reflective middleware system for mobile applications. *Software Engineering, IEEE Transactions on*, 29(10):929–945.

[Cassou et al., 2011] Cassou, D., Balland, E., Consel, C., and Lawall, J. (2011). Leveraging software architectures to guide and verify the development of sense/compute/control applications. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 431–440. ACM.

[Cassou et al., 2012] Cassou, D., Bruneau, J., Consel, C., and Balland, E. (2012). Toward a tool-based development methodology for pervasive computing applications. *Software Engineering, IEEE Transactions on*, 38(6):1445–1463.

[Castelli et al., 2015] Castelli, G., Mamei, M., Rosi, A., and Zambonelli, F. (2015). Engineering pervasive service ecosystems: the sapere approach. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(1):1.

[Castro et al., 2015] Castro, D., Hickson, S., Bettadapura, V., Thomaz, E., Abowd, G., Christensen, H., and Essa, I. (2015). Predicting daily activities from egocentric images using deep learning. In *proceedings of the 2015 ACM International symposium on Wearable Computers*, pages 75–82. ACM.

[Cetina et al., 2009] Cetina, C., Giner, P., Fons, J., and Pelechano, V. (2009). Autonomic computing through reuse of variability models at runtime: The case of smart homes. *Computer*, 42(10):37–43.

[Chan and Chuang, 2003] Chan, A. T. and Chuang, S.-N. (2003). Mobipads: a reflective middleware for context-aware mobile computing. *Software Engineering, IEEE Transactions on*, 29(12):1072–1085.

[Chan et al., 2008] Chan, M., Estève, D., Escriba, C., and Campo, E. (2008). A review of smart homes—present state and future challenges. *Computer methods and programs in biomedicine*, 91(1):55–81.

[Chauhan et al., 2017] Chauhan, J., Hu, Y., Seneviratne, S., Misra, A., Seneviratne, A., and Lee, Y. (2017). Breathprint: Breathing acoustics-based user authentication. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 278–291. ACM.

[Chen et al., 2011] Chen, C., Ye, C., and Jacobsen, H.-A. (2011). Hybrid context inconsistency resolution for context-aware services. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 10–19. IEEE.

[Chen et al.,] Chen, L., Ali Babar, M., and Zhang, H. Towards evidence-based understanding of electronic data sources. In *14th International Conference on Evaluation and Assessment in Software Engineering (EASE)*.

[Chia et al., 2015] Chia, S.-C., Mandal, B., Xu, Q., Li, L., and Lim, J.-H. (2015). Enhancing social interaction with seamless face recognition on google glass: Leveraging opportunistic multi-tasking on smart phones. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, pages 750–757. ACM.

[Chuang et al., 2011] Chuang, K.-C., Shih, C.-S., and Hung, S.-H. (2011). User behavior augmented software testing for user-centered gui. In *Proceedings of the 2011 ACM Symposium on Research in Applied Computation*, pages 200–208. ACM.

[Chuang and Chan, 2008] Chuang, S.-N. and Chan, A. T. (2008). Dynamic qos adaptation for mobile middleware. *Software Engineering, IEEE Transactions on*, 34(6):738–752.

[Ciria et al., 2014] Ciria, J. C., Domínguez, E., Escario, I., Francés, Á., Lapeña, M. J., and Zapata, M. A. (2014). The history-based authentication pattern. In *Proceedings of the 19th European Conference on Pattern Languages of Programs*, page 30. ACM.

[Conti et al., 2012] Conti, M., Das, S. K., Bisdikian, C., Kumar, M., Ni, L. M., Passarella, A., Roussos, G., Tröster, G., Tsudik, G., and Zambonelli, F. (2012). Looking ahead

in pervasive computing: Challenges and opportunities in the era of cyber–physical convergence. *Pervasive and Mobile Computing*, 8(1):2–21.

[Cook and Das, 2004] Cook, D. and Das, S. K. (2004). *Smart environments: Technology, protocols and applications*, volume 43. John Wiley & Sons.

[Cook et al., 2009] Cook, D. J., Augusto, J. C., and Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298.

[Cook and Das, 2007] Cook, D. J. and Das, S. K. (2007). How smart are our environments? an updated look at the state of the art. *Pervasive and mobile computing*, 3(2):53–73.

[Cook et al., 2013] Cook, D. J., Krishnan, N. C., and Rashidi, P. (2013). Activity discovery and activity recognition: A new partnership. *IEEE transactions on cybernetics*, 43(3):820–828.

[Cooray et al., 2013] Cooray, D., Kouroshfar, E., Malek, S., and Roshandel, R. (2013). Proactive self-adaptation for improving the reliability of mission-critical, embedded, and mobile software. *Software Engineering, IEEE Transactions on*, 39(12):1714–1735.

[Coronato and De Pietro, 2012] Coronato, A. and De Pietro, G. (2012). Tools for the rapid prototyping of provably correct ambient intelligence applications. *Software Engineering, IEEE Transactions on*, 38(4):975–991.

[Dalpiaz et al., 2013] Dalpiaz, F., Giorgini, P., and Mylopoulos, J. (2013). Adaptive socio-technical systems: a requirements-based approach. *Requirements engineering*, 18(1):1–24.

[Das et al., 2017] Das, A., Panda, R., and Roy-Chowdhury, A. K. (2017). Continuous adaptation of multi-camera person identification models through sparse non-redundant representative selection. *Computer Vision and Image Understanding*, 156:66–78.

[Das et al., 2013] Das, S., Hayashi, E., and Hong, J. I. (2013). Exploring capturable everyday memory for autobiographical authentication. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 211–220. ACM.

[Dawadi et al., 2016] Dawadi, P. N., Cook, D. J., and Schmitter-Edgecombe, M. (2016). Modeling patterns of activities using activity curves. *Pervasive and mobile computing*, 28:51–68.

[De Lemos et al., 2013] De Lemos, R., Giese, H., Müller, H. A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N. M., Vogel, T., et al. (2013). Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer.

[De Masi et al., 2016] De Masi, A., Ciman, M., Gustarini, M., and Wac, K. (2016). mqol smart lab: quality of life living lab for interdisciplinary experiments. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 635–640. ACM.

[Degeler et al., 2014] Degeler, V., Lazovik, A., Leotta, F., and Mecella, M. (2014). Itemset-based mining of constraints for enacting smart environments. In *Pervasive Computing*

and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on, pages 41–46. IEEE.

[Denning et al., 2009] Denning, T., Matuszek, C., Koscher, K., Smith, J. R., and Kohno, T. (2009). A spotlight on security and privacy risks with future household robots: attacks and lessons. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 105–114. ACM.

[Dey, 2001] Dey, A. K. (2001). Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7.

[Driver and Clarke, 2008] Driver, C. and Clarke, S. (2008). An application framework for mobile, context-aware trails. *Pervasive and Mobile Computing*, 4(5):719–736.

[Dybå and Dingsøyr, 2008] Dybå, T. and Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9):833–859.

[Dyba et al., 2007] Dyba, T., Dingsoyr, T., and Hanssen, G. K. (2007). Applying systematic reviews to diverse study types: An experience report. In *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 225–234. IEEE Computer Society.

[Edwards and Grinter, 2001] Edwards, W. K. and Grinter, R. E. (2001). At home with ubiquitous computing: Seven challenges. In *Ubicomp 2001: Ubiquitous Computing*, pages 256–272. Springer.

[Eklund and Bosch, 2012] Eklund, U. and Bosch, J. (2012). Architecture for large-scale innovation experiment systems. In *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on*, pages 244–248. IEEE.

[Esfahani et al., 2011] Esfahani, N., Kouroshfar, E., and Malek, S. (2011). Taming uncertainty in self-adaptive software. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 234–244. ACM.

[Esling and Agon, 2012] Esling, P. and Agon, C. (2012). Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12.

[Esposito et al., 2018] Esposito, M., Minutolo, A., Megna, R., Forastiere, M., Magliulo, M., and De Pietro, G. (2018). A smart mobile, self-configuring, context-aware architecture for personal health monitoring. *Engineering Applications of Artificial Intelligence*, 67:136–156.

[European Commission (Eurostat), 2017] European Commission (Eurostat) (2017). Household composition statistics.

[Fagerholm et al., 2013] Fagerholm, F., Oza, N., and Münch, J. (2013). A platform for teaching applied distributed software development: The ongoing journey of the helsinki software factory. In *Collaborative Teaching of Globally Distributed Software Development (CTGDSD), 2013 3rd International Workshop on*, pages 1–5. IEEE.

[Farrahi and Gatica-Perez, 2008a] Farrahi, K. and Gatica-Perez, D. (2008a). Discovering human routines from cell phone data with topic models. In *Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on*, pages 29–32. IEEE.

[Farrahi and Gatica-Perez, 2008b] Farrahi, K. and Gatica-Perez, D. (2008b). What did you do today?: discovering daily routines from large-scale mobile data. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 849–852. ACM.

[Favela et al., 2010] Favela, J., Tentori, M., and Gonzalez, V. M. (2010). Ecological validity and pervasiveness in the evaluation of ubiquitous computing technologies for health care. *Intl. Journal of Human–Computer Interaction*, 26(5):414–444.

[Ferrara et al., 2014] Ferrara, M., Franco, A., and Maio, D. (2014). On the use of the kinect sensor for human identification in smart environments. *Journal of Ambient Intelligence and Smart Environments*, 6(4):435–446.

[Forte et al., 2008] Forte, M., de Souza, W. L., and do Prado, A. F. (2008). Using ontologies and web services for content adaptation in ubiquitous computing. *Journal of Systems and Software*, 81(3):368–381.

[Frank et al., 2010] Frank, J., Mannor, S., and Precup, D. (2010). Activity and gait recognition with time-delay embeddings. In *AAAI*.

[Füller et al., 2012] Füller, M., Nüßer, W., and Rustemeyer, T. (2012). Context driven process selection and integration of mobile and pervasive systems. *Pervasive and Mobile Computing*, 8(3):467–482.

[Gafurov et al., 2011] Gafurov, D., Bours, P., and Snekkenes, E. (2011). User authentication based on foot motion. *Signal, Image and Video Processing*, 5(4):457.

[Galushka et al., 2006] Galushka, M., Patterson, D., and Rooney, N. (2006). Temporal data mining for smart homes. In *Designing Smart Homes*, pages 85–108. Springer.

[Gámez and Fuentes, 2011] Gámez, N. and Fuentes, L. (2011). Famiware: A family of event-based middleware for ambient intelligence. *Personal and Ubiquitous Computing*, 15(4):329–339.

[Gamez and Fuentes, 2013] Gamez, N. and Fuentes, L. (2013). Architectural evolution of famiware using cardinality-based feature models. *Information and Software Technology*, 55(3):563–580.

[Garnier-Moiroux et al., 2013] Garnier-Moiroux, D., Silveira, F., and Sheth, A. (2013). Towards user identification in the home from appliance usage patterns. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 861–868. ACM.

[Gehrke and Madden, 2004] Gehrke, J. and Madden, S. (2004). Query processing in sensor networks. *IEEE Pervasive Computing*, (1):46–55.

[Geihs et al., 2009] Geihs, K., Barone, P., Eliassen, F., Floch, J., Fricke, R., Gjorven, E., Hallsteinsen, S., Horn, G., Khan, M. U., Mamelli, A., et al. (2009). A comprehensive solution for application-level adaptation. *Software: Practice and Experience*, 39(4):385–422.

[Gonzalez and Amft, 2015] Gonzalez, L. I. L. and Amft, O. (2015). Mining relations and physical grouping of building-embedded sensors and actuators. In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, pages 2–10. IEEE.

[Gonzalez and Amft, 2016] Gonzalez, L. I. L. and Amft, O. (2016). Mining hierarchical relations in building management variables. *Pervasive and Mobile Computing*, 26:91–101.

[Gopalan and Znati, 2010] Gopalan, A. and Znati, T. (2010). Sara: A service architecture for resource aware ubiquitous environments. *Pervasive and Mobile Computing*, 6(1):1–20.

[Greenberg and Buxton, 2008] Greenberg, S. and Buxton, B. (2008). Usability evaluation considered harmful (some of the time). In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 111–120. ACM.

[Group et al., 2004] Group, G. W. et al. (2004). Grading quality of evidence and strength of recommendations. *BMJ: British Medical Journal*, 328(7454):1490.

[Gu et al., 2004] Gu, X., Nahrstedt, K., Messer, A., Greenberg, I., and Milojicic, D. (2004). Adaptive offloading for pervasive computing. *Pervasive Computing, IEEE*, 3(3):66–73.

[Gui et al., 2013] Gui, N., De Florio, V., and Holvoet, T. (2013). Transformer: an adaptation framework supporting contextual adaptation behavior composition. *Software: Practice and Experience*, 43(8):937–967.

[Gui et al., 2011] Gui, N., De Florio, V., Sun, H., and Blondia, C. (2011). Toward architecture-based context-aware deployment and adaptation. *Journal of Systems and Software*, 84(2):185–197.

[Guillén-Gámez et al., 2017] Guillén-Gámez, F. D., García-Magariño, I., Bravo-Agapito, J., Lacuesta, R., and Lloret, J. (2017). A proposal to improve the authentication process in m-health environments. *IEEE Access*, 5:22530–22544.

[Guo et al., 2010] Guo, B., Zhang, D., and Imai, M. (2010). Enabling user-oriented management for ubiquitous computing: The meta-design approach. *Computer Networks*, 54(16):2840–2855.

[Guo et al., 2011] Guo, B., Zhang, D., and Imai, M. (2011). Toward a cooperative programming framework for context-aware applications. *Personal and ubiquitous computing*, 15(3):221–233.

[Gupta et al., 2008] Gupta, R., Fang, G., Field, B., Steinbach, M., and Kumar, V. (2008). Quantitative evaluation of approximate frequent pattern mining algorithms. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 301–309. ACM.

[Gustarini et al., 2013] Gustarini, M., Ickin, S., and Wac, K. (2013). Evaluation of challenges in human subject studies in-the-wild using subjects' personal smartphones. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 1447–1456. ACM.

[Halfaker et al., 2015] Halfaker, A., Keyes, O., Kluver, D., Thebault-Spieker, J., Nguyen, T., Shores, K., Uduwage, A., and Warncke-Wang, M. (2015). User session identification

based on strong regularities in inter-activity time. In *Proceedings of the 24th International Conference on World Wide Web*, pages 410–418. International World Wide Web Conferences Steering Committee.

[Hallsteinsen et al., 2012] Hallsteinsen, S., Geihs, K., Paspallis, N., Eliassen, F., Horn, G., Lorenzo, J., Mamelli, A., and Papadopoulos, G. A. (2012). A development framework and methodology for self-adapting applications in ubiquitous computing environments. *Journal of Systems and Software*, 85(12):2840–2859.

[Hamilton, 1844] Hamilton, W. R. (1844). On quaternions; or on a new system of imaginaries in algebra. *Philosophical Magazine Series 3*, 25(163):10–13.

[Han et al., 2007] Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86.

[Harrington and Cahill, 2011] Harrington, A. and Cahill, V. (2011). Model-driven engineering of planning and optimisation algorithms for pervasive computing environments. *Pervasive and Mobile Computing*, 7(6):705–726.

[Hartmann, 2016] Hartmann, T. (2016). *Enabling Model-Driven Live Analytics For Cyber-Physical Systems: The Case of Smart Grids*. PhD thesis, University of Luxembourg, Luxembourg.

[Hartmann et al., 2014] Hartmann, T., Fouquet, F., Nain, G., Morin, B., Klein, J., and Le Traon, Y. (2014). Reasoning at runtime using time-distorted contexts: A models@ run. time based approach. In *Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering*, pages 586–591. Knowledge Systems Institute Graduate School, USA.

[Hartmann et al., 2015] Hartmann, T., Moawad, A., Fouquet, F., Reckinger, Y., Mouelhi, T., Klein, J., and Le Traon, Y. (2015). Suspicious electric consumption detection based on multi-profiling using live machine learning. In *Smart Grid Communications (SmartGridComm), 2015 IEEE International Conference on*, pages 891–896. IEEE.

[Hayashi et al., 2014] Hayashi, E., Maas, M., and Hong, J. I. (2014). Wave to me: user identification using body lengths and natural gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3453–3462. ACM.

[Hazlewood et al., 2011] Hazlewood, W. R., Stolterman, E., and Connelly, K. (2011). Issues in evaluating ambient displays in the wild: two case studies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 877–886. ACM.

[Heierman and Cook, 2003] Heierman, E. O. and Cook, D. J. (2003). Improving home automation by discovering regularly occurring device usage patterns. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 537–540. IEEE.

[Henderson and Clark, 1990] Henderson, R. M. and Clark, K. B. (1990). Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Administrative science quarterly*, pages 9–30.

[Henricksen and Indulska, 2006] Henricksen, K. and Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. *Pervasive and mobile computing*, 2(1):37–64.

[Hess and Campbell, 2003] Hess, C. K. and Campbell, R. H. (2003). An application of a context-aware file system. *Personal and Ubiquitous Computing*, 7(6):339–352.

[Hestbek et al., 2012] Hestbek, M. R., Nickel, C., and Busch, C. (2012). Biometric gait recognition for mobile devices using wavelet transform and support vector machines. In *Systems, Signals and Image Processing (IWSSIP), 2012 19th International Conference on*, pages 205–210. IEEE.

[Hoareau and Mahéo, 2008] Hoareau, D. and Mahéo, Y. (2008). Middleware support for the deployment of ubiquitous software components. *Personal and ubiquitous computing*, 12(2):167–178.

[Hoffmann and Söllner, 2014] Hoffmann, H. and Söllner, M. (2014). Incorporating behavioral trust theory into system development for ubiquitous applications. *Personal and ubiquitous computing*, 18(1):117–128.

[Holz and Knaust, 2015] Holz, C. and Knaust, M. (2015). Biometric touch sensing: Seamlessly augmenting each touch with continuous authentication. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 303–312. ACM.

[Holzmann and Ferscha, 2010] Holzmann, C. and Ferscha, A. (2010). A framework for utilizing qualitative spatial relations between networked embedded systems. *Pervasive and Mobile Computing*, 6(3):362–381.

[Hu et al., 2017] Hu, N., Englebienne, G., Lou, Z., and Kröse, B. (2017). Learning to recognize human activities using soft labels. *IEEE transactions on pattern analysis and machine intelligence*, 39(10):1973–1984.

[Hu et al., 2008] Hu, P., Indulska, J., and Robinson, R. (2008). An autonomic context management system for pervasive computing. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 213–223. IEEE.

[Huang et al., 2010] Huang, H.-M., Tidwell, T., Gill, C., Lu, C., Gao, X., and Dyke, S. (2010). Cyber-physical systems for real-time hybrid structural testing: a case study. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 69–78. ACM.

[Huebscher and McCann, 2008] Huebscher, M. C. and McCann, J. A. (2008). A survey of autonomic computing—degrees, models, and applications. *ACM Computing Surveys (CSUR)*, 40(3):7.

[Huynh et al., 2008] Huynh, T., Fritz, M., and Schiele, B. (2008). Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 10–19. ACM.

[Inoue, 2006] Inoue, K. (2006). Supporting a dynamic program signature: An intrusion detection framework for microprocessors. In *Electronics, Circuits and Systems, 2006. ICECS'06. 13th IEEE International Conference on*, pages 160–163. IEEE.

[Inverardi and Tivoli, 2013] Inverardi, P. and Tivoli, M. (2013). Automatic synthesis of modular connectors via composition of protocol mediation patterns. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 3–12. IEEE Press.

[Iosifidis et al., 2012] Iosifidis, A., Tefas, A., and Pitas, I. (2012). Activity-based person identification using fuzzy representation and discriminant learning. *IEEE Transactions on Information Forensics and Security*, 7(2):530–542.

[Iqbal et al., 2018] Iqbal, A., Ullah, F., Anwar, H., Kwak, K. S., Imran, M., Jamal, W., and ur Rahman, A. (2018). Interoperable internet-of-things platform for smart home system using web-of-objects and cloud. *Sustainable Cities and Society*, 38:636–646.

[Ishida et al., 2017] Ishida, A., Murao, K., Terada, T., and Tsukamoto, M. (2017). A user identification method based on features of opening/closing a refrigerator door. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 533–538. IEEE.

[Jakobsson et al., 2009] Jakobsson, M., Shi, E., Golle, P., and Chow, R. (2009). Implicit authentication for mobile devices. In *Proceedings of the 4th USENIX conference on Hot topics in security*, pages 9–9.

[Jaroucheh et al., 2012] Jaroucheh, Z., Liu, X., and Smith, S. (2012). An approach to domain-based scalable context management architecture in pervasive environments. *Personal and Ubiquitous Computing*, 16(6):741–755.

[Jorgensen and Yu, 2011] Jorgensen, Z. and Yu, T. (2011). On mouse dynamics as a behavioral biometric for authentication. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 476–482. ACM.

[Juefei-Xu et al., 2012] Juefei-Xu, F., Bhagavatula, C., Jaech, A., Prasad, U., and Savvides, M. (2012). Gait-id on the move: Pace independent human identification using cell phone accelerometer dynamics. In *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*, pages 8–15. IEEE.

[Julien and Roman, 2006] Julien, C. and Roman, G.-C. (2006). Egospaces: Facilitating rapid development of context-aware mobile applications. *Software Engineering, IEEE Transactions on*, 32(5):281–298.

[Juristo and Moreno, 2013] Juristo, N. and Moreno, A. M. (2013). *Basics of software engineering experimentation*. Springer Science & Business Media.

[Kanneh and Sakr, 2008] Kanneh, A. and Sakr, Z. (2008). Biometric user verification using haptics and fuzzy logic. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 937–940. ACM.

[Katsivelis et al., 2017] Katsivelis, N., Anastasiou, A., Petropoulou, O., Lambrou, G., Giokas, K., and Koutsouris, D. (2017). Applied technologies and smart home applications in the health sector. In *2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 544–549. IEEE.

[Kawamoto et al., 2017] Kawamoto, Y., Yamada, N., Nishiyama, H., Kato, N., Shimizu, Y., and Zheng, Y. (2017). A feedback control-based crowd dynamics management in iot system. *IEEE Internet of Things Journal*, 4(5):1466–1476.

[Kawsar et al., 2008] Kawsar, F., Nakajima, T., and Fujinami, K. (2008). Deploy spontaneously: supporting end-users in building and enhancing a smart home. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 282–291. ACM.

[Kiani et al., 2013] Kiani, S. L., Anjum, A., Knappmeyer, M., Bessis, N., and Antonopoulos, N. (2013). Federated broker system for pervasive context provisioning. *Journal of Systems and Software*, 86(4):1107–1123.

[Kibria et al., 2017] Kibria, M. G., Jarwar, M. A., Ali, S., Kumar, S., and Chong, I. (2017). Web objects based energy efficiency for smart home iot service provisioning. In *Ubiquitous and Future Networks (ICUFN), 2017 Ninth International Conference on*, pages 55–60. IEEE.

[Kidd et al., 1999] Kidd, C. D., Orr, R., Abowd, G. D., Atkeson, C. G., Essa, I. A., MacIntyre, B., Mynatt, E., Starner, T. E., and Newstetter, W. (1999). The aware home: A living laboratory for ubiquitous computing research. In *International Workshop on Cooperative Buildings*, pages 191–198. Springer.

[Kim et al., 2010] Kim, E., Helal, S., and Cook, D. (2010). Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1).

[Kiran et al., 2016] Kiran, R. U., Kitsuregawa, M., and Reddy, P. K. (2016). Efficient discovery of periodic-frequent patterns in very large databases. *Journal of Systems and Software*, 112:110–121.

[Kiran et al., 2017] Kiran, R. U., Venkatesh, J., Toyoda, M., Kitsuregawa, M., and Reddy, P. K. (2017). Discovering partial periodic-frequent patterns in a transactional database. *Journal of Systems and Software*, 125:170–182.

[Kitchenham and Brereton, 2013] Kitchenham, B. and Brereton, P. (2013). A systematic review of systematic review process research in software engineering. *Information and software technology*, 55(12):2049–2075.

[Kitchenham and Charters, 2007] Kitchenham, B. and Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.

[Kohavi et al., 2009] Kohavi, R., Crook, T., and Longbotham, R. (2009). Online experimentation at microsoft. *Data Mining Case Studies*, 11.

[Kohavi et al., 2012] Kohavi, R., Deng, A., Frasca, B., Longbotham, R., Walker, T., and Xu, Y. (2012). Trustworthy online controlled experiments: Five puzzling outcomes explained. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 786–794. ACM.

[Kohavi et al., 2013] Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y., and Pohlmann, N. (2013). Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1168–1176. ACM.

[Koreshoff et al., 2013] Koreshoff, T. L., Robertson, T., and Leong, T. W. (2013). Internet of things: a review of literature and products. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, pages 335–344. ACM.

[Krupitzer et al., 2015] Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., and Becker, C. (2015). A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, 17:184–206.

[Kulkarni et al., 2012] Kulkarni, D., Ahmed, T., and Tripathi, A. (2012). A generative programming framework for context-aware cscw applications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21(2):11.

[Kuncheva, 2004] Kuncheva, L. I. (2004). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

[Kwapisz et al., 2010] Kwapisz, J. R., Weiss, G. M., and Moore, S. A. (2010). Cell phone-based biometric identification. In *2010 Fourth IEEE International Conference on Biometrics: Theory Applications and Systems (BTAS)*, pages 1–7. IEEE.

[Lester et al., 2009] Lester, J., Hartung, C., Pina, L., Libby, R., Borriello, G., and Duncan, G. (2009). Validated caloric expenditure estimation using a single body-worn sensor. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 225–234. ACM.

[Lézoray et al., 2011] Lézoray, J.-B., Segarra, M.-T., Phung-Khac, A., Thépaut, A., Gilliot, J.-M., and Beugnard, A. (2011). A design process enabling adaptation in pervasive heterogeneous contexts. *Personal and ubiquitous computing*, 15(4):353–363.

[Li et al., 2016] Li, S., Ashok, A., Zhang, Y., Xu, C., Lindqvist, J., and Gruteser, M. (2016). Whose move is it anyway? authenticating smart wearable devices using unique head movement patterns. In *Pervasive Computing and Communications (PerCom), 2016 IEEE International Conference on*, pages 1–9. IEEE.

[Liebling and Preibusch, 2014] Liebling, D. J. and Preibusch, S. (2014). Privacy considerations for a pervasive eye tracking world. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 1169–1177. ACM.

[Liker and Franz, 2011] Liker, J. K. and Franz, J. K. (2011). *The Toyota way to continuous improvement: Linking strategy and operational excellence to achieve superior performance*, volume 1. McGraw-Hill New York.

[Liogkas et al., 2004] Liogkas, N., MacIntyre, B., Mynatt, E., Smaragdakis, Y., Tilevich, E., and Voida, S. (2004). Automatic partitioning for prototyping ubiquitous computing applications. *Pervasive Computing, IEEE*, 3(3):40–47.

[Liu and Cheng, 2011] Liu, S. and Cheng, L. (2011). A context-aware reflective middleware framework for distributed real-time and embedded systems. *Journal of Systems and Software*, 84(2):205–218.

[Liu et al., 2013] Liu, Y., Xu, C., and Cheung, S. C. (2013). Afchecker: Effective model checking for context-aware adaptive applications. *Journal of Systems and Software*, 86(3):854–867.

[Lockhart and Weiss, 2014] Lockhart, J. W. and Weiss, G. M. (2014). The benefits of personalized smartphone-based activity recognition models. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 614–622. SIAM.

[Louis et al., 2016] Louis, W., Komeili, M., and Hatzinakos, D. (2016). Continuous authentication using one-dimensional multi-resolution local binary patterns (1dmrlbp) in ecg biometrics. *IEEE Transactions on Information Forensics and Security*, 11(12):2818–2832.

[Lu et al., 2006] Lu, H., Chan, W., and Tse, T. (2006). Testing context-aware middleware-centric programs: a data flow approach and an rfid-based experimentation. In *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 242–252. ACM.

[Lu et al., 2008] Lu, H., Chan, W., and Tse, T. (2008). Testing pervasive software in the presence of context inconsistency resolution services. In *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*, pages 61–70. IEEE.

[Lu et al., 2014] Lu, H., Huang, J., Saha, T., and Nachman, L. (2014). Unobtrusive gait verification for mobile phones. In *Proceedings of the 2014 ACM international symposium on wearable computers*, pages 91–98. ACM.

[Maekawa et al., 2016] Maekawa, T., Masuda, A., and Namioka, Y. (2016). Identification from ceiling: unconstrained person identification for tabletops using multiview learning. In *Proceedings of the 15th International Conference on Mobile and Ubiquitous Multimedia*, pages 273–284. ACM.

[Malandrino et al., 2010] Malandrino, D., Mazzoni, F., Riboni, D., Bettini, C., Colajanni, M., and Scarano, V. (2010). Mimosa: context-aware adaptation for ubiquitous web access. *Personal and ubiquitous computing*, 14(4):301–320.

[Malatras et al., 2017] Malatras, A., Geneiatakis, D., and Vakalis, I. (2017). On the efficiency of user identification: a system-based approach. *International Journal of Information Security*, 16(6):653–671.

[Malek et al., 2010] Malek, S., Edwards, G., Brun, Y., Tajalli, H., Garcia, J., Krka, I., Medvidovic, N., Mikic-Rakic, M., and Sukhatme, G. S. (2010). An architecture-driven software mobility framework. *Journal of Systems and Software*, 83(6):972–989.

[Malek et al., 2005] Malek, S., Mikic-Rakic, M., and Medvidovic, N. (2005). A style-aware architectural middleware for resource-constrained, distributed systems. *Software Engineering, IEEE Transactions on*, 31(3):256–272.

[Mallat, 2008] Mallat, S. (2008). *A wavelet tour of signal processing: the sparse way*. Academic press.

[Mamei and Zambonelli, 2004] Mamei, M. and Zambonelli, F. (2004). Programming pervasive and mobile computing applications with the tota middleware. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 263–273. IEEE.

[Mamei and Zambonelli, 2009] Mamei, M. and Zambonelli, F. (2009). Programming pervasive and mobile computing applications: The tota approach. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 18(4):15.

[Mancini et al., 2017] Mancini, C., Lawson, S., and Juhlin, O. (2017). Animal-computer interaction: The emergence of a discipline.

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1(3):259–289.

[Marcel and Millán, 2007] Marcel, S. and Millán, J. d. R. (2007). Person authentication using brainwaves (eeg) and maximum a posteriori model adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 29(4).

[Martin et al., 2011] Martin, S., Diaz, G., Plaza, I., Ruiz, E., Castro, M., and Peire, J. (2011). State of the art of frameworks and middleware for facilitating mobile and ubiquitous learning development. *Journal of Systems and Software*, 84(11):1883–1891.

[Martinovic et al., 2017] Martinovic, I., Rasmussen, K., Roeschlin, M., and Tsudik, G. (2017). Authentication using pulse-response biometrics. *Communications of the ACM*, 60(2):108–115.

[Martinson et al., 2010] Martinson, E., Lawson, W., and Trafton, J. G. (2010). Person identification through perceptual fusion. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 358–364. IEEE.

[Matovski et al., 2012] Matovski, D. S., Nixon, M. S., Mahmoodi, S., and Carter, J. N. (2012). The effect of time on gait recognition performance. *IEEE transactions on information forensics and security*, 7(2):543–552.

[McGlinn et al., 2014] McGlinn, K., Hederman, L., and Lewis, D. (2014). Simcon: A context simulator for supporting evaluation of smart building applications when faced with uncertainty. *Pervasive and Mobile Computing*, 12:139–159.

[Meier and Cahil, 2010] Meier, R. and Cahil, V. (2010). On event-based middleware for location-aware mobile applications. *Software Engineering, IEEE Transactions on*, 36(3):409–430.

[Meier et al., 2009] Meier, R., Harrington, A., Beckmann, K., and Cahill, V. (2009). A framework for incremental construction of real global smart space applications. *Pervasive and Mobile Computing*, 5(4):350–368.

[Meng, 2012] Meng, Y. (2012). Designing click-draw based graphical password scheme for better authentication. In *Networking, Architecture and Storage (NAS), 2012 IEEE 7th International Conference on*, pages 39–48. IEEE.

[Moawad et al., 2015] Moawad, A., Hartmann, T., Fouquet, F., Nain, G., Klein, J., and Le Traon, Y. (2015). Beyond discrete modeling: A continuous and efficient model for iot. In *Model Driven Engineering Languages and Systems (MODELS), 2015 ACM/IEEE 18th International Conference on*, pages 90–99. IEEE.

[Mondal and Bours, 2016] Mondal, S. and Bours, P. (2016). Combining keystroke and mouse dynamics for continuous user authentication and identification. In *Identity, Security and Behavior Analysis (ISBA), 2016 IEEE International Conference on*, pages 1–8. IEEE.

[Montoliu and Gatica-Perez, 2010] Montoliu, R. and Gatica-Perez, D. (2010). Discovering human places of interest from multimodal mobile phone data. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, page 12. ACM.

[Moreno et al., 2017] Moreno, L. V., Ruiz, M. L. M., Hernández, J. M., Duboy, M. Á. V., and Lindén, M. (2017). The role of smart homes in intelligent homecare and healthcare environments. In *Ambient Assisted Living and Enhanced Living Environments*, pages 345–394. Elsevier.

[Morin et al., 2009] Morin, B., Barais, O., Nain, G., and Jezequel, J.-M. (2009). Taming dynamically adaptive systems using models and aspects. In *Proceedings of the 31st International Conference on Software Engineering*, pages 122–132. IEEE Computer Society.

[Morla and Davies, 2004] Morla, R. and Davies, N. (2004). Evaluating a location-based application: A hybrid test and simulation environment. *IEEE Pervasive computing*, (3):48–56.

[Moros et al., 2013] Moros, B., Toval, A., Rosique, F., and Sánchez, P. (2013). Transforming and tracing reused requirements models to home automation models. *Information and Software Technology*, 55(6):941–965.

[Morris et al., 2015] Morris, K. A., Allison, M., Costa, F. M., Wei, J., and Clarke, P. J. (2015). An adaptive middleware design to support the dynamic interpretation of domain-specific models. *Information and Software Technology*, 62:21–41.

[Müller, 2004] Müller, M. E. (2004). Can user models be learned at all? inherent problems in machine learning for user modelling. *The Knowledge Engineering Review*, 19(1):61–88.

[Münch et al., 2013] Münch, J., Fagerholm, F., Kettunen, P., Pagels, M., and Partanen, J. (2013). Experiences and insights from applying gqm+ strategies in a systems product development organisation. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, pages 70–77. IEEE.

[Munnelly et al., 2007] Munnelly, J., Fritsch, S., and Clarke, S. (2007). An aspect-oriented approach to the modularisation of context. In *Pervasive Computing and Communications, 2007. PerCom'07. Fifth Annual IEEE International Conference on*, pages 114–124. IEEE.

[Murukannaiah and Singh, 2015] Murukannaiah, P. K. and Singh, M. P. (2015). Platys: an active learning framework for place-aware application development and its evaluation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):19.

[Nahrstedt et al., 2005] Nahrstedt, K., Yu, B., Liang, J., and Cui, Y. (2005). Hourglass multimedia content and service composition framework for smart room environments. *Pervasive and Mobile Computing*, 1(1):43–75.

[Nguyen et al., 2015] Nguyen, L. T., Zeng, M., Tague, P., and Zhang, J. (2015). Recognizing new activities with limited training data. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 67–74. ACM.

[Nishikawa et al., 2006] Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., and Ito, M. (2006). Ubireal: realistic smartspace simulator for systematic testing. In *UbiComp 2006: Ubiquitous Computing*, pages 459–476. Springer.

[Nyan et al., 2006] Nyan, M., Tay, F., Seah, K., and Sitoh, Y. (2006). Classification of gait patterns in the time–frequency domain. *Journal of biomechanics*, 39(14):2647–2656.

[Ohno, 1988] Ohno, T. (1988). *Toyota production system: beyond large-scale production*. crc Press.

[Olsson et al., 2012] Olsson, H. H., Alahyari, H., and Bosch, J. (2012). Climbing the" stairway to heaven"–a mulitiple-case study exploring barriers in the transition from

agile development towards continuous deployment of software. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pages 392–399. IEEE.

[Olsson and Bosch, 2014] Olsson, H. H. and Bosch, J. (2014). The hypex model: from opinions to data-driven software development. In *Continuous software engineering*, pages 155–164. Springer.

[O'Neill et al., 2013] O'Neill, E., Conlan, O., and Lewis, D. (2013). Situation-based testing for pervasive computing environments. *Pervasive and Mobile Computing*, 9(1):76–97.

[Organisation for Economic Co-operation and Development (OECD), 2016] Organisation for Economic Co-operation and Development (OECD) (2016). Five family facts.

[Ou et al., 2007] Ou, S., Yang, K., and Zhang, J. (2007). An effective offloading middleware for pervasive services on mobile devices. *Pervasive and Mobile Computing*, 3(4):362–385.

[Pal et al., 2018] Pal, D., Triyason, T., Funilkul, S., and Chutimaskul, W. (2018). Smart homes and quality of life for the elderly: Perspective of competing models. *IEEE Access*, 6:8109–8122.

[Pallapa et al., 2014] Pallapa, G., Das, S. K., Di Francesco, M., and Aura, T. (2014). Adaptive and context-aware privacy preservation exploiting user interactions in smart environments. *Pervasive and Mobile Computing*, 12:232–243.

[Paluska et al., 2008] Paluska, J. M., Pham, H., Saif, U., Chau, G., Terman, C., and Ward, S. (2008). Structured decomposition of adaptive applications. *Pervasive and Mobile Computing*, 4(6):791–806.

[Park et al., 2018] Park, E., Kim, S., Kim, Y., and Kwon, S. J. (2018). Smart home services as the next mainstream of the ict industry: determinants of the adoption of smart home services. *Universal Access in the Information Society*, 17(1):175–190.

[Paspallis and Papadopoulos, 2014] Paspallis, N. and Papadopoulos, G. A. (2014). A pluggable middleware architecture for developing context-aware mobile applications. *Personal and Ubiquitous Computing*, 18(5):1099–1116.

[Payton et al., 2012] Payton, J., Julien, C., Rajamani, V., and Roman, G.-C. (2012). Using snapshot query fidelity to adapt continuous query execution. *Pervasive and Mobile Computing*, 8(3):317–330.

[Payton et al., 2010] Payton, J., Julien, C., Roman, G.-C., and Rajamani, V. (2010). Semantic self-assessment of query results in dynamic environments. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 19(4):12.

[Perry et al., 2004] Perry, D. E., Sim, S. E., and Easterbrook, S. M. (2004). Case studies for software engineers. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 736–738. IEEE.

[Pham et al., 2009] Pham, H., Paluska, J. M., Saif, U., Stawarz, C., Terman, C., and Ward, S. (2009). A dynamic platform for run-time adaptation. *Pervasive and Mobile Computing*, 5(6):676–696.

[Poghosyan et al., 2017] Poghosyan, G., Pefkianakis, I., Le Guyadec, P., and Christophides, V. (2017). Extracting usage patterns of home iot devices. In *Computers and Communications (ISCC), 2017 IEEE Symposium on*, pages 1318–1324. IEEE.

[Pohl et al., 2015] Pohl, H., Krause, M., and Rohs, M. (2015). One-button recognizer: exploiting button pressing behavior for user differentiation. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 403–407. ACM.

[Poladian et al., 2004] Poladian, V., Sousa, J. P., Garlan, D., and Shaw, M. (2004). Dynamic configuration of resource-aware services. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 604–613. IEEE.

[Poppendieck and Cusumano, 2012] Poppendieck, M. and Cusumano, M. A. (2012). Lean software development: A tutorial. *IEEE software*, 29(5):26–32.

[Poppendieck and Poppendieck, 2003] Poppendieck, M. and Poppendieck, T. (2003). *Lean software development: an agile toolkit*. Addison-Wesley.

[Rahman et al., 2013] Rahman, K. A., Balagani, K. S., and Phoha, V. V. (2013). Snoop-forge-replay attacks on continuous verification with keystrokes. *IEEE Transactions on information forensics and security*, 8(3):528–541.

[Ranjan, 2015] Ranjan, J. (2015). Object user recognition in multi-person homes. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 477–482. ACM.

[Rashid et al., 2012] Rashid, J., Broxvall, M., and Saffiotti, A. (2012). A middleware to integrate robots, simple devices and everyday objects into an ambient ecology. *Pervasive and Mobile Computing*, 8(4):522–541.

[Rashidi et al., 2011] Rashidi, P., Cook, D. J., Holder, L. B., and Schmitter-Edgecombe, M. (2011). Discovering activities to recognize and track in a smart environment. *IEEE transactions on knowledge and data engineering*, 23(4):527–539.

[Raychoudhury et al., 2013] Raychoudhury, V., Cao, J., Kumar, M., and Zhang, D. (2013). Middleware for pervasive computing: A survey. *Pervasive and Mobile Computing*, 9(2):177–200.

[Ries, 2011] Ries, E. (2011). *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Books.

[Rioul and Duhamel, 1992] Rioul, O. and Duhamel, P. (1992). Fast algorithms for discrete and continuous wavelet transforms. *IEEE transactions on information theory*, 38(2):569–586.

[Riva and Toivonen, 2007] Riva, O. and Toivonen, S. (2007). The dynamos approach to support context-aware service provisioning in mobile environments. *Journal of Systems and Software*, 80(12):1956–1972.

[Robinson et al., 2008] Robinson, J., Wakeman, I., and Chalmers, D. (2008). Composing software services in the pervasive computing environment: Languages or apis? *Pervasive and Mobile Computing*, 4(4):481–505.

[Romero et al., 2013] Romero, D., Hermosillo, G., Taherkordi, A., Nzekwa, R., Rouvoy, R., and Eliassen, F. (2013). The digihome service-oriented platform. *Software: Practice and Experience*, 43(10):1205–1218.

[Roth et al., 2015] Roth, J., Liu, X., Ross, A., and Metaxas, D. (2015). Investigating the discriminative power of keystroke sound. *IEEE Transactions on Information Forensics and Security*, 10(2):333–345.

[Roussaki et al., 2010] Roussaki, I., Strimpakou, M., Pils, C., Kalatzis, N., and Liampotis, N. (2010). Optimising context data dissemination and storage in distributed pervasive computing systems. *Pervasive and Mobile Computing*, 6(2):218–238.

[Ruiz-López et al., 2013] Ruiz-López, T., Noguera, M., RodríGuez, M. J., Garrido, J. L., and Chung, L. (2013). Reubi: A requirements engineering method for ubiquitous systems. *Science of Computer Programming*, 78(10):1895–1911.

[Sae-Bae et al., 2012] Sae-Bae, N., Ahmed, K., Isbister, K., and Memon, N. (2012). Biometric-rich gestures: a novel approach to authentication on multi-touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 977–986. ACM.

[Salifu et al., 2012] Salifu, M., Yu, Y., Bandara, A. K., and Nuseibeh, B. (2012). Analysing monitoring and switching problems for adaptive systems. *Journal of Systems and Software*, 85(12):2829–2839.

[Salvaneschi et al., 2012] Salvaneschi, G., Ghezzi, C., and Pradella, M. (2012). Contexterlang: introducing context-oriented programming in the actor model. In *Proceedings of the 11th annual international conference on Aspect-oriented Software Development*, pages 191–202. ACM.

[Sama et al., 2010] Sama, M., Elbaum, S., Raimondi, F., Rosenblum, D. S., and Wang, Z. (2010). Context-aware adaptive applications: Fault patterns and their automated identification. *Software Engineering, IEEE Transactions on*, 36(5):644–661.

[Sama et al., 2008] Sama, M., Rosenblum, D. S., Wang, Z., and Elbaum, S. (2008). Model-based fault detection in context-aware adaptive applications. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 261–271. ACM.

[Sanderson and Erbetta, 2000] Sanderson, S. and Erbetta, J. (2000). Authentication for secure environments based on iris scanning technology. In *Visual Biometrics (Ref. No. 2000/018), IEE Colloquium on*, pages 8–1. IET.

[Schreiber et al., 2012] Schreiber, F., Camplani, R., Fortunato, M., Marelli, M., Rota, G., et al. (2012). Perla: A language and middleware architecture for data management and integration in pervasive information systems. *Software Engineering, IEEE Transactions on*, 38(2):478–496.

[Schuhmann et al., 2010] Schuhmann, S., Herrmann, K., and Rothermel, K. (2010). Efficient resource-aware hybrid configuration of distributed pervasive applications. In *Pervasive Computing*, pages 373–390. Springer.

[Schuhmann et al., 2013] Schuhmann, S., Herrmann, K., Rothermel, K., and Boshmaf, Y. (2013). Adaptive composition of distributed pervasive applications in heterogeneous environments. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 8(2):10.

[Schürmann et al., 2017] Schürmann, D., Brüsch, A., Sigg, S., and Wolf, L. (2017). Bandana—body area network device-to-device authentication using natural gait. In *Pervasive Computing and Communications (PerCom), 2017 IEEE International Conference on*, pages 190–196. IEEE.

[Seinturier et al., 2012] Seinturier, L., Merle, P., Rouvoy, R., Romero, D., Schiavoni, V., and Stefani, J.-B. (2012). A component-based middleware platform for reconfigurable service-oriented architectures. *Software: Practice and Experience*, 42(5):559–583.

[Seiter et al., 2015] Seiter, J., Chiu, W.-C., Fritz, M., Amft, O., and Tröster, G. (2015). Joint segmentation and activity discovery using semantic and temporal priors. In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, pages 71–78. IEEE.

[Serral et al., 2010] Serral, E., Valderas, P., and Pelechano, V. (2010). Towards the model driven development of context-aware pervasive systems. *Pervasive and Mobile Computing*, 6(2):254–280.

[Sethi et al., 2014] Sethi, M., Oat, E., Di Francesco, M., and Aura, T. (2014). Secure bootstrapping of cloud-managed ubiquitous displays. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 739–750. ACM.

[Shen et al., 2018] Shen, Y., Wen, H., Luo, C., Xu, W., Zhang, T., Hu, W., and Rus, D. (2018). Gaitlock: Protect virtual and augmented reality headsets using gait. *IEEE Transactions on Dependable and Secure Computing*.

[Shi et al., 2017] Shi, C., Liu, J., Liu, H., and Chen, Y. (2017). Smart user authentication through actuation of daily activities leveraging wifi-enabled iot. In *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, page 5. ACM.

[Shi et al., 2011a] Shi, J., Wan, J., Yan, H., and Suo, H. (2011a). A survey of cyber-physical systems. In *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, pages 1–6. IEEE.

[Shi et al., 2011b] Shi, W., Yang, J., Jiang, Y., Yang, F., and Xiong, Y. (2011b). Senguard: Passive user identification on smartphones using multiple sensors. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*, pages 141–148. IEEE.

[shimmer, 2017] shimmer (2017). Shimmer3 wireless sensor platform. http://www.shimmersensing.com/images/uploads/docs/Shimmer3_Spec_Sheet_V1.6.pdf. [Online; last-accessed 1 September 2017].

[Silva et al., 2013] Silva, S. E., Calado, R. D., Silva, M. B., and Nascimento, M. (2013). Lean startup applied in healthcare: A viable methodology for continuous improvement in the development of new products and services. *IFAC Proceedings Volumes*, 46(24):295–299.

[Sim et al., 2007] Sim, T., Zhang, S., Janakiraman, R., and Kumar, S. (2007). Continuous verification using multimodal biometrics. *IEEE transactions on pattern analysis and machine intelligence*, 29(4):687–700.

[Sitová et al., 2016] Sitová, Z., Šeděnka, J., Yang, Q., Peng, G., Zhou, G., Gasti, P., and Balagani, K. S. (2016). Hmog: New behavioral biometric features for continuous authentication of smartphone users. *IEEE Transactions on Information Forensics and Security*, 11(5):877–892.

[Solaimani et al., 2015] Solaimani, S., Keijzer-Broers, W., and Bouwman, H. (2015). What we do–and don't–know about the smart home: An analysis of the smart home literature. *Indoor and Built Environment*, 24(3):370–383.

[Soldatos et al., 2007] Soldatos, J., Dimakis, N., Stamatis, K., and Polymenakos, L. (2007). A breadboard architecture for pervasive context-aware services in smart spaces: middleware components and prototype applications. *Personal and Ubiquitous Computing*, 11(3):193–212.

[Spínola and Travassos, 2012] Spínola, R. O. and Travassos, G. H. (2012). Towards a framework to characterize ubiquitous software projects. *Information and Software Technology*, 54(7):759–785.

[Srikant and Agrawal, 1996] Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. *Advances in Database Technology—EDBT'96*, pages 1–17.

[Ståhl and Bosch, 2014] Ståhl, D. and Bosch, J. (2014). Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software*, 87:48–59.

[Steiber and Alänge, 2013] Steiber, A. and Alänge, S. (2013). A corporate system for continuous innovation: the case of google inc. *European Journal of Innovation Management*, 16(2):243–264.

[Sultana et al., 2014] Sultana, M., Paul, P. P., and Gavrilova, M. (2014). A concept of social behavioral biometrics: motivation, current developments, and future trends. In *Cyberworlds (CW), 2014 International Conference on*, pages 271–278. IEEE.

[Sun et al., 2014] Sun, F.-T., Yeh, Y.-T., Cheng, H.-T., Kuo, C., and Griss, M. (2014). Nonparametric discovery of human routines from sensor data. In *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*, pages 11–19. IEEE.

[Surana et al., 2011] Surana, A., Kiran, R. U., and Reddy, P. K. (2011). An efficient approach to mine periodic-frequent patterns in transactional databases. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 254–266. Springer.

[Szvetits and Zdun, 2013] Szvetits, M. and Zdun, U. (2013). Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Software & Systems Modeling*, pages 1–39.

[Takayama, 2017] Takayama, L. (2017). The motivations of ubiquitous computing: revisiting the ideas behind and beyond the prototypes. *Personal and Ubiquitous Computing*, 21(3):557–569.

[Tanbeer et al., 2009] Tanbeer, S. K., Ahmed, C. F., Jeong, B.-S., and Lee, Y.-K. (2009). Discovering periodic-frequent patterns in transactional databases. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 242–253. Springer.

[Tang et al., 2010] Tang, D., Agarwal, A., O'Brien, D., and Meyer, M. (2010). Overlapping experiment infrastructure: More, better, faster experimentation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 17–26. ACM.

[Tang et al., 2011] Tang, L., Yu, Z., Zhou, X., Wang, H., and Becker, C. (2011). Supporting rapid design and evaluation of pervasive applications: challenges and solutions. *Personal and ubiquitous computing*, 15(3):253–269.

[Tax and Duin, 2002] Tax, D. M. and Duin, R. P. (2002). Using two-class classifiers for multiclass classification. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 124–127. IEEE.

[Thomas et al., 2014] Thomas, K., Bandara, A. K., Price, B. A., and Nuseibeh, B. (2014). Distilling privacy requirements for mobile applications. In *Proceedings of the 36th International Conference on Software Engineering*, pages 871–882. ACM.

[Toch et al., 2018] Toch, E., Bettini, C., Shmueli, E., Radaelli, L., Lanzi, A., Riboni, D., and Lepri, B. (2018). The privacy implications of cyber security systems: A technological survey. *ACM Computing Surveys (CSUR)*, 51(2):36.

[Turcu and Turcu, 2018] Turcu, C. E. and Turcu, C. O. (2018). New perspectives on sustainable healthcare delivery through web of things. In *Handbook of Research on Contemporary Perspectives on Web-Based Systems*, pages 166–187. IGI Global.

[United Nations (UN), 2017] United Nations (UN) (2017). Household size and composition around the world 2017.

[Unterkalmsteiner et al., 2012] Unterkalmsteiner, M., Gorschek, T., Cheng, C. K., Permadi, R. B., Feldt, R., et al. (2012). Evaluation and measurement of software process improvement—a systematic literature review. *Software Engineering, IEEE Transactions on*, 38(2):398–424.

[Van Kasteren et al., 2008] Van Kasteren, T., Noulas, A., Englebienne, G., and Kröse, B. (2008). Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 1–9. ACM.

[VanSyckel et al., 2014] VanSyckel, S., Schäfer, D., Majuntke, V., Krupitzer, C., Schiele, G., and Becker, C. (2014). Comity: A framework for adaptation coordination in multi-platform pervasive systems. *Pervasive and Mobile Computing*, 10:51–65.

[Venkatesh et al., 2016] Venkatesh, J., Kiran, R. U., Reddy, P. K., and Kitsuregawa, M. (2016). Discovering periodic-frequent patterns in transactional databases using all-confidence and periodic-all-confidence. In *International Conference on Database and Expert Systems Applications*, pages 55–70. Springer.

[Verbelen et al., 2011] Verbelen, T., Stevens, T., Simoens, P., De Turck, F., and Dhoedt, B. (2011). Dynamic deployment and quality adaptation for mobile augmented reality applications. *Journal of Systems and Software*, 84(11):1871–1882.

[Von Alan et al., 2004] Von Alan, R. H., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

[Wang et al., 2014] Wang, H., Chan, W., and Tse, T. (2014). Improving the effectiveness of testing pervasive software via context diversity. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 9(2):9.

[Wang et al., 2012] Wang, J.-H., Ding, J.-J., Chen, Y., and Chen, H.-H. (2012). Real time accelerometer-based gait recognition using adaptive windowed wavelet transforms. In *Circuits and Systems (APCCAS), 2012 IEEE Asia Pacific Conference on*, pages 591–594. IEEE.

[Wang et al., 2016] Wang, W., Liu, A. X., and Shahzad, M. (2016). Gait recognition using wifi signals. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 363–373. ACM.

[Wang et al., 2013] Wang, X., Hovakimyan, N., and Sha, L. (2013). L1simplex: fault-tolerant control of cyber-physical systems. In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems*, pages 41–50. ACM.

[Wang et al., 2007] Wang, Z., Elbaum, S., and Rosenblum, D. S. (2007). Automated generation of context-aware tests. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, pages 406–415. IEEE.

[Want and Pering, 2005] Want, R. and Pering, T. (2005). System challenges for ubiquitous & pervasive computing. In *Proceedings of the 27th international conference on Software engineering*, pages 9–14. ACM.

[Wei and Chan, 2013] Wei, E. J. and Chan, A. T. (2013). Campus: A middleware for automated context-aware adaptation decision making at run time. *Pervasive and Mobile Computing*, 9(1):35–56.

[Wei et al., 2012] Wei, H., Huang, Y., Cao, J., Ma, X., and Lu, J. (2012). Formal specification and runtime detection of temporal properties for asynchronous context. In *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, pages 30–38. IEEE.

[Weis et al., 2007] Weis, T., Knoll, M., Ulbrich, A., Muhl, G., and Brändle, A. (2007). Rapid prototyping for pervasive applications. *Pervasive Computing, IEEE*, 6(2):76–84.

[Weiser, 1991] Weiser, M. (1991). The computer for the 21 st century. *Scientific american*, 265(3):94–105.

[Weiss and Hirsh, 1998] Weiss, G. M. and Hirsh, H. (1998). Learning to predict rare events in event sequences. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 359–363. AAAI Press.

[Wen et al., 2016] Wen, J., Indulska, J., and Zhong, M. (2016). Adaptive activity learning with dynamically available context. In *Pervasive Computing and Communications (PerCom), 2016 IEEE International Conference on*, pages 1–11. IEEE.

[Wen et al., 2015] Wen, J., Loke, S., Indulska, J., and Zhong, M. (2015). Sensor-based activity recognition with dynamically added context. In *Proceedings of the 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking*

and Services (Mobiquitous), pages 1–10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Wen and Wang, 2017] Wen, J. and Wang, Z. (2017). Learning general model for activity recognition with limited labelled data. *Expert Systems with Applications*, 74:19–28.

[Wen and Zhong, 2015] Wen, J. and Zhong, M. (2015). Activity discovering and modelling with labelled and unlabelled data in smart environments. *Expert Systems with Applications*, 42(14):5800–5810.

[Wirsing et al., 2008] Wirsing, M., Banâtre, J.-P., Hölzl, M., and Rauschmayer, A. (2008). *Software-Intensive Systems and New Computing Paradigms: Challenges and Visions*, volume 5380. Springer Science & Business Media.

[Wohlin, 2014] Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, page 38. ACM.

[Wohlin et al., 2012] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

[Wojciechowski and Xiong, 2008] Wojciechowski, M. and Xiong, J. (2008). On context modeling in ambient assisted living. In *Fifth International Workshop Modeling and Reasoning in Context (MRC 08), Delft, Niederlande*.

[Xu and Cheung, 2005] Xu, C. and Cheung, S. (2005). Inconsistency detection and resolution for context-aware middleware support. In *ESEC/FSE'05-Proceedings of the Joint 10th European Software Engineering Conference (ESEC) and 13th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-13)*, page 336.

[Xu et al., 2010] Xu, C., Cheung, S., Chan, W. K., and Ye, C. (2010). Partial constraint checking for context consistency in pervasive computing. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 19(3):9.

[Xu et al., 2012a] Xu, C., Cheung, S., Ma, X., Cao, C., and Lu, J. (2012a). Adam: Identifying defects in context-aware adaptation. *Journal of Systems and Software*, 85(12):2812–2828.

[Xu et al., 2006] Xu, C., Cheung, S.-C., and Chan, W. (2006). Incremental consistency checking for pervasive context. In *Proceedings of the 28th international conference on Software engineering*, pages 292–301. ACM.

[Xu et al., 2012b] Xu, C., Ma, X., Cao, C., and Lu, J. (2012b). Minimizing the side effect of context inconsistency resolution for ubiquitous computing. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 285–297. Springer.

[Xu et al., 2015] Xu, C., Xi, W., Cheung, S., Ma, X., Cao, C., and Lu, J. (2015). Cina: Suppressing the detection of unstable context inconsistency. *IEEE Transactions on Software Engineering*, PP(99):1–1.

[Xu et al., 2017] Xu, W., Shen, Y., Zhang, Y., Bergmann, N., and Hu, W. (2017). Gait-watch: A context-aware authentication system for smart watch based on gait recognition. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pages 59–70. ACM.

[Xue et al., 2013] Xue, W., Pung, H. K., and Sen, S. (2013). Managing context data for diverse operating spaces. *Pervasive and Mobile Computing*, 9(1):57–75.

[Yampolskiy, 2007a] Yampolskiy, R. V. (2007a). Indirect human computer interaction-based biometrics for intrusion detection systems. In *Security Technology, 2007 41st Annual IEEE International Carnahan Conference on*, pages 138–145. IEEE.

[Yampolskiy, 2007b] Yampolskiy, R. V. (2007b). Motor-skill based biometrics. In *6th Annual Security Conference, Las Vegas, NV*.

[Yampolskiy and Govindaraju, 2008] Yampolskiy, R. V. and Govindaraju, V. (2008). Behavioural biometrics: a survey and classification. *International Journal of Biometrics*, 1(1):81–113.

[Yang et al., 2015] Yang, Z., Wu, C., Zhou, Z., Zhang, X., Wang, X., and Liu, Y. (2015). Mobility increases localizability: A survey on wireless indoor localization using inertial sensors. *ACM Computing Surveys (Csur)*, 47(3):54.

[Yao et al., 2018] Yao, S., Zhao, Y., Zhang, A., Hu, S., Shao, H., Zhang, C., Su, L., and Abdelzaher, T. (2018). Deep learning for the internet of things. *Computer*, 51(5):32–41.

[Yau et al., 2006] Yau, S. S., Huang, D., Gong, H., and Yao, Y. (2006). Support for situation awareness in trustworthy ubiquitous computing application software. *Software: Practice and Experience*, 36(9):893–921.

[Ye et al., 2012] Ye, J., Dobson, S., and McKeever, S. (2012). Situation identification techniques in pervasive computing: A review. *Pervasive and mobile computing*, 8(1):36–66.

[Yin, 2009] Yin, R. K. (2009). Case study research: Design and methods (applied social research methods). *London and Singapore: Sage*.

[Yu et al., 2013] Yu, P., Ma, X., Cao, J., and Lu, J. (2013). Application mobility in pervasive computing: A survey. *Pervasive and Mobile Computing*, 9(1):2–17.

[Zabit et al., 2011] Zabit, U., Bernal, O., and Bosch, T. (2011). A self-mixing displacement sensor compensating parasitic vibration with a mems accelerometer. In *Sensors, 2011 IEEE*, pages 1386–1389. IEEE.

[Zachariadis et al., 2006] Zachariadis, S., Mascolo, C., and Emmerich, W. (2006). The satin component system-a metamodel for engineering adaptable mobile systems. *Software Engineering, IEEE Transactions on*, 32(11):910–927.

[Zhang and Babar, 2013] Zhang, H. and Babar, M. A. (2013). Systematic reviews in software engineering: An empirical investigation. *Information and Software Technology*, 55(7):1341–1354.

[Zisman et al., 2013] Zisman, A., Spanoudakis, G., Dooley, J., and Siveroni, I. (2013). Proactive and reactive runtime service discovery: a framework and its evaluation. *Software Engineering, IEEE Transactions on*, 39(7):954–974.