OULU 2013
C 458

ACTA UNIVERSITATIS OULUENSIS

Denzil Ferreira

AWARE: A MOBILE CONTEXT INSTRUMENTATION MIDDLEWARE TO COLLABORATIVELY UNDERSTAND HUMAN BEHAVIOR

UNIVERSITY OF OULU GRADUATE SCHOOL;
UNIVERSITY OF OULU,
FACULTY OF TECHNOLOGY,
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ACTA UNIVERSITATIS OULUENSIS
C Technica 458

DENZIL FERREIRA

AWARE: A MOBILE CONTEXT INSTRUMENTATION MIDDLEWARE TO COLLABORATIVELY UNDERSTAND HUMAN BEHAVIOR

Academic dissertation to be presented with the assent of the Doctoral Training Committee of Technology and Natural Sciences of the University of Oulu for public defence in OP-sali (Auditorium L10), Linnanmaa, on 6 September 2013, at 12 noon

UNIVERSITY OF OULU, OULU 2013

Acta Univ. Oul. C 458, 2013

Supervised by
Professor Vassilis Kostakos
Professor Anind K. Dey

Reviewed by
Professor Cecilia Mascolo
Doctor Florian Michahelles

Opponents
Professor Andrew T. Campbell

ISBN 978-952-62-0189-4 (Paperback)
ISBN 978-952-62-0190-0 (PDF)

ISSN 0355-3213 (Printed)
ISSN 1796-2226 (Online)

Cover Design
Raimo Ahonen

JUVENES PRINT
TAMPERE 2013

Ferreira, Denzil, AWARE: A mobile context instrumentation middleware to collaboratively understand human behavior.

University of Oulu Graduate School; University of Oulu, Faculty of Technology, Department of Computer Science and Engineering

Acta Univ. Oul. C 458, 2013

University of Oulu, P.O. Box 8000, FI-90014 University of Oulu, Finland

Abstract

This thesis presents a mobile instrumentation middleware, AWARE, aimed at facilitating our understanding of human behavior. We demonstrate how to use AWARE to build context-aware applications, collect data, and study human behavior.

Mobile phones are resource-constrained and several considerations need to be taken into account to create a research tool that ensures problem-free data collection. AWARE can mitigate researchers' effort when building mobile data-logging tools and context-aware applications. By encapsulating implementation details of sensor data retrieval and exposing the sensed data as higher-level abstractions, researchers spend less time developing software and save more time for doing research and analyzing the collected data, both quantitative and qualitative.

This thesis demonstrates AWARE's use in a number of case studies. These vary in the research methods we have used: prototype-building; large-scale deployment; surveys; interviews; cognitive walkthroughs; heuristic evaluation; laboratory & field studies data logs; Day Reconstruction Method (DRM); and Experience Sampling Method (ESM). Together with these methods, we demonstrate how AWARE helps study human behavior in different research scenarios, such as: enabling human-smartphone awareness, understanding concerns on battery life, modeling the proximity of users to their smartphones, and capturing location sharing concerns.

The thesis' contributions are: the design, implementation and evaluation of a novel mobile instrumentation middleware to facilitate an understanding of human behavior.

Keywords: computer, context, framework, human, instrumentation, interaction, middleware, mobile, sensing

Ferreira, Denzil, AWARE: Välikerrosohjelmisto mobiililaitteiden kontekstin instrumentointiin yhteistyövälineeksi ihmisen käyttäytymisen ymmärtämiseen.
Oulun yliopiston tutkijakoulu; Oulun yliopisto, Teknillinen tiedekunta, Tietotekniikan osasto
Acta Univ. Oul. C 458, 2013
Oulun yliopisto, PL 8000, 90014 Oulun yliopisto

Tiivistelmä

Tämä väitöskirja esittelee mobiililaitteiden välikerrosohjelmisto AWAREn, jonka tarkoituksena on helpottaa ihmisen käyttäytymisen ymmärtämistä. Esitämme, kuinka AWAREa voidaan käyttää apuvälineenä kontekstitietoisten sovellusten kehittämisessä, aineiston keräyksessä ja ihmisen käyttäytymisen tutkimisessa.

Mobiililaitteiden rajallisista resursseista johtuen ongelmattoman aineiston keräämisen mahdollistavan tutkimustyökalun kehityksessä tulee huomioida useita seikkoja. AWARE voi helpottaa mobiilien aineistonkeräystyökalujen ja kontekstitietoisten sovellusten kehittämistä tiivistämällä anturitiedon hankintaan liittyviä yksityiskohtia ja esittämällä anturitiedon korkeamman tason abstraktioina. Näin tutkijat voivat käyttää vähemmän aikaa ohjelmistojen kehitykseen ja enemmän aikaa itse tutkimukseen ja kerätyn aineiston sekä kvantitatiiviseen että kvalitatiiviseen analysointiin.

Tässä työssä havainnollistamme AWAREn käyttöä useissa tutkimuksissa, joissa on sovellettu erilaisia tutkimusmenetelmiä: prototyyppien kehittäminen, suuren kokoluokan tutkimukset, kyselytutkimukset, haastattelut, kognitiiviset läpikäynnit, heuristiset arvioinnit, aineiston kerääminen laboratorio- ja kenttätutkimuksista, päivärekonstruktiomenetelmä (Day Reconstruction Method, DRM) ja kokemusotosmenetelmä (Experience Sampling Method, ESM). Näiden menetelmien avulla osoitamme, kuinka AWARE helpottaa ihmisen käyttäytymisen tutkimusta erilaisissa tutkimusskenaarioissa, kuten ihmisen älypuhelintietoisuuden mahdollistaminen, akun kestoon vaikuttavien tekijöiden ymmärtäminen, älypuhelimen ja käyttäjän etäisyyden mallinnus sekä paikkatiedon jakamiseen kohdistuvien asenteiden selvittäminen.

Tämän väitöskirjan kontribuutiot ovat: uudenlaisen, ihmisen käyttäytymisen ymmärtämistä helpottavan mobiililaitteiden instrumentointiohjelmiston (AWARE) suunnittelu, toteutus ja arviointi.

Asiasanat: anturointi, ihminen, instrumentointi, konteksti, mobiili, ohjelmistokehys, tietokone, vuorovaikutus, välikerrosohjelmisto

To my BIG family and especially my sunshine, Eija

Acknowledgments

I would like to express my gratitude and appreciation to both my thesis advisors, Prof. Vassilis Kostakos (University of Oulu, Finland) and Prof. Anind K. Dey (Carnegie Mellon University, US), for the support and guidance throughout my research studies and pushing me "out of the bubble." Thank you.

My thanks also go to Prof. Mika Ylianttila, Dr. Tech. Susanna Pirttikangas and Prof. Timo Ojala (UniOGS follow-up group chair) for the feedback on my manuscript. I would like to thank the thesis pre-examiners, Prof. Cecilia Mascolo (University of Cambridge, UK) and Dr. Florian Michahelles (ETH Zürich, Switzerland) for the positive feedback and further suggestions. Moreover, I would like to extend my appreciation and gratitude to Prof. Andrew T. Campbell (University of Dartmouth, US) for accepting the opponent role at my doctoral thesis defense.

I also would like to thank all of my co-researchers at MediaTeam Oulu research group, for the fruitful research discussions and informal social gatherings. Especially to Jorge Gonçalves, Simo Hosio, Marko Jurmu, Dr. Hannu Kukka, Fabio Krüger, Danielle Zanni, Tommi Heikkinen and Dr. Yong Liu, who endure my daily dwelling at the office. Extending my gratitude to other research groups, I would like to especially thank Dr. Heli Koskimäki, Henna Tiensuu, Pekka Siirtola, Lauri Tuovinen, Tuomo Alasalmi from DataAI; Dr. Jaakko Suutala and Mikko Perttunen from ISG (Intelligent Systems Group); and Tomi Juntunen from OUSPG (Oulu University Secure Programming Group). You have taught me a lot on how to survive in Finland, "*kiitoksia ja hölökyn kölökyn!*" Crossing inter-continental boundaries, my gratitude to all my friends and co-researchers at UbicompLab (HCII, CMU) and M-ITI (Madeira Interactive Technologies Institute) research groups.

This dissertation was partially funded by the Nokia Foundation, Tauno Tönning-säätio, KAUTE-säätio and a UniOGS Traveling Grant. The financial support is greatly appreciated.

Lastly, but not the least, I would like to thank my family, especially my parents, Agostinho and Dorita Ferreira, and my sister, Loyola Ferreira. They witnessed first-hand to my "electrifying and freezing" experiments: "*para trás nem para tomar balanço*." My last thank you is for my wife, Eija Ferreira, who provided company during late working nights and kidnapped me on adventures to explore the world, countless times.

Oulu, 2013 – Denzil Ferreira

Abbreviations

ADB	*Android Debug Bridge*
ADT	*Android Developer Tools*
CSV	*Comma Separated Values*
DRM	*Day Reconstruction Method*
ECA	*Event-Condition-Action*
ESM	*Experience Sampling Method*
GMM	*Gaussian Mixture Model*
HTTP	*Hypertext Transfer Protocol*
IoT	*Internet of Things*
JSON	*JavaScript Object Notation*
MAC	*Media Access Control*
MMS	*Multimedia Messaging Service*
MQTT	*MQ Telemetry Transport*
MVC	*Model-View-Controller*
M2M	*Machine-to-machine*
OS	*Operating System*
QoC	*Quality of Context*
QoS	*Quality of Service*
RPC	*Remote Procedure Call*
RSSI	*Received Signal Strength Indication*
SHA	*Secure Hash Algorithm*
SMS	*Short Message Service*
UI	*User Interface*
UUID	*Universal Unique IDentifier*
XML	*eXtensible Markup Language*
XMPP	*eXtensible Messaging and Presence Protocol*

List of original publications

I Ferreira D, Kostakos V, Rogstadius J & Venkatanathan J (2010) Design, Prototyping and Evaluation of Ambient Media: Lessons Learned from the Ambient Notifier. Adj. Proc. of MobileHCI'10, Int WS on Tool-Support for Mobile and Pervasive Application Development (TSMPAD 2010), September 7–10, Lisbon, Portugal.

II Venkatanathan J, Ferreira D, Benisch M, Lin J, Karapanos E, Kostakos V, Sadeh N & Toch E (2011) Improving Users' Consistency When Recalling Location Sharing Preferences. Proc INTERACT '11: 380–387, Springer-Verlag.

III Dey AK, Wac K, Ferreira D, Tassini K, Hong J-H & Rojas J (2011) Getting Closer: An Empirical Investigation on The Proximity of User to Their Smart Phones. Proc UbiComp '11: 163–172, Beijing, China, ACM Press.

IV Ferreira D, Dey AK & Kostakos V (2011) Understanding Human-Smartphone Concerns: A Study of Battery Life. Proc Pervasive '11: 19–33, San Francisco, USA, Springer-Verlag.

V Ferreira D, Dey AK & Kostakos V (2011) Lessons Learned From Large-Scale User Studies: Using Android Market as A Source Of Data. Int J Mobile Human-Computer Interaction (IJMHCI) (4): 28–43, IGI Global.

Contents

1 Introduction

"Why be limited to thinking about computing as being delivered through single devices? Computing, and the instruments that deliver its services do not have to be intimately connected."

– Gregory D. Abowd (Abowd 2012)

1.1 Motivation

The notion of a computer has changed in the past 30 years. Initially, multiple users shared the computer (or mainframe), where each one of us was allocated a time-slot of use. Then came the personal computers, which allowed us to have our own computer at our disposal. With the evolution of technology, came mobile computers. First as laptops and netbooks, now they take the shape of tablets and smartphones. They are miniaturized computers that fit in a pocket, and have projected sales of 1.2 billion units in 2013 (Gartner Inc. 2012).

Mobile phones are inherently personal and the potential to sense the user's environment, or in other words, the user's context, is appealing to researchers. The convenience and availability of mobile phones and application stores makes it easier for a researcher to reach thousands of study participants. More importantly, mobile phones have several built-in sensors (e.g. accelerometer, proximity sensor, gyroscope). These mobile sensors are primarily used by the mobile operating system to enhance the user experience, such as application functionality or mobile phone user interaction (e.g. vibration feedback, screen orientation detection), but they are increasingly being leveraged for research purposes.

For example, mobile phones have been used to understand population movement flows in a city (O'Neill *et al.* 2006), and work by the Reality Mining team led the way on user-focused data collection via mobile phones (Eagle & Pentland 2005). O'Neill *et al.* (2006) provided researchers with the ability to abstract a city into a graph, where streets were nodes and intersections were the connection links. Furthermore, they provided the ability to interrogate the spatial structure of a city plan, and to investigate what factors lead to the presence or absence of people on the street. Soon after, they proposed a conceptual framework for designing and analyzing pervasive systems for urban environments

that takes into account architectural space, interaction spaces and information spheres (Kostakos *et al*. 2006).

While this was one of the first city-wide attempt to understand users' needs in an urban ubiquitous environment, using mobile devices and an array of distributed Bluetooth scanners, it still lacked the bottom-up approach where *individual* user data is collected and analyzed to contribute towards a synthesis of the big picture. The ability to *detect*, *infer* and *predict individual* and *collective* users' needs is *fundamental* for a *ubiquitous* computer.

Ubiquitous, from Latin "*ubique*", means *everywhere*. Research in ubiquitous and mobile computing is challenging because different applications and environments have different requirements, and furthermore different environments impose different and changing contexts (Weiser 1999). In Weiser's words, the future computer is described as a non-centralized, distributed computer amongst multiple devices, working together to sense the world around us. More specifically, it is a computer that disappears and makes intelligent inferences about what its sensors can capture and provides us with information when we need it.

Mobile phones are currently the most widespread sensing device. Can we instrument mobile devices to become the ubiquitous computer for the user? Widespread, mobile instrumentation offers opportunities for research and in facilitating a better understanding of human behavior. Challenges such as *heterogeneity*, *transparency*, *security, privacy*, *scalability*, *stability*, *reliability, redundancy*, to name only a few, require a collaborative effort to manage user's context. If collectively instrumented, mobile devices can become the ubiquitous computer.

An important first step is addressing the challenge of *reusability* of context. Researchers and application developers needs *tools* to *detect*, *manage* and *reuse* context, from various sources without having to build software and logging tools from scratch, over and over again. Thus, this thesis research question is:

> *"What are the challenges in creating a mobile instrumentation and context framework for studying human behavior, routines and context as a tool for researchers, application developers and users?"*

1.2 Research scope, methodology and thesis author's role

The scope of the research work presented in this thesis is on ubiquitous and mobile computing. We created a mobile instrumentation middleware, AWARE, for researchers of context-aware mobile computing, application developers and users, where raw data sensed from hardware, software and human sensors are converted to units of information (mobile context) that can be shared between other applications, sensors and humans alike. AWARE provides a foundation to create new mobile research tools for data mining and visualization. Specifically, AWARE takes into account the wide range of interrelated sources of context information and the relationships amongst them, including the user's individual and social behavior.

The methodology used in this work is an empirical, iterative design and engineering approach for developing, testing and evaluating AWARE, while at the same time exploring different research questions in this compilation of publications and manuscripts, as follows:

Article I presents the design, prototyping and evaluation of an ambient display, Ambient Notifier, which passively increased users' awareness of incoming calls, missed calls and received messages on their mobile phones. The discussion provides a reflection on the tools that were used to build the ambient display prototype, their shortcomings and suggestions on how these tools can be improved to assist the design and development of mobile and pervasive systems. Ambient display heuristics (Mankoff *et al.* 2003) were used to iterate the design of Ambient Notifier. Moreover, interviews with ten participants in a laboratory setup tested the conceptual match between the events on the mobile phone and the ambient display. We concluded with a field study, where participants used the Ambient Notifier on a daily basis for two weeks. Lastly, in a final interview, participants reported on their experiences with the prototype. The thesis author's contributions where the ideation, design, prototyping, implementation and evaluation of Ambient Notifier.

Article II introduces "trajectory reminder" as a contextual cue for the recollection of events that happened before and after the user was physically present at a specific location. More specifically, we wanted a better understanding of the user's preferences and needs in terms of location sharing for location-aware applications. In this article, we demonstrate that trajectory reminders improve users' consistency when recollecting the context for a specific event. Using the Day Reconstruction Method (DRM), we assessed the consistency between the

recollection from the user's past locations and mobile phone logged location data. The thesis author's contribution is in the implementation of the mobile software used in the study, and part of data collection and analysis.

Article III assesses an important assumption regarding the use of mobile phones: the user always carries his or her mobile phone. Previous work claims this is not true (Patel *et al.* 2006), where users kept their phones at arm's reach 58% of the time. In this article, we explore if these results still remain valid, given the increase of mobile smartphones adoption in recent years, especially since the introduction of Apple's iPhone, and the availability of Android devices. We instrumented 28 users (selected with surveys and interviews) with a wearable Bluetooth transciever and users phones to capture quantitative data on the distance between himself and his mobile phone, for the time period of four weeks. Furthermore, we used the DRM to capture discrepancies on the logged data on assessing the users proximity to their device. While it is still true that users keep their mobile phones at arm's reach 50% of the time, our results highlight the fact that proximity between the user and the smartphone is actually 90% of the time in the same room. Moreover, we also show that we can accurately predict the proximity of the device to the user at the arm and room level with over 90% accuracy. The mobile phone proximity to the user has significant implications on application development, particularly those that collect user and environment context and notifications. The thesis author's contributions were: implementation and deployment of AWARE to sense multiple contexts of proximity to the smartphone, calibration of distance metrics and data analysis on Bluetooth tag-smartphone proximity.

Article IV presents a large-scale, 4-week study of more than 4,000 people to assess their smartphone charging habits to identify timeslots suitable for opportunistic data uploading and power intensive operations on such devices, as well as opportunities to provide interventions to support better charging behavior. We describe how people charge their smartphones, the implications on battery life and energy usage, and finally, discuss how to improve users' experience with battery life. The thesis author's contributions are: the ideation, design, implementation, deployment, and data analysis of charging habits.

Article V continues the work of IV and provides an insight into how we exploited a mobile application store (Google Play) to deploy and recruit participants for a study. User studies with mobile devices have typically been cumbersome, since researchers have had to recruit participants, hand out or configure devices, and offer incentives and rewards. The increasing popularity of

application stores has allowed researchers to use such mechanisms to recruit participants and conduct large-scale studies in authentic settings with relatively little effort. Most researchers who use application stores do not consider the side effects or biases that such an approach may introduce. We synthesize our own experiences with prior reported findings to discuss the challenges, advantages, limitations and considerations of using application stores as a recruitment and distribution approach for conducting large-scale studies. The thesis author's contributions are: evaluation and data analysis of study participation in a large-scale application store deployment environment.

Articles I, II, III focused on building and designing AWARE, while IV and V on evaluating AWARE as a research tool to study human behavior, routines and context. Together, the articles contributed to iteratively elicit and address mobile middleware challenges. We discuss the articles contributions in more detail in Chapter 4.

1.3 Thesis outline

The rest of the thesis is organized as follows: in Chapter 2 we introduce the related work. We start by clarifying what is context and what we consider as mobile context. We then present middleware for mobile phones that have focused on studying and understanding human behavior, and explain how they inspired the design rationale of AWARE.

More specifically, we highlight CORTEX (Biegel & Cahill 2004; Sørensen *et al.* 2004), Context Studio (Korpipää *et al.* 2004), ContextPhone (Raento *et al.* 2005), AWARENESS (Sinderen *et al.* 2006), Momento (Carter *et al.* 2007), CenceMe (Miluzzo *et al.* 2008), EmotionSense (Rachuri *et al.* 2010), Empath (Dickerson *et al.* 2011), Funf (Aharony *et al.* 2011), Ginger.io (Ginger.io 2011) SystemSens (Falaki *et al.* 2011), Ohmage (Ramanathan *et al.* 2012) and lastly, ODK Sensors (Brunette *et al.* 2012).

In Chapter 3, we introduce AWARE, our mobile instrumentation middleware, and emphasize its contributions as a mobile instrumentation platform. In Chapter 4, we revisit the related work with a list of contributions and limitations of AWARE. We conclude with an assessment of AWARE as a research tool by considering five different use-cases and how these studies contributed to AWARE. Chapter 5 concludes the thesis.

2 Related work

In this chapter, we will revisit context definitions and appropriation to mobile context and the challenges in context, mobile context and mobile instrumentation middleware. We discuss context characteristics to describe the user's context. We follow up with a literature review on research in mobile context and provide an assessment of how previous mobile context middleware manage context.

2.1 Context

In this section, we begin by defining what context is, and more importantly, how it is used in context-aware applications. According to the Oxford's Dictionary of English, context is defined as "*the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood*." This definition pertains to a broad sense notion of context and it is, by itself, inadequate for context-aware computing.

The first definition of *context* refers to context as location, people and objects (identities), and changes to those identities (Schilit & Theimer 1994). Since then, researchers elicited additional dimensions to context: environment, time of day (Ryan *et al.* 1997), season, temperature (Brown & Bovey 1997), user's emotional state, focus, location and orientation (Dey 1998). As highlighted by Dey *et al.* (2001), these definitions rely on examples and are difficult to apply to context-aware applications. In this thesis, we borrow from a definition of context (Abowd *et al.* 1999) which overarches prior definitions and conveys to context-aware computing as follows:

> *"Context: any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects."*

Context is acquired from various sources (e.g. sensors) to categorize and describe the surroundings of entities (i.e. person, place or object). There are four essential categories, or characteristics, of context information: *identity*, *location*, *status* (or activity) and *time* (Dey *et al.* 2001). *Identity* is the ability to associate a unique identifier to an entity. *Location* is any geographical information associated with an entity, such as GPS coordinates, proximity, elevation, *etc*. *Status* (or activity)

harvests all possible sensed entity's information. Lastly, *time* is used for historical and logging context information.

In Weiser's vision (Weiser 1999) of the 21st century computer, computers can understand the users' context to provide relevant services or applications. In other words, ubiquitous systems require context-aware applications. The challenge is, however, how to create these context-aware applications.

Context Toolkit (Dey *et al.* 2001) is the reference conceptual framework for developing context-aware applications. Context Toolkit separates the acquisition and representation of context from the use of context by a context-aware application. The requirements for dealing with context are as follows:

- *Separation of concerns*: separate how context is acquired from how it is used.
- *Context interpretation*: the use of multiple layers of context should be transparent, provided by the architecture.
- *Transparent, distributed communications*: transparent communication between context sensors and applications.
- *Constant availability of context acquisition*: the components that acquire context must be executing independently from the applications that use them.
- *Context storage*: context history can be used to establish trends and predict future context values.
- *Resource discovery*: for an application to communicate with a sensor, it must know what kind of information the sensor can provide, where it is located and how to communicate with it.

Also fundamental is the *reusability* of context. For this purpose, Context Toolkit proposed the following conceptual components: *Context Widgets* are reusable building blocks for context information. They hide the complexity of the actual sensors used from the application and abstract the context information to suit the expected needs of the application. *Interpreters* take information from one or more context sources and produce a new piece of context information. *Aggregators* collect multiple pieces of context information that are logically related into a common repository of context. *Services* are components in the framework that execute actions on behalf of applications, synchronous or asynchronously. *Discoverers* are responsible for maintaining a registry of what capabilities exist in the framework.

In this thesis, we appropriate the conceptual framework in a mobile instrumentation middleware to capture and share context between applications, thus enabling context-aware mobile applications, presented in Chapter 3.

2.2 Mobile Context

In this section, we discuss context in the domain of mobile computing. Mobile phones present a unique opportunity for ubiquitous and mobile computing researchers as mobile phones are increasingly intertwined with our daily lives and offer more functionalities than communication (i.e. phone and messages). More importantly, these devices are gradually more equipped with various sensors, which can be used by researchers to understand the user's mobile context.

For example, researchers have used GPS and Internet connectivity to facilitate context-aware applications (Corey 2010; Oliver 2010), or accelerometers for motion tracking (Reddy *et al.* 2010), Bluetooth for distance measurements from the device (Patel *et al.* 2006) and anomaly detection (Buennemeyer *et al.* 2008; Schmidt *et al.* 2008).

The mobile nature of such mobile devices is the key to the idea of *mobile context-awareness*. The mobile phone is both a sensing platform and a computer and improvements in mobile computing power and sensing capabilities on these devices allow for a whole new era of mobile context research and development (Lovett & O'Neill 2010). Therefore, mobile context is context available on a mobile device and applications.

In this thesis, we adapted and extended the previous categories of context (Abowd *et al.* 1999) to our mobile instrumentation framework, as follows:

- *Who*: the unique identity of the entities (e.g. sensor or application)
- *What*: the characteristics of the entities that can be labeled, measured or inferred (e.g. a label for a geo-location coordinate, currently engaged physical activity for a person, *etc*.)
- *When*: the instance of time in which the event is occurring
- *Where*: the location (e.g. place, application, sensor) of the event
- *Why*: the intelligibility of the system or application, the user intent and accountability of the system, application and user.

The AWARE framework captures data to describe these five categories of mobile context: who, what, when, where and why. However, it is challenging to instrument all categories. For examples, a challenge for ubiquitous systems is to determine what is the users' intention for using a ubiquitous system (Church & Smyth 2008), without explicitly asking the user for input, which might not always be possible (e.g. a system without users' input).

The category *why* is, therefore, a very challenging category to capture as mobile context. From a human-perspective, *intelligibility* and *accountability* allow users to make informed decisions based on context. Intelligibility is the systems' ability to present to the user the current context, how it was acquired and where it is used (Lim & Dey 2011). Intelligibility can be considered, therefore, as an extension of Nielsen's visibility of the system status heuristic (Nielsen & Molich 1990), applied to context aware systems. The premise is that, if the system is clear about which context it is sensing, it should facilitate the users in deciding how to use the system and increase their trust using the system. Lim's work (2012) in intelligibility highlights the most important questions users need answered when using a context-aware application.

Accountability is the system's ability to mediate users' actions that might impact others. It is challenging to completely represent and sense human and social aspects of context (Bellotti & Edwards 2001), thus challenging the existence of fully autonomous ubiquitous systems. In AWARE, accountability is handled by explicitly asking users for their input on their mobile device. Although likely intrusive, AWARE can use several contextual cues (e.g. idle time, currently visible application, *etc.*) to determine if it is appropriate to interrupt the user while using his mobile phone.

2.3 Challenges in mobile context

In this section, we discuss the challenges in mobile context with a literature review. Mobile phones have come a long way since their conception. They are no longer used merely for communication needs (Böhmer *et al.* 2011). Mobile phones empower users with Internet access, music, video, navigation and entertainment. More importantly, these devices have more processing power and sensors built in. This is attractive to researchers as they can now use mobile phones as the most ubiquitous sensor that users carry at almost all times (Ferreira *et al.* 2012).

Traditionally, ubiquitous computing researchers have addressed specific challenges in the field. However, most of previous ubiquitous research has been restricted to sample size, areas as small as an office or laboratory, and very little of the instrumentation work is reusable: most studies build software and logging tools from scratch.

Extracting information from raw data is computationally intensive and not trivial. For example, parallel computing can be used to label sensor data by

offloading processing to a network of computers (Parthasarathy & Subramonian 2001); abstractions of raw data by categorization (i.e. labeled- and timestamp-locations as realms and user states) (Narayanan 2001); and machine learning case-based reasoning for context assessment (Mikalsen & Kofod-Petersen 2004).

We can *delegate* information extraction to a server, like in the CASS (Fahy & Clarke 2004) server-based context middleware. To exchange information, or raw data between different platforms and devices, *portability* is a requirement for a context-aware middleware. There are several strategies to support portability. Object serialization is used in Klava (Bettini *et al.* 2002) to transfer data over a network. For resource constrained devices such as mobile phones, a data stream clustering algorithm, such as RA-Cluster (Gaber & Yu 2006), can be used to reduce the amount of bandwidth required to transfer context-aware information.

Information might require *adaptation* before delivery to a context consumer (e.g. application, server, user). For example, QCompiler (Wichadakul *et al.* 2002) adapts video-on-demand streaming between different devices on the same and different networks with different bandwidth availability. Also, proxies can be used to adapt the information on (?) the server side before sending it to the client (Ardon *et al.* 2003; Zhang 2007).

Not all data and context can be treated the same, as it can be sensitive and private (i.e. CAMIS focused on delivering secure mobile location information (Olla & Patel 2003); Quercia & Hailes' (2005) risk-aware framework only allowed sharing context information if there was a trust relationship between applications); or time-critical information exchange as synchronous (i.e. WildCAT (David & Ledoux 2005) for shared application execution context and SeeMon (Kang *et al.* 2008) for mobile resources management) or asynchronous (e.g. component-based framework DREAM (Leclercq *et al.* 2005)) requirements.

Context is *volatile*. The context which applications and devices encounter can be dynamic, changing in runtime. Reflection can be used to allow applications to examine and modify their runtime behavior according to the current context (i.e. CARISMA (Capra *et al.* 2003), MobiPADS (Chan & Chuang 2003)), although at the expense of performance and reliability. A better approach is to allow the use of *ad-hoc* components (i.e. CORTEX (Sørensen *et al.* 2004)). By adding or removing components, a framework is able to extend its context sensing ability in runtime time.

A context-aware middleware needs to be intelligent. SOCAM (Gu *et al.* 2005; Berri *et al.* 2006) leveraged ontologies for semantic representation, context reasoning, classification and dependency. Although ontologies provide a

vocabulary for representing knowledge about a domain, it is restricted to describe specific situations in a domain.

Context information is *not only sensor-based, but also human-based*. Humans are, of course, the ultimate user of a context-aware system, and as such, they are an important concern in ubiquitous computing, with their intents, emotions and unpredictability. Context-mediated social awareness, collaboration, coordination, interruptions and mobility contribute to context in a working environment (Bardram & Hansen 2004).

The following sections highlight various attempts to develop mobile middleware that consider hardware-, software- and human-based sensors to instrument mobile phones for context-aware applications. We present them in chronological order. More importantly, besides the Context Toolkit guidelines, the middleware we review greatly informed design and architecture decisions for AWARE, as we discuss further.

2.3.1 CORTEX

CORTEX (Biegel & Cahill 2004; Sørensen *et al.* 2004) allows researchers to fuse data from mobile sensors, represent application context and reason about context. It incorporates STEAM (Meier & Cahill 2003), an event-based communication protocol for ad-hoc wireless environments to support loose coupling between sensors, actuators and application components.

CORTEX introduced the concept of a sentient object model for the development of context-aware applications. A sentient object is a software abstraction for sensors and actuators, with the following properties:

- *Sentience*: the ability to perceive the state of the environment via sensors;
- *Autonomy*: the ability to operate independently of human control in a decentralized manner;
- *Proactiveness*: the ability to act in anticipation of future goals or problems.

By combining sentient objects and an event-based communication protocol for ad-hoc wireless environments, CORTEX targeted mobile context-aware researchers to define inputs and outputs, contexts, fusion services and rules using CLIPS inference engine (CLIPS 1985). Inferences followed an event-condition-action (ECA) execution model (Ipiña & Katsiri 2001).

AWARE inspiration

Use an *event-based communication protocol* for exchanging events and data between sensors, actuators and applications; provide *abstractions* on sensors and actuators, as *ad-hoc components* for context; consider ECA for inferring context and creating new mobile contexts.

2.3.2 Context Studio

Context Studio (Korpipää *et al.* 2004) is a middleware that takes into account *users' mediation and accountability* in context inference, as it is challenging to fully automate actions based on context alone (Bellotti & Edwards 2001).

Context Studio considers *mediation* of context-dependent actions as manual, semi-automated, and fully automated. Manual actions are user-controlled actions based on the current context, which is detected by the user. Semi-automated actions are based on a context inferred by the mobile device and later adjusted and confirmed by the user. Fully automated actions are pre-programmed actions according to the context detected by the mobile device.

Context Studio uses a blackboard approach (i.e. a problem is separated into multiple sub-problems that combined solve the problem) to create contextual rules, actions and triggers (i.e. outcome). Users could combine the existing contextual probes to generate actions by the mobile device, therefore adding context-awareness to the mobile phone.

AWARE inspiration

Support *human-based input* for *controlling context*; use a *blackboard* approach to *combine multiple contexts* to generate new higher-level contexts for context probes, applications and users.

2.3.3 ContextPhone

ContextPhone (Raento *et al.* 2005) is a widget-based mobile middleware. A widget is a small, full-featured application. ContextPhone is built on top of four essential components: sensors; communications; widgets and system services.

Available sensors probed location, user interaction, communication behavior and physical environment. Fundamental to ContextPhone was the idea of context

as an understandable resource for the users, in other words, *intelligibility* of context.

By leveraging communications (e.g. calls, messages) and system services (e.g. Bluetooth), ContextPhone allowed researchers to share location and proximity contexts remotely via HTTP to a remote server. Using widgets, users had control over the sensors data collection.

AWARE inspiration

Consider *context as a resource*, not just for context-aware research tools, but also users; leverage *intelligibility* of context to users, as to understand and control context information exchange; support context information *exchange* with other applications, devices and users.

2.3.4 AWARENESS

AWARENESS (Sinderen *et al.* 2006) is a middleware that prioritizes users' privacy concerns. The middleware applies the concept of Quality of Context (QoC) (Buchholz *et al.* 2003) to express the quality characteristics of the context information.

Users' privacy concerns would increase or decrease QoC, depending on how much context is shared at any given time (e.g. disabling GPS would reduce the QoC for the context of location). Moreover, context would only be shared with previously trusted devices. More importantly, the mobile phone user is the sole controller of privacy aspects.

AWARENESS supported the development of mobile healthcare applications for patients and focused medical researchers.

AWARE inspiration

Support user's *privacy* concerns by enabling *control* over shared context.

2.3.5 Momento

Momento (Carter *et al.* 2007) was a middleware that provided integrated support for situated evaluation of ubiquitous computing applications. Momento had a mobile client, which displayed information requests to the user and was able to

log location, nearby people and audio. The researcher had a desktop client to configure and oversee a remote deployment. The server would send SMS/MMS or use HTTP messages between the mobile device and the server. Momento was integrated with the Context Toolkit (Dey *et al.* 2001) for fixed applications.

Momento used Short Message Service (SMS) and Multimedia Messaging Service (MMS) as communication channels for self reports, logged events and ESM from study participants' mobile phones. The SMS functionality allowed participants to answer researchers' questions, up to 160 characters, while MMS allowed text, pictures, audio, video anywhere the participants were located. These communication channels were motivated by the reduced availability of mobile Internet connectivity. Whenever possible and available, HTTP was used to send the answers back to the server.

With researchers in mind, Momento was motivated to consider leveraging existing devices as much as possible; provide support for multiple communication options; support qualitative, quantitative and context data in a unified client system; not requiring fully implemented applications; support monitoring and notifications; and support lengthy and remote studies.

AWARE inspiration

Support *human-based sensing,* as not all contexts are hardware- or software-based; support *remote control* over deployment; *network unreliability* is granted.

2.3.6 MyExperience

MyExperience (Froehlich *et al.* 2007) middleware captured both sensor- and human-based data to understand the user's motivation, perception and satisfaction on mobile technology.

Human-based data collection took form as surveys and the user experience sampling was triggered off sensor readings and pre-established researcher's rules. These triggers could be: launching an application, displaying a notification, play an audio file, vibrate or toggle the LED flash, take a screenshot of the screen, send an SMS or display an survey. MyExperience supported remote opportunistic synchronization of the collected mobile data and survey answers to a remote server, to ensure access to the data as soon as possible.

AWARE inspiration

Support *customizable human-based qualitative data* collection, such as surveys; support *opportunistic synchronization* of data to remote server.

2.3.7 CenceMe

CenceMe (Miluzzo *et al.* 2008) middleware inferred the presence of individuals by leveraging sensors on the mobile phone and sharing information through social network applications (e.g. Facebook and MySpace).

CenceMe introduces a split-level classification approach for sharing social context. Social context detected locally on the device is transferred to a backend server to match common shared social contexts to raise social awareness. With the split-level classification approach, some of the classification can be done on the phone with the support of the backend servers, or entirely on the phone.

In order to sense social events, the mobile phone's accelerometer, Bluetooth, audio and GPS were instrumented. The GPS and accelerometer detected possible activities by the user, namely, sitting, standing, walking and running. Bluetooth scans would detect the presence of other mobile phones. Audio analysis would detect people talking. The data captured on the mobile device was sent to the backend server via XML-RPC messages for social classification. On the server side, a classifier parsed social messages coupled with the mobile phone's data to infer three social states: talking, dancing, partying or alone.

CenceMe focused on users' social experience in: sharing presence information; stimulate curiosity amongst users while on the move; learn from their own activity patterns and social statuses.

AWARE inspiration

Support *social networks* context; support *process offload* for *asynchronous* data mining and context inference to higher processing capable devices.

2.3.8 EmotionSense

EmotionSense (Rachuri *et al.* 2010) focused on social psychology context. The middleware could sense individual emotions, activities, and verbal as well as proximity interactions amongst friends.

The middleware could detect speakers' identities, emotions and location by instrumenting the microphone, Bluetooth and GPS. An inference engine would adapt the sampling rate of the system according to the status of the user (e.g. in motion) and his surroundings (e.g. people around, talking) in order to minimize power consumption during data capture and processing, but still maintaining accuracy of inference.

Speaker and emotion recognition was performed on the mobile phone, using a previously trained Gaussian Mixture Model (GMM) classifier with the study participants, and implemented using an adapted Symbian version of Hidden Markov Model Toolkit (HTK 2003).

EmotionSense is meant for social scientists, allowing them to describe sensing tasks and rules to manage sensors according to the detected users' social context.

AWARE inspiration

Support *adaptive sampling* for power efficiency; support *social* context (e.g. online, offline, away, idle); and support *scalable* context for multiple research disciplines.

2.3.9 Empath

Empath (*Emotional Monitoring for PATHology*) (Dickerson *et al.* 2011) was a middleware to remotely monitor emotional health for depressive illness.

Empath is composed of a set of integrated wireless sensors, a touch screen station and mobile phones. Patients' diagnosis and therapeutic treatment planning were supported by reports generated by aggregating context such as sleep, weight, activities of daily living, and speech prosody.

The sensors were distributed in different locations at patients' homes and mobile phone, and data collected locally. Timely synchronization would upload the data to a remote web server to a MySQL database, pre-scheduled at daily, weekly or bi-weekly intervals. The behavior analysis routines run on the server and results would be displayed on the touch screen fixed station at patients' homes.

AWARE inspiration

Support *heterogeneous* remote devices; support *visualizations* of data.

2.3.10 Funf

Funf (*Friends and Family*) (Aharony *et al.* 2011) middleware focused on social and behavior sensing. Funf instruments the available hardware and software sensors on mobile phones, such as GPS, accelerometer, Bluetooth, communication activities (e.g. calls, messages), installed applications, running applications, multimedia information and others.

Sensor data is scheduled in slots of time, predefined by the researcher, with duration (minimum length of time the probe will run in seconds); period (length of time between probe executions); start (date after which the probe is allowed to run); end (date before which the probe is allowed to run).

The data is stored and encrypted locally on the mobile phone as several time-fragmented SQLite databases. The data is accessible via a decrypting and database merging desktop application *post-hoc*, which converts the data from SQLite file to MySQL database script or a Comma Separated Values (CSV) file. The data is visualized on a desktop browser using a Python script.

Funf's targets are researchers interested in collecting social and behavioral data and studies. "Self-tracking" users can use the Funf Journal application to collect their personal mobile data.

AWARE inspiration

Increase *security* of collected data by *encrypting* potential private data and identifiers; support visualizations of data *online*.

2.3.11 Ginger.io

Ginger.io (Ginger.io 2011) is a behavioral analytics middleware that turns mobile data into health insights.

Ginger.io provides a web-based dashboard for healthcare researchers and providers and a mobile application for patients. The mobile application passively collects movement, call and texting patterns. In a daily or weekly basis, the

mobile application requests feedback from the patients, as 3-5 steps questionnaires.

AWARE inspiration

Support *multi-step* questionnaires for human-based sensing.

2.3.12 SystemSens

SystemSens (Falaki *et al.* 2011) middleware captures usage context of mobile phones. Usage context is defined as the collection of users' interactions with research applications.

The users' interactions include battery, call, CPU usage, cell location, data connection active and traffic and telephony information events. The data would be captured and then sent remotely as JSON objects to an HTTP server. New application usage monitoring components could be added via AIDL interfaces to extend the middleware.

SystemSens is a researchers' middleware to instrument research applications and loggers.

AWARE inspiration

Built-in *debug messages* support for middleware actions, context inferences and data collection.

2.3.13 ohmage

ohmage (Ramanathan *et al.* 2012) is middleware that records, analyzes and visualizes data from both prompted experience samples by the user, as well as continuous mobile sensing for research in health.

ohmage is a mobile phone-to-web platform designed to create and manage experience sampling based data collection campaigns in support of mobile health pilot studies. The middleware supports time- and location-triggered self-reports; activity recognition based on sensor-fusion of accelerometer, GPS, Wi-Fi and cell tower radios; location tracking; exercise and sleep tracking; acoustic traces for social interaction detection; motivational messages for participant engagement. Data is uploaded to secure remote server, where they are accessed and visualized.

ohmage has been evaluated in seven independent studies, with different population: breast cancer survivors, new moms, HIV+, immigrant women, ADHD, PTSD and high school students.

AWARE inspiration

Support *online access* and *visualizations* of data.

2.3.14 ODK Sensors

ODK (Open Data Kit) Sensors (Brunette *et al*. 2012) is a middleware to simplify the interface between external sensors and mobile phones.

ODK Sensors abstracts application and driver development from user applications and device drivers, by management of discovery, communication channels and data buffers.

It is component-based, allowing developers to focus on writing minimal pieces of sensor-specific code, enabling an ecosystem of reusable sensor drivers. That way, it provides a high-degree of isolation between applications and sensor-specific code and applications should be able to continue to function even if sensors become inoperative. Integration of new sensors into applications is possible by downloading new sensor capabilities from an application market, without modifications to the operating system.

AWARE inspiration

Support *abstractions* of context for applications and sensors; support *ad-hoc* context sensors.

2.4 Summary

Building context-aware systems is a complex and time-consuming task due to the lack of adequate infrastructure support (Chen & Kotz 2000). The literature review further demonstrates how fragmented context-aware research is, as researchers focus on a specific domain of expertise. Application developers struggle to gather and reuse high-level context, and users experience poorly designed context-aware applications. The target *audiences* then for mobile context middleware are researchers, application developers and the users.

The literature reviewed suggests that contextual data can be hardware-, software- and human-based. Hardware-based sensors include both built-in mobile sensors as well as external sensors in the environment. Software-based sensors include network data sensors, algorithm-based sensors (e.g. filtered signal), or a derivative of sensor-fusion (e.g. the combination of two hardware sensors data to overcome a sensor's limitations or improve accuracy) of hardware sensors. Human-based sensors have traditionally been qualitative and capture data that is impossible or challenging to capture by hardware or software sensors. We consider then sources of context as hardware-, software- and human-based sensors.

For this thesis, we categorized mobile context-aware middleware according to how they *manage* context. By management, we mean the acquisition and creation of context. *Centralized* middleware manage context locally on the mobile phone, while *distributed* can also manage context distributed with other clients and servers or provided by external sensors.

Finally, we included desirable *properties* of context such as *shared*, *dynamic* and *scalable* context (Dey *et al.* 2001). First, *shared* context is desirable for *multidisciplinary research* and *collaboration,* and addresses challenges of *reusability, delegation,* and *accessibility* (e.g. *security, privacy, online visualization*) of context. Secondly, *dynamic* context is desirable for the challenges of context *volatility*, such as *runtime adaptation* and *manipulation* (i.e. *reflection, frequency*). Lastly, *scalable* context is desirable for the challenges of *heterogeneity, transparency, redundancy and portability* of context.

Summarizing the variations in the previous middleware design and usage, Table 1 presents existing middleware according the following properties:

- *Audience*: we highlighted *researchers*, *developers* and mobile phone *users* as target audiences of context-aware middleware;
- *Sensing*: the sources of contextual data, *hardware-*, *software-* and *human-based* context sensing;
- *Management*: how the middleware manage context, locally on the mobile phone (*centralized*) or distributed between itself and other devices (*distributed*);
- *Properties*: *shared* if context can be used locally on the mobile phone for another applications or devices; *dynamic* if context can be extended in runtime and adapts the current context; and *scalable* if the middleware supports adding new sources of context beyond core contextual sources.

With these requirements in mind, AWARE is a *collaborative*, *adaptive*, *event-driven*, mobile context instrumentation middleware for researchers, context-aware application developers and users. Researchers can use AWARE as a standalone middleware to collect data from study participants. Furthermore, researchers can create their own research-domain specific context sensors. Application developers can embed AWARE as a library API, to create context-aware applications for the benefit of the users. Finally, users can use AWARE as a standalone application to collect personal data and visualize contextualized information of their daily lives. The next chapter presents AWARE in more detail.

Table 1. Summary of existing context-aware middleware.

Middleware	Audience			Sensing			Management		Properties		
	R	D	U	HW	SW	H	Centralized	Distributed	Shared	Dynamic	Scalable
CORTEX	x			x	x		x		x		x
ContextStudio			x	x	x	x	x		x	x	
ContextPhone	x		x	x	x	x		x	x		
AWARENESS	x			x	x			x	x	x	
Momento	x			x	x	x		x		x	
MyExperience	x			x	x	x		x			
CenceMe			x	x	x			x		x	
EmotionSense	x			x	x		x			x	
Empath	x		x	x	x			x			x
Funf	x		x	x	x		x		x		x
Ginger.io	x		x	x	x	x		x	x		
SystemSens	x			x	x			x			x
ohmage	x		x	x	x	x	x		x		
ODK Sensors		x		x				x	x	x	x
AWARE	x	x	x	x	x	x	x	x	x	x	x

R – Researcher, D – Developer, U – User, HW – Hardware, SW – Software, H – Human

3 AWARE

In this chapter, we describe AWARE and our theoretical framework for mobile instrumentation. We discuss how the theoretical framework conceptually addresses different mobile instrumentation challenges, elicited from our literature review.

AWARE reduces the burden of building mobile context logging tools and increases the reusability of research results by encapsulating different research challenges into reusable research solutions for mobile context-aware applications.

Instrumentation is by definition a collection of instruments regarded collectively. *Middleware* is a software layer that sits above the operating system and below the application layer and abstracts the heterogeneity of the underlying environment (Mahmoud 2004). AWARE leverages hardware, software and human sensors on mobile phones to sense its user's environment, context and actions. More importantly, it then shares this mobile context, thus facilitating access and reusability to mobile context to other researchers, developers and to users with mobile context-aware applications.

AWARE is a mobile context instrumentation middleware that focuses on *sensing* context (hardware-, software-, human-based sensors), *storing* context (mobile, web and server data repositories), *sharing* context for research and the instrumentation framework itself (as shared knowledge) and *using* context in applications (for researchers, application developers and users) (Figure 1).

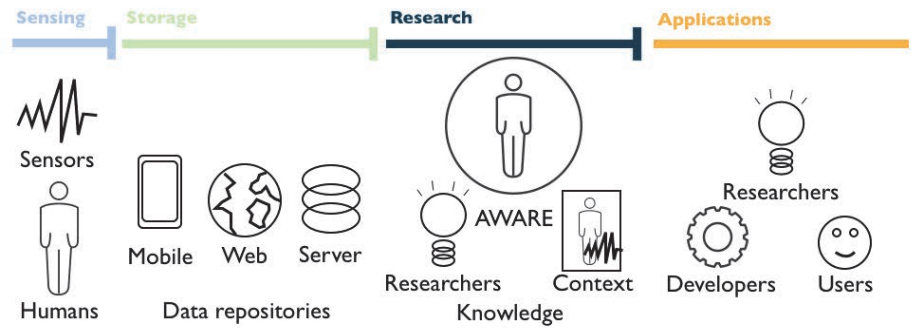

Fig. 1. AWARE's high-level overview.

Driven by our literature review, we defined a layered theoretical framework to structure AWARE in terms of different challenge areas shown as layers in

Figure 2. Each layer emphasizes different mobile research areas, as to support multidisciplinary collaboration. We will now describe each layer more in detail.

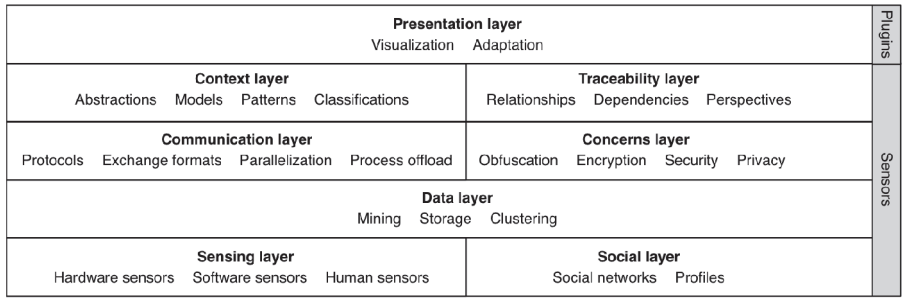

Fig. 2. AWARE's multiple layer theoretical architecture.

3.1 Sensing layer

The sensing layer is where context is *instrumented* using hardware, software and human sensors. In the sensing layer, AWARE collects data about the user and the surroundings. Besides the mobile sensors, AWARE considers *humans* as sensors to capture data that is challenging to sense using physical methods. Sensing layer challenges include sensor availability, proprietary access, device fragmentation and others when instrumenting hardware, software and human-sensors.

Hardware sensors are physical sensors that capture various kinds of environmental data, e.g., weather conditions, room temperature and others. *Software sensors* are non-physical sensors that gather data from non-tangible sources, such as Internet documents, files or emails. *Human sensors* collect human-subjective data, such as provided from mobile Experience Sampling Method (ESM) questionnaires, both qualitative and quantitative.

3.2 Social layer

The social layer is where context is *scrapped* from social networks and profiles. In the social layer, AWARE collects data about the user and the surroundings as social context.

Social layer challenges include social network integration, social availability, social use and social engagement. The social layer scrapes data from social networks, such as Facebook, Google+ and other available social networks. Social

data can be also sensed with hardware, software and other human sensors. Social profiles also provide information that would otherwise be missed.

3.3 Data layer

The data layer is where context is *stored*. AWARE is flexible on where to store context information, supporting both local and remote storage. AWARE collects context data locally on the mobile phone and stores the data into SQLite (SQLite 2009) database files on the mobile's storage, one per sensor. AWARE uses *Context Providers* to store sensor and add-on context and data, as an extension of Android's ContentProviders (Android API - ContentProvider).

Context Providers provide access to context. We use Android's Cursors to provide *passive* access to the data in a Context Provider. In other words, the data is *pulled* when necessary from the Context Provider. We use Context Observers to *actively* monitor and access the data as it is inserted to a Context Provider. In other words, the data is *pushed* to the Context Observers.

Each Context Provider provides a unique Content URI, that is, the location where context data is stored (*where*), stores an unique ID per context record (*who*), the data itself (*what*), the instance of time the data was collected (*when*) and an intrinsic trigger to save the data (*why*). For remote storage, AWARE provides integration with web services to upload the context data to a remote MySQL (MySQL 2008) database.

Data layer challenges include data mining, storage and clustering. In storage, we implicitly consider data management. Inference algorithms and applications will read the data to generate new context information, abstractions, models, detecting data patterns and classifications for the context layer.

3.4 Communication layer

The communication layer is where context is *shared*. AWARE provides communication mechanisms for local and external exchange of context data.

To support distributed deployments, AWARE leverages MQ Telemetry Transport (MQTT 2011) for exchanging context messages in a *publish/subscribe* approach between mobile phones and other servers and devices. MQTT is designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to *minimize network bandwidth* and *device resource requirements* whilst also attempting to ensure

reliability and some degree of *assurance of delivery*. These principles also turn out to make the protocol ideal of the emerging "machine-to-machine" (M2M) or "Internet of Things" (IoT) world of connected devices, and for mobile applications where bandwidth and battery power are scarce.

AWARE has a built-in MQTT client, which is used to exchange context with other devices, remotely manage sensors or issue ESM questionnaires on mobile devices. We have tested AWARE with IBM's Really Small Message Broker (RSMB) MQTT server, but other options do exist in the official MQTT website.

AWARE supports *message persistence*, both on the server and mobile phone. In other words, if the device or the server is unreachable, the messages are queued locally for delivery at a later time. By default, we enforce a Quality of Service (QoS) of *level 2* (*each message is delivered exactly once*), but AWARE also supports level 0 (no guarantee of message delivery) or level 1 (message is delivered at least once).

For *obfuscation*, we use a *one-way hash* to *encode* the device's AWARE Device ID (128-bit character string, provided by the traceability layer) and assign it as a MQTT client ID. By default, when activating the AWARE's MQTT client, the client automatically subscribes to three subscription topics (*broadcasts*, *esm***,** and *configuration*), under its own unique MQTT Device ID. They offer the following functionality:

- *broadcasts*: exchanges broadcasted context data and events
- *esm*: queues an ESM questionnaire
- *configuration*: remotely activate or deactivate context sensors.

Locally on the mobile device, communication or exchange of context data takes the form for *context broadcasts* and *context observers*.

- *Context Broadcasts*: AWARE uses Context Broadcasts to update other sensors and add-ons of the user's context, as an extension of Android Broadcasts (Android API - Broadcast). Context Broadcasts notify *changes of context*, but do not provide the data captured with the context sensors and plugins. Whenever a Context Sensor or Plugin detects or infers a change of the user's context, a broadcast is issued with a Context Broadcast. More importantly, Context Broadcasts can be received regardless of if the AWARE Add-on or another application is currently running or not, using Android's BroadcastReceivers (Android API - BroadcastReceiver). Moreover, a BroadcastReceiver can receive one or more Context Broadcasts at the same

time, thus enabling combinations of contexts to generate new higher-level context.

- *Context Observers*: AWARE uses Context Observers to observe changes in context and sensor data, as an extension of Android's ContentObservers (Android API - ContentObserver). Context Observers provide *active* and *event-driven* access to context. In other words, the Context Providers *push* the data to any *registered* Context Observer, thus becoming battery efficient. The Context Providers manage all the registered Context Observers and notifies them when new context data is available.

Communication challenges include transmitting data from multiple devices and sources leveraging multiple communication protocols. The communication layer handles different protocols and exchange formats, thus allowing cooperation between different sensors and applications.

Resource constrained devices such as mobile phones, might require offloading analysis and context generation to more able machines for faster processing. When completing such a request, the AWARE Server MQTT can notify the Context Sensor when the process is finished via a published MQTT message to the broadcasts topic.

3.5 Concerns layer

The concerns layer is where context is *protected*. In the concerns layer, AWARE obfuscates and encrypts the personal data using a Secure Hash Algorithm (SHA-1) one-way hashing of logged personal identifiers, such as phone numbers, thus providing privacy for the user's data. Some security is achieved by leveraging permissions, certificates and secure connections to access and transfer AWARE's logged data.

Security, privacy, obfuscation and encryption challenges need to be addressed when dealing with sensing context and potentially personal information. Other forms of security such as certificates, firewalls and secure connections should exist in this layer.

3.6 Context layer

The context layer is where data is *abstracted* as context and context is *produced*. AWARE is a plugin-oriented software architecture, which we refer to as AWARE

Add-ons. More crucially, AWARE Add-ons provide *reusable* context. There are two types of add-ons: *Context Sensors* and *Context Plugins*:

- *Context Sensor*: to *retrieve*, *abstract* data to *produce* context. Produced context is shared at the communication layer via Context Observers and Context Broadcasts. Context Sensors can *reuse* add-ons produced context data to produce new higher-level contexts. Data mining, machine learning techniques and conditional rules are built-in by researchers to *abstract data* as context, locally and remotely. Context Sensors *do not have an interface for the user* and are *unobtrusive*.
- *Context Plugin*: primarily to *use, reuse, visualize*, *explain* and/or *interact* with context. Context Plugins have built-in user interfaces (UIs) and allow users to *visualize*, *understand*, and *interact* with context. Moreover, Context Plugins provide *reusable* context intelligibility (Lim & Dey 2011) and context accountability (Bellotti & Edwards 2001) and can secondarily *produce* higher-level context.

Context layer challenges include creation of data abstractions, models, pattern recognition and machine learning algorithms classifications. As in Figure 2, Context Sensors inherit challenges from the sensing, social, data, communication, concerns, context, and traceability layers, while Context Plugins inherit the presentation layer challenges.

3.7 Traceability layer

The traceability layer is where context *dependencies*, *relationships* and *perspectives* are *implicitly* created and stored.

- *Dependencies*: implicit to Context Sensors and Context Plugins Android manifest declarations. A dependency is established by requesting access to another Context Sensor in the manifest (Android API - Manifest).
- *Relationships*: implicit to Context Sensors and Context Plugins implementation. When add-ons are implemented, context relationships are established when reusing other add-ons' context, either by registering Context Observers, listening for Context Broadcasts or with conditional rules when reusing contexts.
- *Perspectives*: implicit to Context Providers implementation. Modifying the visibility of Context Providers database table columns create different

perspectives of context data for different purposes (analogous to relational database views).

Traceability layer challenges include managing *context relationships*, managing *sensor relationships*, and providing different *context perspectives*, depending on who is exploring the data by leveraging data selection. More importantly, the traceability layer handles the *ownership* of the context data.

A Universal Unique Identifier (UUID) is created once during the client installation and assigned to that mobile device. The user can modify the assigned UUID by editing the AWARE Device ID in the AWARE Client. The UUID is reset every time the client is removed and reinstalled. However, updating the client does not affect the AWARE Device ID.

3.8 Presentation layer

The presentation layer is where context is *visualized*, *adapted* and where users *interact* with context. In Context Plugins, the presentation layer is responsible for providing context intelligibility and context accountability. Context Plugins and context-aware applications *visualize* context data in different mediums user interfaces, potentially requiring *adaptations* of the context information. The user interface can be as simple as displaying a number to as complex as an overlaid map.

Presentation layer challenges include user interface usability, understandability, visibility and adaptation to different media and display form factors to present context information.

3.9 Infrastructure

3.9.1 AWARE Client

For users and researchers, AWARE Client is the presentation view for *interacting* and *managing* Context Sensors and Plugins and is bundled as an Android application. For application developers, AWARE Client can be embedded as an Android library.

The AWARE Client is implemented as an Android Accessibility Service (Android API - AccessibilityService). The Android operating system (OS) automatically manages resources and intermittently terminates them. As an

Accessibility Service, AWARE becomes integrated with the OS, thus raising its priority (Figure 3).

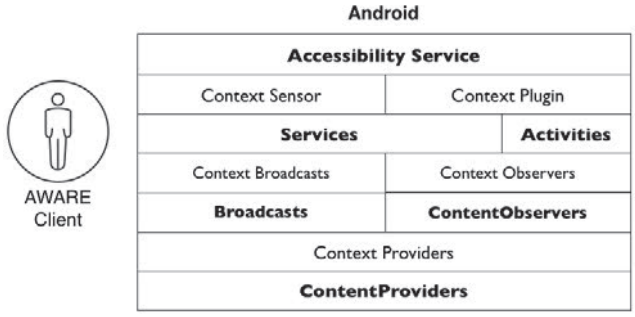

Fig. 3. AWARE Client software architecture.

From top-down, AWARE Client as an Accessibility Service increases the framework and add-ons' *reliability* and decreases the chances of missing data in the logs as it guarantees that AWARE is running for all Context Plugins and Context Sensors and other context-aware applications.

The Context Sensors are implemented as extensions of Android Services (Android API - Service), which are long-lasting background processes. Context Plugins are also implemented as extensions of Android Services, but include also Android Activities (Android API - Activity), for presenting context information to the user. The Context Broadcasts are implemented as Android Broadcasts (Android API - Broadcasts) and Context Observers are implemented as Android ContentObservers (Android API - ContentObservers) and respectively provide passive and active access to the Context Providers context data, implemented as an extension of Android ContentProviders (Android API - ContentProvider).

AWARE can be used as a *standalone* mobile logging tool, with *centralized context management*. The data is stored locally on the users' device and it is not shared with remote devices or servers, ideal for small scale and laboratory deployments. The AWARE Client has two interfaces: sensor dashboard and AWARE Add-ons (Figure 4).

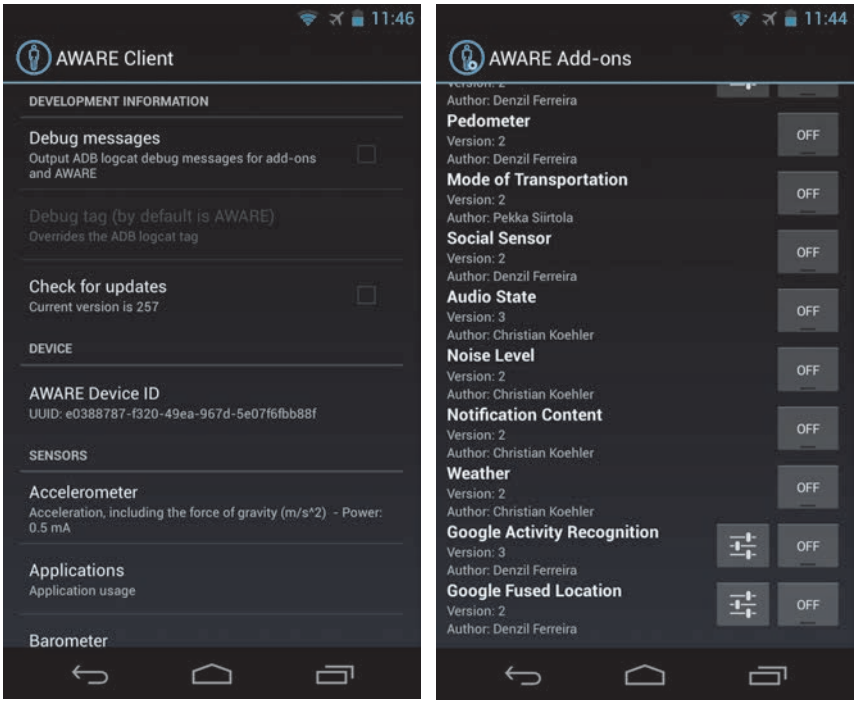

Fig. 4. AWARE Client user interfaces: the main client (left), AWARE Add-ons (right).

Using the sensor dashboard (Figure 4 – left), the user can manage the collected data on the mobile phone. The AWARE Add-ons interface (Figure 4 - right) allows the user to manage installed add-ons and configure Context Plugins accordingly. AWARE provides *control* of the context data *by the user*, and *not* by an external entity (i.e. developer or researcher).

The AWARE Client controls the instrumented hardware-, software- and human-based sensors. Besides the type of sensor, we further describe the data each sensor manages in Table 2.

Table 2. AWARE core sensors.

Sensor	Type			Description
	HW	SW	H	
Accelerometer	x			sensor profile and acceleration force along the device's axis, *including* gravity, in m/s^2
Applications		x		foreground and background applications used on the device
Barometer	x			sensor profile and atmospheric air pressure, in mbar/hPa
Battery	x	x		battery and power events (e.g. reboot, shutdown) data
Bluetooth	x			built-in Bluetooth sensor information and performs interval scans for visible neighbor Bluetooth devices
Communication		x		users' communication usage (i.e. phone call states, messages states)
ESM			x	user-provided data from ESM questionnaires. The ESM can be triggered by other context events, time or remotely using the AWARE Server dashboard
Gravity	x			sensor profile and the force of gravity along the device's axis, in m/s^2
Gyroscope	x			sensor profile and the rate of rotation around the device's axis, in rad/s
Installations		x		applications added, removed or updated on the mobile device
Light	x			sensor profile and ambient luminance, in lux
Linear accelerometer				sensor profile and acceleration force along the device's axis, *excluding* gravity, in m/s^2
Locations	x	x		network and/or GPS locations from the mobile device. The best estimated location is provided
Magnetometer	x			sensor profile and the geomagnetic field strength along the device's axis, in micro-tesla (μT)
Network		x		network usage (i.e. airplane mode, Wi-Fi, mobile network, Bluetooth, GPS, Internet availability)
Orientation	x			sensor profile and the orientation angle along the device's axis, in degrees
Processor		x		system, user and idle mobile processor workload
Proximity	x			sensor profile and the distance between the device and an object, in near/far (binary) or centimeters (cm)
Rotation	x			sensor profile and the rotation vector along device's axis
Screen	x	x		screen status (e.g. on or off) and user's locking and unlocking events
Telephony	x			mobile phone's telephony capabilities (e.g. network speed, type) and GSM/CDMA towers and neighbor towers, if available
Temperature	x			sensor profile and the ambient air temperature, in Celsius
Traffic		x		network traffic (i.e. packets and bytes) on Wi-Fi, Bluetooth, and mobile network
Wi-Fi	x			built-in Wi-Fi sensor information and performs interval scans for visible neighbor Wi-Fi devices

HW – Hardware, SW – Software, H – Human

When installing a new AWARE Add-on, either a Context Sensor or Plugin, the AWARE Client adds it to the available add-ons list and the user can activate and configure any settings available, as in Figure 5.

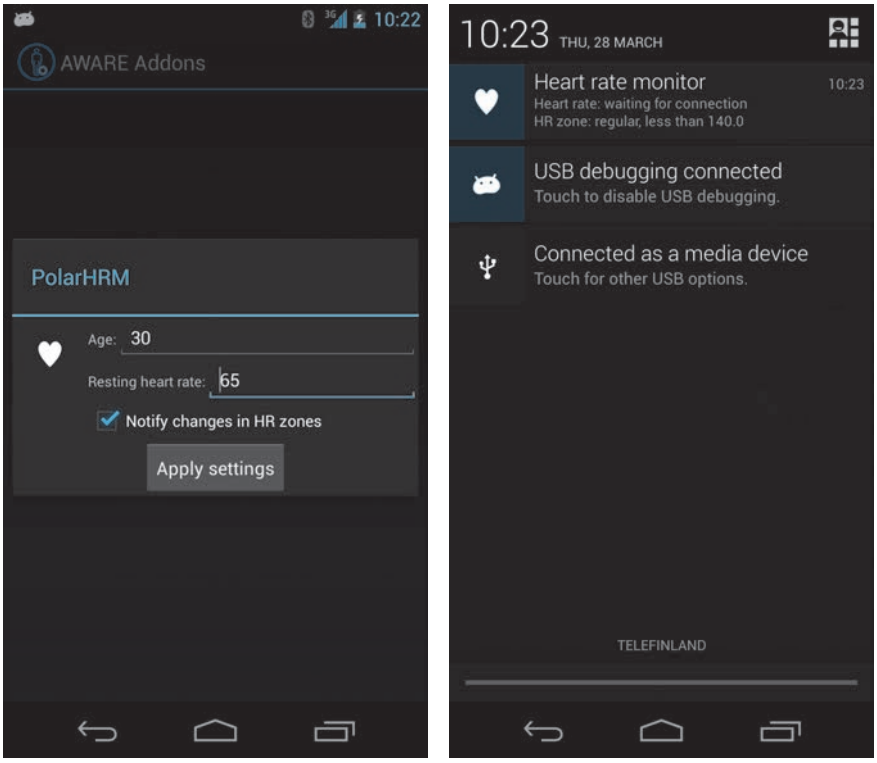

Fig. 5. AWARE Add-on PolarHRM: options (left), user interface (right).

For researchers and developers, AWARE APIs simplify context data acquisition from AWARE, as well as the add-ons, with simple requests for enabling or disabling a context sensor and registering Context Observers and listening to Context Broadcasts.

Different strategies are in place to attempt to minimize data collection loss, such as the use of processor wake locks (to keep the sensors alive even when the phone is idle), multi-threading (to reduce delays in context storage and allow cooperation between multiple sensors) and exception fallbacks (to decide what to do if one sensor is not available or is for some reason faulty).

For sensors built into the mobile device, AWARE captures sensors' maximum range (maximum measurable value), minimum delay (the minimum sensing frequency), name of the sensor, power requirements of the sensor (mA), resolution in the sensor's units, type of sensor (hardware, software), vendor and version of the sensor. This provides to researchers and application developers a profile on the sensors' capabilities, power requirements and limitations.

3.9.2 AWARE Server

For researchers and application developers, AWARE Server is a distributed infrastructure to share and reuse context data with other external sensors and applications, installable in a web server.

The AWARE Server allows remote context data storage and exchange with other devices, with two different approaches: web services and MQTT (Figure 6).

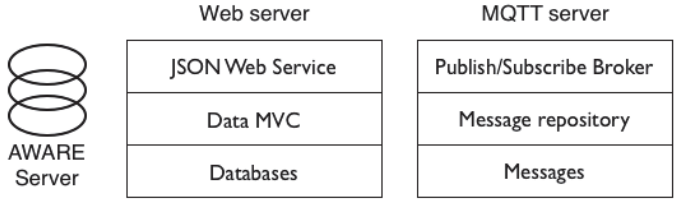

Fig. 6. AWARE Server: web server and MQTT server.

For researchers who want to be able to send the data remotely to an AWARE Server, in the AWARE Client, they can configure the data exchange settings for the Web server and the MQTT server.

The MQTT server provides support for distributed context infrastructure and provides real-time context message exchange to all AWARE Clients. The MQTT server is an instance of IBM's RSMB publish/subscribe broker. The servers can be clustered, to support load-balance of context information. For added security and privacy, RSMB also supports secure and authenticated connections. The message repository can be file-based or database-based and can store context messages, events and data.

The Web server synchronizes the mobile context data with an AWARE Server web service. The synchronization can be scheduled, triggered locally with an event or requested on-demand remotely. The Web server services were built using

CodeIgniter PHP framework (CodeIgniter 2010) and receive HTTP requests as POST from the AWARE Client. The requests contain JavaScript Object Notation (JSON) representations of the context data stored in the client.

The web services follow a Model-View-Controller (MVC) design pattern. Depending on the request, the controller loads the corresponding object model and executes the command. If the request is for visualization, the controller additionally loads the corresponding presentation layer view. The commands supported by the web services are:

a) *latest*: get the latest entry from the web server repository
b) *insert*: add new entries to the web server repository
c) *create_table*: replicates a new database table on the web server repository
d) *clear_table*: remove a specific user's data from the web server repository
e) *alive*: used as a keep-alive mechanism for AWARE Clients. Useful for remote study deployments and monitors participants' device availability
f) *visualize*: request a presentation layer view for a sensor or add-on
g) *mqtt*: loads the AWARE Server dashboard for publishing MQTT messages to AWARE Clients, useful for remote study deployments
h) *send_esm*: request an ESM questionnaire to be displayed remotely, for human-based context sensing and qualitative data acquisition.

Items c) and d) allow extensibility on the server side. As new add-ons are available, we needed to provide a mechanism to dynamically expand the web server database. The c) command receives the table schema and replicates it on the server side. The d) allows users to remove their logged data from the web server on-demand. To support remote deployments research, a keep-alive mechanism performed with the e) command, allows researchers to monitor the participant device's availability and participation.

To visualize the data collected, f) allows web visualizations snippets using D3 JavaScript library (D3js 2012) (Figure 7).

Fig. 7. The information displayed on the users' phone (left) can be seen in AWARE Server visualizations: pedometer and mode of transportation (right).

The AWARE Server Dashboard provides researchers with a tool to issue broadcasts, queue ESMs and change client configurations (Figure 8).

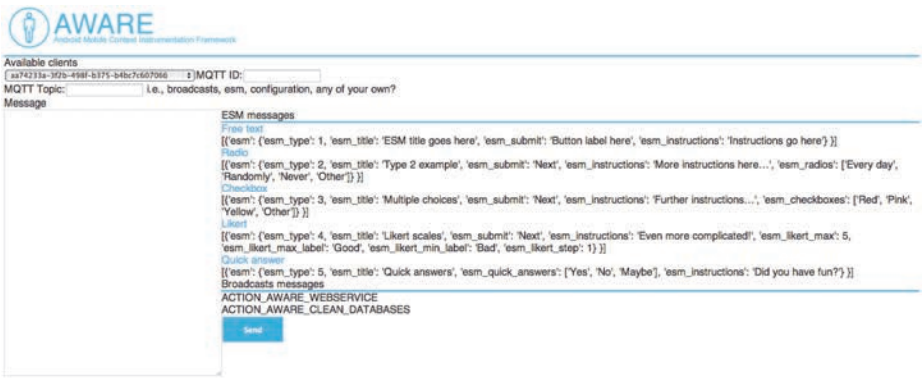

Fig. 8. AWARE Server Dashboard for researchers.

The ESM requests, provided with h), adds support for *user-provided context* by leveraging mobile Experience Sampling Method (ESM). AWARE provides a flexible ESM questionnaire-building schema, defined as an ESM JSON object. Diverse ESM questions can be chained together to support a step-by-step ESM, as a JSON array of ESM JSON objects. The questionnaires can be triggered by context, time or on-demand, locally (via a broadcast request) or remotely (with AWARE Server Dashboard). Although user-subjective, this functionality allows crowdsourcing information that is challenging to instrument with physical sensors (Figure 9).

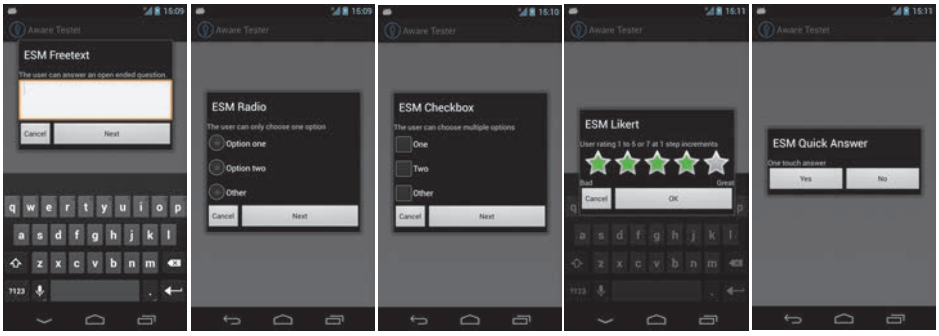

Fig. 9. AWARE ESM questionnaires interfaces, issued at the mobile device, from left to right: free text, radio, checkbox, Likert scale and quick answers.

While AWARE offers the flexibility to create and issue a mobile ESM, we have followed certain guidelines: based on our experiments and study participants feedback, answering an ESM should *never* be mandatory, or have a default answer. If the user presses the Home or the Back button on the mobile device, the ESM is dismissed. AWARE tracks the ESM current statuses: *new*, *dismissed*, *expired* and *answered*. Furthermore, if the user receives an ESM and the device's screen is turned off, the device vibrates as to get the users' attention, and lastly, an ESM can be set to expire, to support *in-situ*, temporal-dependent human-based context.

3.9.3 AWARE end-to-end overview

Here, we provide a high-level overview of AWARE, from end-to-end. We will start with sensor instrumentation and end with context visualizations on the browser. We refer to Figure 10, from top-down.

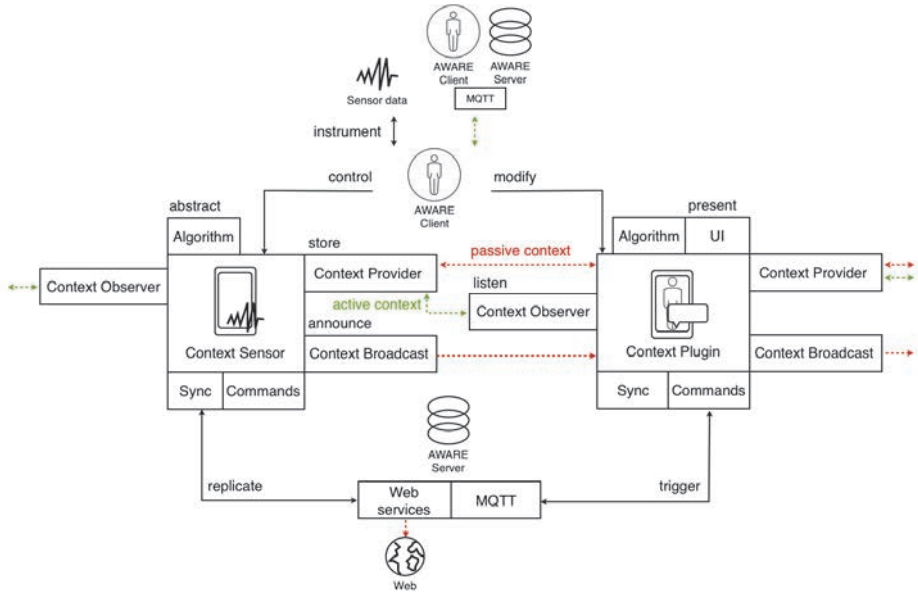

Fig. 10. AWARE's infrastructure overview.

Sensor data is *instrumented* through AWARE Client. The sensor data can be collected from one or more sensors (e.g. sensor fusion). The sensor implementation details are encapsulated in Context Sensors, components of AWARE Client. The Context Sensors can be hardware-, software-, or human-based sensors from the mobile device or from *external* sensors. The AWARE Client allows *users* to *control* Context Sensors and Plugins (AWARE Add-ons), which *abstract* the sensor data into context. AWARE Client allows *researchers* to *modify* the Context Sensors and Plugins to different requirements (i.e. increased frequently, sensor sources and other sensor specific settings). Users can also modify Context Plugins settings or visualize context data in available user interfaces (UIs).

Context Sensors and Plugins *abstract* the sensor data into context with *algorithms* (i.e. sensor implementation details, machine learning algorithms, and others). Context data is *stored* locally with Context Providers and *announced* with Context Broadcasts. The Context Sensors and Plugins also *listen* to other Context Sensors and Plugins context data with Context Observers and Android's BroadcastReceivers.

AWARE provides *active* and *passive* access to the sensor data and context data. For *active* sharing (i.e. *push*), the add-ons register a Context Observer to listen to changes (i.e. insert, delete, update) in a specific Context Provider, including the data. If required, the data can be *modified* or *labeled* in runtime.

For *passive* sharing (i.e. *pull*), the add-ons have two options: listen to broadcasts from Context Broadcasts and query the Context Provider database using Android's Cursors when there is new context data available; or directly query the Context Provider database on demand.

The AWARE Server serves three purposes: replicate, command, exchange and visualize context data: for *replication*, encapsulated in the Context Sensors and Plugins are synchronization mechanisms for remote storage of the collected sensor and context data via web services. The replicated data can then be *visualized* on a web browser. Using MQTT, the AWARE Server Dashboard triggers *commands* that can be delivered to other AWARE Clients and Servers. On the other hand, AWARE Clients triggers commands on other devices and *exchange* context data actively.

3.10 Summary

Despite years of research on context-aware computing, there are still very few "smart" mobile applications and systems that make use of context data in real-time. In the literature review, we emphasized that building a tool that fits all is a herculean task if tackled alone.

Making the transition from mobile phones to "smartphones", in the true sense of the word, requires more tools that offer programming and development support. Android has become the most popular platform for developing mobile applications due to the great suite of development tools it provides, namely, the Android Developer Tools (ADT) and the Android Debug Bridge (ADB). However, the development of "context-aware" applications remains challenging because developers have to deal with obtaining raw sensor data, analyzing the data to produce context, and often writing code from scratch. There is a lack of a coherent and modular repository of relevant tools.

AWARE is a middleware for Android that supports three types of users: developers, researchers and end-users. AWARE allows *developers* to create context-aware applications that listen to Context Broadcasts and embed AWARE as an Android library. For example, a sports tracking application can use the context provided by the Context Plugin Pedometer to know how many steps the

user has taken and when did this happen. Developers have now access to context that could be challenging to create from scratch, often in areas of expertise that might require years of experience or calibration and testing.

For *researchers*, AWARE offers an opportunity to contribute, collaborate and reuse other researchers' expertise when preparing a mobile data probe. The multiple layers of the theoretical architecture focus on different areas of research, thus enabling different researchers' participation. Effectively acting as an instrumentation toolkit, AWARE is packaged to facilitate user studies where data is stored either locally on the phone or remotely. The AWARE Client is deployed into participants' phones, while the AWARE Server allows researchers to monitor and manage the study participants.

The built-in ESM mechanism allows researchers to collect in-situ, real-time qualitative context from the users, triggered either locally, or remotely using the AWARE Server Dashboard. But more importantly for the *users*, the AWARE Client offers control and ownership over what is collected on their device, at the benefit of the added context-awareness capabilities and applications AWARE provides.

Next, in Chapter 4, we discuss how the literature and our research have driven the design, evolution and evaluation of AWARE. We demonstrate how AWARE mitigated the development effort when building mobile data-logging tools and context-aware applications for different research questions. By encapsulating implementation details on sensor retrieval and exposing the sensed data as higher-level abstractions, we spent less time coding and more time doing research and analyzing the collected data, both quantitative and qualitative to better understand human behavior.

4 Discussion

Dey *et al*. (2001) classified computing context based on four characteristics: identity, location, status or activity, and time. In Section 2.2, we adapted these previous categories as *who*, *what*, *when*, and *where* for mobile context and we added one more category: *why*. The *why* can be the most multifaceted of the context categories.

Context intelligibility (Lim & Dey 2011; Lim 2012) provides to the user an answer on explaining context (e.g. why did this happen?). Context accountability (Bellotti & Edwards 2001) provides to the user an answer managing context (e.g. who is responsible and what can I do?). Context intelligibility and accountability provide an answer to *why* from a systems' perspective.

From a users' perspective, mobile ESMs have been traditionally used as qualitative sensors (Consolvo & Walker 2003). AWARE considers mobile ESM also as quantitative sensor. By using these five categories of context, AWARE is a tool for mobile researchers that helps to collect contextual data from a large number of users to support studies on sensing of human behavior, as to better understand it.

4.1 State of the art

There are other mobile research tools available. To the best of the author's knowledge, Table 1 highlights the most relevant tools and summarizes their capabilities in terms of context management. We discuss further how they inspired AWARE's theoretical architecture and software architecture.

- CORTEX's (see section 2.3.1) sentient object model inspired the use of an event-based communication protocol (locally with Context Broadcasts and Context Observers, remotely with MQTT publish/subscribe messages) to exchange context events and data locally between AWARE Add-ons and applications, and remotely between AWARE Clients and Servers. More importantly, the notion of a sentient object model as encapsulated sensing, autonomous and pro-activeness behavior motivated the creation of Context Sensors and Plugins as sentient *ad-hoc* components. While CORTEX took an event-condition-action (ECA) approach to infer context (integrating with CLIPS), AWARE introduces a *context-action* (CA) approach to *abstract context inference* itself. CA reduces the need to understand the underlying

events and conditions that generate a specific context and encapsulate expert knowledge into *reusable*, *ad-hoc* context components, available as AWARE Add-ons;

Contributions: data, communication, context and traceability layer.

- Context Studio's (see section 2.3.2) use of mediation and accountability motivated distinguishing between Context Sensors, as invisible context generation; and Context Plugins as visible context generation and as user settings. Context Studio blackboard approach to combine multiple contexts to generate higher-level contexts reinforced the AWARE Add-ons.

  Contributions*:* context, traceability and presentation layer.

- ContextPhone (see section 2.3.3) inspired AWARE to consider context, not just a resource for the users (Context Plugins and applications), but also a resource for other researchers and developers, with Context Sensors. More importantly, motivated sharing the local context resource as Context Providers, and remotely with AWARE Server services.

  Contributions: data layer.

- AWARENESS (see section 2.3.4) inspired AWARE Client as user-managed context controller for all available context sensors, to not-collected private information and hash personal identifiers to obfuscate the source of context.

  Contributions: concerns layer.

- Momento (see section 2.3.5) inspired the creation of AWARE Server Dashboard for researchers to oversee study participants and to visualize the collected data and motivated regarding mobile ESMs as a context human-based sensor.

  Contributions: sensing and presentation layer.

- MyExperience's (see section 2.3.6) and Ginger.io (see section 2.3.11) inspired AWARE's support for step-by-step questionnaires, customizable for different research and input needs. AWARE supports remote synchronization of context data, based on time, context events and remotely triggered. AWARE uses JSON objects to transfer data remotely as this provides flexibility to add new types of sensors and data record types without changing

database schemas. In addition, handling JSON objects on Android is supported natively (Sharkey 2009).

Contributions: sensing, data, communication and presentation layer.

- CenceMe (see section 2.3.7) inspired AWARE to consider social networks as sources of social context (e.g. online availability) and add support for asynchronous server-side data mining.

  Contributions: social and communication layer.

- EmotionSense (see section 2.3.8) inspired AWARE to support adaptive sampling of context data, where researchers and developers can modify the sampling rate based on contextual events (e.g. time of day, network availability, battery level).

  Contributions: sensing layer.

- Empath (see section 2.3.9) and ODK Sensors (see section 2.3.14) inspired AWARE to provide an interface for integration with remote, heterogeneous devices. With Context Sensors, researchers and developers can extend support for other external sensors. Visualizations of collected data can be generated with the AWARE Server Dashboard.

  Contributions: communication and presentation layers.

- Funf (see section 2.3.10) inspired AWARE to obfuscate and encrypts logged personal identifiers, such as phone numbers, thus providing privacy for the user's data.

  Contributions: concerns layer.

- SystemSense (see section 2.3.12) inspired AWARE to provide debugging capabilities for AWARE's actions, context inferences and sensor readings.

  Contributions: sensing, data and traceability layers.

- ohmage (see section 2.3.13) inspired adding to AWARE Server Dashboard support for modifying AWARE's configuration on study participants' devices remotely.

  Contributions: sensing and communication layers.

4.2 Mobile Context Toolkit

As a research tool, AWARE streamlines the effort of developing mobile logging tools by encapsulating solutions for different research challenges. More importantly, AWARE provides a middleware to *reuse* and *collaborate* with other researchers to solve pending research challenges.

Besides being a research tool, AWARE provides an API for application developers for building context-aware applications. The AWARE API follows the reference conceptual Context Toolkit (Dey *et al.* 2001) context-aware application development requirements:

- *Separation of concerns*: each sensor and AWARE Add-on collects data, independently of where it is used or how it is used.
- *Context Interpretation*: abstractions of multiple layers of context are transparent to the researcher and developer, using the Context Providers and AWARE Add-ons.
- *Transparent, distributed communications*: the context sharing mechanisms (e.g. Context Provider, Observer, Broadcast) offer transparent communication between context sensors and applications.
- *Constant availability of context acquisition*: the context sensors operate independently from the applications that use them.
- *Context storage*: context is stored locally, and optionally, remotely to the AWARE Client and can be used to establish trends and predictions of context.
- *Resource discovery*: for an application to communicate with a context sensor, it needs to know where the context is stored, provided by the Context Provider content URI.

We can make an analogy between the conceptual components in the *Context Toolkit* and AWARE:

- *Context Widgets*: as Context Plugins
- *Interpreters*: as Context Sensors
- *Aggregators*: as Context Providers
- *Services*: as the collection of AWARE's core sensors and add-ons
- *Discoverers*: as Context Broadcasts and Context Observers.

As a library, developers can request context data from AWARE Add-ons or Context Sensors, and with the users' consent, have access to high-level context

inferences, shared as Context Broadcasts. Alternatively, developers can register Context Observers with Context Providers.

4.3 Challenges, contributions and limitations

The review of mobile research literature and tools on managing users' context elicited an exhaustive list of challenges for mobile context-aware middleware. In this section, we discuss mobile instrumentation middleware challenges, AWARE's contributions and limitations.

4.3.1 Multidisciplinary research, reusability

Research fragmentation is the biggest challenge for mobile instrumentation middleware. The theoretical framework highlights different domains in mobile computing which researchers can contribute to, as support for multidisciplinary research. However, other research domains exist and would extend AWARE further to other research domains.

The AWARE Add-ons allow researchers work independently or to collaborate with others, by isolating add-on's core functionality from its research and application development use. AWARE context sharing mechanisms support context *exchange* and *reusability*, and application developers can create better context-aware applications for the users. AWARE also supports *reusability* of sensors, which we discuss next.

4.3.2 Heterogeneity, scalability, transparency and portability

There is a diversity of sensors available, both built-in and external to the mobile device (*heterogeneity*) and more are increasingly available. A mobile instrumentation middleware needs to support ad-hoc support to hardware-, software-, or human-based sensors and allow *transparent* access to their data. More importantly, the data should be *portable* to other platforms besides the mobile device.

AWARE's Context Sensors abstract sensor data acquisition and processing into reusable context components, from local and external sensors. Moreover, AWARE's context sharing mechanisms, such as Context Providers, Context Observers and Context Broadcasts, offer *transparent* context data exchange between other AWARE add-ons and applications, both locally and remotely.

AWARE Add-ons provide the building blocks to extend the middleware to support novel contexts and sensors, in an *ad-hoc* basis. The AWARE Server components support clustering for load-balance and scale up the number of AWARE Clients in the future.

At the moment, AWARE Client is limited to Android devices. However, the AWARE Server functionalities, namely the web services JSON objects and MQTT messages, allow exchanging and using context information in other platforms.

4.3.3 Security and privacy

Context harvests a lot from the user, and a mobile context middleware should keep the user's data secure and private. AWARE offers control to the users on their personal data. AWARE supports encrypted connection and client authentication to the AWARE Server.

Despite our efforts to minimize security breaches, more robust security procedures must be considered to protect context data, which was not the focus of this thesis. AWARE does not collect personal data and hashes personal identifiers to protect users' privacy. Nonetheless, this alone might not be enough for other more privacy strict domains, such as healthcare.

4.3.4 Volatility

Context is volatile and mobile middleware should support runtime adaptation of context. AWARE uses a Context-Action and add-on approach to dynamically modify context in runtime.

Other methods exist to further enhance the context dynamics (Capra *et al.* 2003; Korpipää *et al.* 2004; Sørensen *et al.* 2004; Berri *et al.* 2006).

4.3.5 Reliability

Mobile phones are resource-constrained environments, in terms of network, storage, battery life and processing power. AWARE takes advantage of MQTT messaging protocol *resilience* mechanisms to exchange context data between different devices while being network efficient.

Not just to overcome storage limitations, some *redundancy* is supported. Context data can be stored locally and synchronized remotely. The local storage can then be restored. However, more sophisticated methods should be considered.

AWARE supports *delegation* and *process offload* for demanding data operations in remote capable machines. Despite our best efforts, we cannot guarantee a problem-free mobile data collection tool. AWARE is an Android Accessibility Service to increase its importance to the OS task manager and reduce termination likelihood. At five-minute intervals, the built-in watchdog checks active add-ons and services. AWARE supports Android 2.3 or higher; however, sensor deprecation might limit some functionality in the future.

4.3.6 Battery impact

In our case studies, we further evaluated the battery impact of the core sensors. Using a Samsung Galaxy S3 (1600mAh, 4.2V battery) as a reference device, we measured the power consumption of each sensor when in individual use, i.e. actively logging data (Figure 11). We used a multimeter connected to the device's battery.

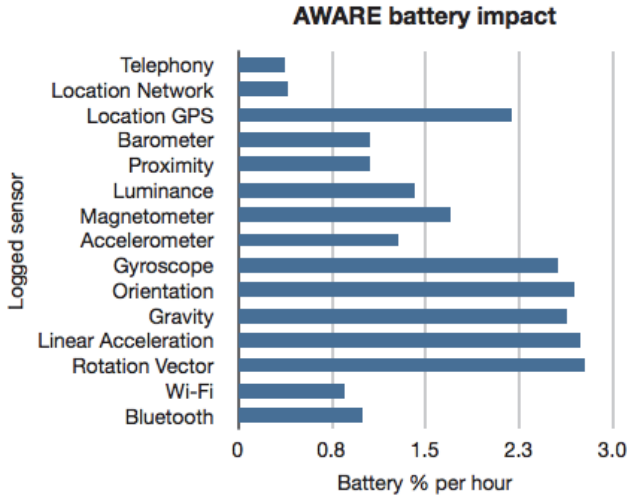

Fig. 11. AWARE's battery impact, in percentage per hour.

Figure 11 demonstrates that AWARE has an *acceptable* battery impact (min = 0.4; mean = 1.6; max = 2.8; SD = 0.9). Comparatively, when the device is not

collecting data, the reference device's battery depletes at a rate of 0.3% per hour. More importantly, in our case studies, participants did not notice a decrease in their device's battery life.

Similar to the literature review (Ipiña & Katsiri 2001; Meier & Cahill 2003; Falaki *et al.* 2011), we found that an event-based mobile instrumentation framework is more efficient than periodic polling the sensors for new information.

We are confident that novel battery technology and power-efficient sensors will reduce even further AWARE's battery impact.

The following sections describe in more detail how the thesis articles contributed to the development of the framework.

4.4 Case studies

In this section, we summarize how each of the thesis articles contributed to AWARE. All articles contributed to the theoretical framework, highlighted with roman numerals (I, II, III, IV, V) in Figure 12. See annexed original articles for specific research questions contributions.

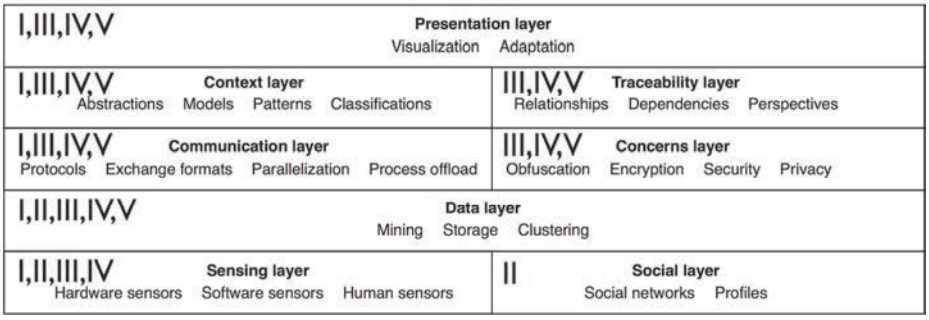

Fig. 12. A summary of thesis article's contributions on AWARE's theoretical architecture.

4.4.1 Article I - Human-smartphone awareness

Article I provides a better understanding on explicit interaction between users and mobile devices through ambient media. It also provides insight into researchers' challenges when designing, prototyping and evaluating ambient media. AWARE instrumented phone and messages events and shared communication context to an

ambient display, Ambient Notifier. Article I addresses challenges that contributed to the *sensing*, *data*, *communication*, *context* and *presentation* layer of the theoretical framework.

In the sensing layer, phone and message meta-data was collected to abstract the communication context. Accelerometer data was collected to abstract the motion context (i.e. shaking, not shaking). The concept of Context Sensor is also introduced in this paper. In the data layer, context was not stored locally, but remotely via HTTP POST to a remote server. The concept of the AWARE Server web service is introduced.

In the communication layer, context was exchanged between the mobile phone and the ambient media prototype, Ambient Notifier, resorting to TCP/IP sockets communication with RPC calls, updating a state machine in Ambient Notifier. RPC calls limitations motivated investigating support for different communication protocols and data exchange formats. The concept of Context Broadcasts is introduced.

In the context layer, phone and messages data was abstracted as communication context, namely a message received or sent, incoming call, rejected call, missed call, accepted call, user in a call, user not in a call. The accelerometer data was classified as shaking and not shaking to dismiss any ambient media notifications. The concept of Context Provider is introduced.

In the presentation layer, visualizing communication context from the phone needed adaptation for a different medium to support users' mental model of color-event matches.

4.4.2 Article II - Concerns of location sharing

Article II provides an understanding of the effects of trajectory reminders on users' ability to recall mobile location sharing preferences. AWARE instrumented location data and shared location context. Article II addresses challenges that contributed to the *sensing, social, data, communication* and *presentation* layer of the theoretical architecture.

In the sensing layer, location geo-locations were collected from participants' phones to abstract important locations context. Support for data labeling is introduced in the Context Providers.

In the social layer, the users provided data on strong social bonds, such as family, close friends and colleagues. This data was collected in questionnaires prior to deployment, but was not logged in AWARE. This motivated adding

support for user-provided context, and AWARE's ESM were introduced for later studies.

In the data layer, the location context is stored locally on the mobile phone. The participants were instructed to share their location history data at the end of the day. The AWARE Server web services data synchronization mechanism is introduced. In the communication layer, we were required to batch upload location data remotely. Android's native JSON support allows batch uploading an array of JSON location objects to the AWARE Server web services.

In the presentation layer, participants needed to interact with AWARE to share the location traces, and to temporarily disable the location logging, thus supporting context accountability. The Context Plugin is introduced. After synchronizing the data remotely, participants were required to perform an online questionnaire, with their data visualized. The AWARE Server Dashboard and data visualizations are introduced.

4.4.3 Article III - Human-smartphone proximity

Article III provides an understanding of users' proximity habits in relation to their mobile phones. AWARE instrumented the mobile device and external Bluetooth sensors, application usage, battery usage, phone and messages usage, locations, network coverage, screen usage and Wi-Fi usage. Article III addresses challenges that contributed to the *sensing*, *data*, *communication*, *concerns*, *context*, *traceability* and *presentation* layer of the theoretical architecture.

In the sensing layer, AWARE instrumented external Bluetooth dongles and the mobile phone's internal Bluetooth sensor to capture distance values calibrated from Received Signal Strength Indication (RSSI) values. AWARE also captured data from sensors that could potentially indicate the presence or proximity of the user to the mobile phone.

In the data layer, because AWARE collected data from multiple sources simultaneously, individual Context Provider per Context Sensor is required. In the communication layer, AWARE pinged AWARE Server web services to guarantee data collection at a five-minute interval. AWARE's watchdog service is introduced to monitor remote data collection.

In the concerns layer, personal identifiers such as MAC addresses where hashed for security and privacy concerns. AWARE's encryption and hashing mechanisms are introduced. In the context layer, AWARE abstracted Bluetooth RSSI values as distance metrics for users' proximity to the device. Other contexts

include users' location, time of day, application usage, device charging, communication, network, screen usage and Wi-Fi.

In the traceability layer, the dependencies and relationships between multiple contexts sources motivated the Context-Action abstractions for context inferences. AWARE Add-ons concept is introduced as reusable research components, manifested as Context Sensors and Context Plugins.

In the presentation layer, AWARE Client first interface is introduced to manage different Context Sensors statuses.

4.4.4 Article IV - Concerns of battery life

Article IV provides an understanding of users' concerns regarding battery life and charging routines. AWARE instrumented battery usage. Article IV addresses challenges that contributed to the *sensing*, *data*, *communication*, *concerns, context*, *traceability* and *presentation* layer of the theoretical framework.

In the sensing layer, AWARE collected a large number of AWARE Clients battery data. In the data and communication layer, the data stored locally and remotely stored via the AWARE Server web services data synchronization mechanism, triggered by the context of the user charging his device.

In the concerns layer, a secure connection was established with AWARE Server web services to protect the data. In the context layer, charging context and phone status context is introduced.

In the traceability layer, this was the first large-scale study instrumented with AWARE. The automated management of AWARE Device ID as UUID is introduced to link thousands of records to the AWARE Client that produced it.

In the presentation layer, AWARE library is bundled as an Android application and made available on Google Play application store. AWARE is a library for application developers. The application displayed battery context to the user, thus supporting context intelligibility.

4.4.5 Article V - Large-scale deployments with application stores

Article V provides an understanding on researchers' concerns regarding large-scale studies. AWARE reuses Article IV battery usage data. Article V contributed to the *data*, *communication*, *concerns*, *context*, *traceability* and *presentation* layer of the theoretical framework.

In the data layer, collecting a large number of battery data raised concerns of data storage scalability. MySQL servers can be clustered to overcome this limitation or future work could consider elastic cloud storage approach.

In the communication layer, collecting qualitative data from remotely distributed participants required a message-oriented protocol with push functionality. AWARE Server MQTT server ESM dashboard is introduced to issue in-situ ESM on the mobile phone.

In the concerns layer, being able to push information remotely to a device required obfuscation and encryption techniques to safeguard context exchange. AWARE's obfuscation and encryption methods are introduced. More importantly, privacy notices for study participants were introduced and displayed in the presentation layer.

In the context layer, Internet availability context is introduced. In the traceability layer, a different perspective on the collected battery data is created, focusing on a researcher's point of view.

In the presentation layer, the researcher data is shown in the AWARE Server Dashboard, focusing on monitoring study participation.

5 Conclusion

Mobile sensors are traditionally used for added application functionality and to improve user interaction (e.g. detect screen orientation). However, the literature on ubiquitous and mobile computing serves as evidence that mobile sensors can be leveraged to sense the user itself (Lovett & O'Neill 2010), a society (Eagle & Pentland 2005) or even a city (Kostakos *et al.* 2006). Mobile phones provide researchers an opportunity to understand better users' lives and surroundings. We started this thesis with the following research question:

> *"What are the challenges in creating a mobile instrumentation and context framework for studying human behavior, routines and context as a tool for researchers, application developers and users?"*

When Weiser described a ubiquitous computer (Weiser 1999), it was an invisible, distributed computing entity to serve humanity. We argue that collectively instrumented, modern mobile phones can be regarded as the ubiquitous computer, a single coherent system composed of mobile phone sensors and functionality.

In Weiser's vision of the 21st century computer, computing was described as "calm". However, we distance ourselves from "calm" as mobile phones keep seizing the users' attention intermittently, and are more "pervasive" instead. This is not necessarily a drawback. Never before have we been able to keep in touch with friends and family, anytime, anywhere.

A distributed system is a collection of independent computers that appears to its users as a single coherent system (Tanenbaum & Steen 2002). Weiser's predictions are happening right now, and instrumenting different devices and sensors together is a reality, as a distributed system.

5.1 Mobile instrumentation and context framework as a tool

In this thesis, we highlight important challenges when developing a mobile instrumentation middleware. The biggest challenge is the broad range of research domains in ubiquitous and mobile computing. Different research domains have different requirements regarding data and use. A mobile instrumentation middleware must support *multidisciplinary* research and *collaboration* from ground-up.

AWARE's theoretical framework grouped mobile instrumentation challenges into units, conceptualized as layers in Chapter 3 (Figure 2). Each layer

encompasses different research domains in the field of ubiquitous and mobile computing, but not all. New research domains emerge while others fade as the field matures. This means that a mobile instrumentation middleware needs to *adapt* and *evolve*. AWARE Add-ons provide a mechanism for extending, adapting and evolving the middleware's multidisciplinary research requirements in an ad-hoc basis.

AWARE Add-ons support research collaboration. By encapsulating research expertise into *reusable* Context Sensors and Context Plugins, researchers can share expertise more transparently and effortlessly. AWARE Add-ons support device and sensor fragmentation. New sensors and devices require implementation effort to instrument them. As Context Sensors, the implementation effort can be one-time effort, available for later reuse if required.

AWARE makes context available as a resource, not just for researchers, but also application developers and users. However, just having access to context information is just a first step. Context-aware applications must do a lot more: *present information*, *acquire and store context information*, and *relate new information to other captured information* (Dey *et al.* 2001). As a mobile instrumentation middleware, AWARE not only *collects*, *manages* and *shares* context data, but also *presents* context information.

A crucial challenge in mobile instrumentation remains battery life, which is still a bottleneck for ubiquitous and mobile computing. We assessed AWARE's acquisition and storage of context information capabilities, namely its *stability*, *scalability* and *reliability* while collecting large amounts of context data and keeping a small battery footprint. AWARE is event-based, from the devices' core sensors (Android is event-based) and AWARE Add-ons to the context exchange communication protocols (AWARE Server MQTT).

An event-based approach is not only battery efficient. Event propagation is taken care of by the publish/subscribe mechanism provided by AWARE Server MQTT for distributed deployments and by registering Context Observers for Context Providers or listening to broadcast events with Context Broadcasts locally. This ensures that interested Context Sensors and Plugins can listen to and process context information that is relevant to them, without much development effort.

Another challenge is network connectivity. AWARE leverages MQTT as a machine-to-machine (M2M) communication protocol, to exchange context data, but also to issue commands. MQTT QoS level 2-message exchange and low bandwidth requirements make it ideal for mobile middleware instrumentation.

Despite mobile phones' increasing storage capability, it is still limited. We combined local storage with remote storage using AWARE Server web services and a data synchronization mechanism that guarantees that both context repositories are identical and contain the most up-to-date context historical data, especially on the server side. Outside the scope of this thesis, we must acknowledge that AWARE middleware could use data compression techniques to minimize bandwidth use and storage.

5.2 Studying human behaviour, routines and context

By understanding users' context better, we can inform the design of a "pervasive" system. Contrary to an ubiquitous system that fades in the background, a pervasive system makes itself visible and discoverable, especially if it provides consumable resources (Mascolo *et al.* 2004).

In Article I, AWARE is used to understand explicit interaction between users and mobile phones, using ambient media as an interface for awareness. From a researchers' perspective, Article I also provides an understanding on the researchers' challenges when designing, prototyping and evaluating ambient media.

In Article II, AWARE is used to understand the effects of trajectory reminders on users' ability to recall mobile location sharing preferences. Sharing your location with others depends on several contextual factors. We investigated how consistently the users recalled their personal sharing preferences and tried to understand what were the contextual factors in place.

In Article III, we focused on understanding the users' proximity habits to their mobile devices; as to understand how often context information is presented to the user. We found that, contrary to our intuition, the device is not really with the user all the time. This emphasizes the importance of Article I research contributions on maintaining users' awareness by leveraging ambient displays.

In Article IV, we made AWARE available to thousands of users by embedding it as a library in a battery user-study application on Google Play application store. We wanted to understand users' concerns regarding battery life and charging routines.

Lastly, running a large-scale user-study on an application store contributed with important and significant challenges for mobile computing, in Article V. We synthetized our and previous literature findings as researchers' concerns regarding large-scale studies deployment by leveraging application stores.

The Articles I, II, III focused more on building and designing AWARE, while IV and V focused on evaluating AWARE as a research tool to study human behavior, routines and context. AWARE is no single ready-to-use, fits-all, "silver bullet" solution that would meet the requirements of all possible researchers.

The theoretical framework emphasizes AWARE's different components responsibilities, but also highlights how fragmented ubiquitous and mobile computing research really is. Context is dynamic and requires a middleware capable of adaptation, reasoning, inference, and many more in real-time. McKinley *et al.* (2004) distinguish three basic techniques to come to software adaptation: separation of concerns, computational reflection and component-based design. AWARE supports creating context via components, which we name as AWARE Add-ons.

In conclusion, this thesis answers and addresses many of the challenges in creating a mobile instrumentation and context framework for studying human behavior, routines and context as a tool for researchers, application developers and users. Naturally, there are still open challenges for AWARE, but we leave that for fellow researchers to contribute at http://www.awareframework.com.

References

Abowd GD, Dey AK, Brown P & Davies N (1999) Towards a Better Understanding of Context and Context-Awareness. Proc Handheld and Ubiquitous Computing (HUC ’99): 304–307.

Abowd GD (2012) What next, ubicomp? Proc UbiComp ’12: 31–40. Pittsburgh, USA, ACM Press.

Aharony N, Pan W, Ip C, Khayal I & Pentland A (2011) The social fMRI: measuring, understanding, and designing social mechanisms in the real world. Proc UbiComp ’11: 445–454. Beijing, China, ACM Press.

Android API - ContentProvider. URI: http://developer.android.com/reference/android/content/ContentProvider.html. Cited 2013/04/19.

Android API - BroadcastReceiver. URI: http://developer.android.com/reference/android/content/BroadcastReceiver.html. Cited 2013/04/19.

Android API - ContentObserver. URI: http://developer.android.com/reference/android/database/ContentObserver.html. Cited 2013/04/19.

Android API - Manifest. URI: http://developer.android.com/guide/topics/manifest/manifest-intro.html. Cited 2013/04/19.

Android API - AccessibilityService. URI: http://developer.android.com/reference/android/accessibilityservice/AccessibilityService.html. Cited 2013/04/19.

Android API - Service. URI: http://developer.android.com/reference/android/app/Service.html. Cited 2013/04/19.

Android API - Activity. URI: http://developer.android.com/reference/android/app/Activity.html. Cited 2013/04/19.

Ardon S, Gunningberg P, Landfeldt B, Ismailov Y, Portmann M & Seneviratne A (2003) MARCH: A distributed content adaptation architecture. Int J Communication Systems (16): 97–115.

Bardram J & Hansen T (2004) The AWARE architecture: supporting context-mediated social awareness in mobile cooperation. Proc CSCW '04: 192–201, Illinois, USA, ACM Press.

Bellotti V & Edwards K (2001) Intelligibility and Accountability: Human Considerations in Context-Aware Systems. Int J Human-Computer Interaction (16): 193–212.

Berri J, Benlamri R & Atif Y (2006) Ontology-based framework for context-aware mobile learning. Proc Communications and Mobile Computing (IWCMC ’06): 1307–1310, Vancouver, Canada, ACM Press.

Bettini L, De Nicola, R & Pugliese R (2002) Klava: a Java package for distributed and mobile applications. Int J Software: Practice and Experience (32): 1365–1394.

Biegel G & Cahill V (2004) A framework for developing mobile, context-aware applications. Proc IEEE Pervasive Computing and Communications (PERCOM '04): 361–365.

Böhmer M, Hecht B, Schöning J, Krüger A & Bauer G (2011) Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage. Proc MobileHCI '11: 47–56, Stockholm, Sweden, ACM Press.

Brown P & Bovey J (1997) Context-aware applications: from the laboratory to the marketplace. Int J IEEE Personal Communications (4): 58–64.

Brunette W, Sodt R, Chaudhri R, Goel M, Falcone M, Van Orden J & Borriello G (2012) Open Data Kit sensors: a sensor integration framework for android at the application-level. Proc MobiSys ’12: 351–364, Lake District, UK, ACM Press.

Buchholz T, Küpper A & Schiffers M (2003) Quality of Context: What it is and why we need it. Proc WS HP OpenView University Association:1–14.

Buennemeyer TK, Nelson TM, Clagett LM, Dunning JP, Marchany RC & Tront JG (2008) Mobile Device Profiling and Intrusion Detection Using Smart Batteries. Proc Int IEEE Conf System Sciences (HICSS '08): 296–296.

Capra L, Emmerich W & Society IC (2003) CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications. IEEE Transactions on Software Engineering (29): 929–944.

Carter S, Mankoff J & Heer J (2007) Momento: support for situated ubicomp experimentation. Proc CHI '07: 125–134, California, USA, ACM Press.

Chan ATS & Chuang S (2003) MobiPADS : A Reflective Middleware for Context-Aware Mobile Computing. IEEE Transactions on Software Engineering (29): 1072–1085.

Chen G & Kotz D (2000) A survey of context-aware mobile computing research (1): 2.1. Technical Report TR2000-381: 1–16.

Church K & Smyth B (2008) Understanding the intent behind mobile information needs. Proc Int Intelligent User Interfaces (IUI ’09): 247–256, Florida, USA, ACM Press.

CLIPS (1985). CLIPS: A Tool for Building Expert Systems. URI: http://clipsrules.sourceforge.net/. Cited 2013/04/20.

CodeIgniter (2010). CodeIgniter. URI: http://ellislab.com/codeigniter. Cited 2013/04/20.

Consolvo S & Walker M (2003) Using the experience sampling method to evaluate ubicomp applications. Int J IEEE Pervasive Computing (2): 24–31.

Corey G (2010) Nine Ways To Murder Your Battery (These Are Only Some Of The Ways). Battcon ’10 (13): 1–5.

D3js Data Driven Documents (2012). D3js. URI: http://www.d3js.org. Cited 2013/04/20.

David P & Ledoux T (2005) WildCAT: a generic framework for context-aware applications. Proc Int WS Middleware for pervasive and ad-hoc computing: 1–7.

Dey AK (1998) Context-Aware Computing: The CyberDesk Project. Proc AAAI 1998: 51–54.

Dey AK, Abowd GD & Salber D (2001) A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Int J Human-Computer Interaction (16): 97–166.

Dickerson R, Gorlin E & Stankovic J (2011) Empath: a continuous remote emotional health monitoring system for depressive illness. Proc Wireless Health (Art. 5), San Diego, USA, ACM Press.

Eagle N & Pentland A (2005) Reality Mining: sensing complex social systems. Int J Personal and Ubiquitous Computing (10): 255–268.

Fahy P & Clarke S (2004) CASS–a middleware for mobile context-aware applications. Proc WS Context Awareness, MobiSys '04, Boston, USA, ACM Press.

Falaki H, Mahajan R & Estrin D (2011) SystemSens : A Tool for Monitoring Usage in Smartphone. Proc MobiArch'11: 25–30, Maryland, USA, ACM Press.

Ferreira D, Kostakos V & Dey AK (2012) Lessons Learned from Large-Scale User Studies: Using Android Market as a Source of Data. Int J Mobile Human Computer Interaction (IJMHCI) (4): 28–43.

Froehlich J, Chen MY, Consolvo S, Harrison B & Landay JA (2007) MyExperience: A System for In situ Tracing and Capturing of User Feedback on Mobile Phones. Proc MobiSys '07: 57–70, Puerto Rico, USA, ACM Press.

Gaber MM & Yu PS (2006) A framework for resource-aware knowledge discovery in data streams: a holistic approach with its application to clustering. Proc ACM Symposium Applied Computing (SAC '06): 649–656. Dijon, France, ACM Press.

Gartner Inc (2012) Gartner Says 821 Millon Smart Devices Will Be Purchased Worldwide In 2012; Sales to Rise to 1.2 Billion in 2013. URI: http://www.gartner.com/it/page.jsp?id=2227215. Cited 2013/03/12.

Ginger.io (2011) Ginger.io. URI: http://www.ginger.io. Cited 2013/02/11.

Gu T, Pung HK & Zhang DQ (2005) A service-oriented middleware for building context-aware services. Int J of Network and Computer Applications (28): 1–18.

HTK (2003) HTK Speech Recognition Toolkit. URI: http://htk.eng.cam.ac.uk/. Cited 2013/04/11.

Ipiña D de & Katsiri E (2001) An ECA rule-matching service for simpler development of reactive applications. IEEE Distributed Systems Online (2): 1–7.

Kang S, Lee J, Jang H, Lee H & Lee Y (2008) SeeMon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. Proc MobiSys'08: 267–280, Colorado, USA, ACM Press.

Korpipää P, Häkkilä J, Kela J, Ronkainen S & Känsälä I (2004) Utilising context ontology in mobile device application personalisation. Proc Int Mobile and Ubiquitous Multimedia - MUM '04: 133–140, Maryland, USA, ACM Press.

Kostakos V, O'Neill E & Penn A (2006) Designing Urban Pervasive Systems. IEEE Computer (39): 52–59.

Leclercq M, Quema V & Stefani J (2005) DREAM: a component framework for constructing resource-aware, configurable middleware. IEEE Distributed Systems Online (6): 1–12.

Lim, BY (2012) Improving Understanding and Trust with Intelligibility in Context-Aware Applications. Ph.D. thesis, Human-Computer Interaction Institute, Carnegie Mellon University, USA.

Lim BY & Dey AK (2011) Investigating intelligibility for uncertain context-aware applications. Proc UbiComp '11: 415–424, Beijing, China, ACM Press.

Lovett T & O'Neill E (2010) Mobile Context-Awareness: Capabilities, Challenges and Applications. Proc Ubicomp '10: 539–540, Copenhagen, Denmark, ACM Press.

Mahmoud Q (2004) Middleware for Communications. Chichester, England, John Wiley & Sons, Ltd.

Mankoff J, Dey AK, Hsieh G, Kientz J, Lederer S & Ames M (2003) Heuristic Evaluation of Ambient Displays. Proc CHI '03: 169–176, Florida, USA, ACM Press.

Mascolo C, Capra L & Emmerich W (2004) Principles of Mobile Computing Middleware. Middleware for Communications (11): 261–280. John Wiley & Sons, Ltd.

McKinley P, Sadjadi S, Kasten E & Cheng B (2004) Composing adaptive software. IEEE Computer (37): 56–64.

Meier R & Cahill V (2003) Exploiting proximity in event-based middleware for collaborative mobile applications. Proc Distributed Applications and Interoperable Systems (DAIS '03): 285–296, Paris, France, Springer-Verlag.

Mikalsen M & Kofod-Petersen A (2004) Representing and reasoning about context in a mobile environment. Proc WS Modeling and Retrieval of Context: 25–35.

Miluzzo E, Lane ND, Fodor K, Peterson R, Lu H, Musolesi M, Eisenman SB (2008) Sensing meets mobile social networks. Proc Embedded Network Sensor Systems (SenSys '08): 337–350, North Carolina, USA, ACM Press.

MQTT (2011). URI: http://www.mqtt.org. Cited 2013/04/19.

MySQL (2008). URI: http://www.mysql.com. Cited 2013/04/10.

Narayanan AK (2001) Realms and states: a framework for location aware mobile computing. Proc WS Mobile Commerce (WMC '01): 48–54, Rome, Italy, ACM Press.

Nielsen J & Molich R (1990) Heuristic evaluation of user interfaces. Proc CHI 1990: 249–256, Washington, USA, ACM Press.

O'Neill E, Kostakos V, Kindberg T, Schiek AF, Penn A, Fraser DS & Jones T (2006) Instrumenting the city: Developing methods for observing and understanding the digital cityscape. Proc UbiComp '06: 315–332, California, USA, Springer-Verlag.

Oliver E (2010) The challenges in large-scale smartphone user studies. Proc Int WS Hot Topics in Planet-scale Measurement (HotPlanet '10): Art. 5. California, USA, ACM Press.

Olla P & Patel NV (2003) A framework for delivering secure mobile location information. Int J Mobile Communications (1): 289–300.

Parthasarathy S & Subramonian R (2001) An interactive resource-aware framework for distributed data mining. IEEE Technical Committee on Distributed Processing (52): 1–29.

Patel S, Kientz J, Hayes G, Bhat S & Abowd G (2006) Farther than you may think: An empirical investigation of the proximity of users to their mobile phones. Proc UbiComp '06: 123–140, California, USA, Springer-Verlag.

Quercia D & Hailes S (2005) Risk Aware Decision Framework for Trusted Mobile Interactions. Proc IEEE/CreateNet Int WS The Value of Security through Collaboration.

Rachuri K, Musolesi M, Mascolo C, Rentfrow PJ, Longworth C & Aucinas A (2010) EmotionSense: a mobile phones based adaptive platform for experimental social psychology research. Proc UbiComp '10: 281–290, Copenhagen, Denmark, ACM Press.

Raento M, Oulasvirta A, Petit R & Toivonen H (2005) ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. IEEE Pervasive Computing (4): 51–59.

Ramanathan N, Alquaddoomi F, Falaki H, George D, Hsieh C-K, Jenkins J, Ketcham C (2012) ohmage: an open mobile system for activity and experience sampling. Proc Pervasive Computing Technologies for Healthcare (PervasiveHealth '12): 203–204, California, USA.

Really Small Message Broker (RSMB). URI: https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=d5bedadd-e46f-4c97-af89-22d65ffee070. Cited 2013/04/20.

Reddy S, Mun M, Burke J, Estrin D, Hansen M & Srivastava M (2010) Using mobile phones to determine transportation modes. ACM Transactions on Sensor Networks (6): 1–27.

Ryan NS, Pascoe J & Morse DR (1997) Enhanced reality fieldwork: the context-aware archeological assistant. Computer Applications in Archeology, British Archeological Reports, Tempus Reparatum.

Schilit BN & Theimer MM (1994) Disseminating active map information to mobile hosts. IEEE Network (8): 22–32.

Schmidt A-D, Peters F, Lamour F, Scheel C, Çamtepe SA & Albayrak Ş (2008) Monitoring Smartphones for Anomaly Detection. Mobile Networks and Applications (14): 92–106.

Sharkey J (2009) Coding for life – battery life, that is. Google I/O Developer Conference session.

Sinderen MJ, Halteren AT, Wegdam M, Meeuwissen HB & Eertink EH (2006) Supporting context-aware mobile applications: an infrastructure approach. IEEE Communications Magazine (44): 96–104.

Sørensen C, Wu M, Sivaharan T, Blair GS, Okanda P, Friday A & Duran-limon H (2004) A Context-Aware Middleware for Applications in Mobile Ad Hoc Environments. Proc WS Middleware for Pervasive and Ad-hoc Computing (MPAC ’04): 107–110.

SQLite (2009). URI: http://www.sqlite.org. Cited 2013/04/19.

Tanenbaum AS & Van Steen M (2002) Distributed Systems: Principles and Paradigms. Upper Saddle River, New Jersey, Pearson Prentice Hall.

Weiser M (1999) The computer for the 21 st century. ACM SIGMOBILE Mobile Computing and Communications Review (3): 3–11.

Wichadakul D, Gu X & Nahrstedt K (2002) A Programming Framework for Quality-Aware Ubiquitous Multimedia Applications. Proc MUM '02: 631–640.

Zhang D (2007) Web Content Adaptation for mobile handheld devices. Communications of the ACM (50): 75–79.

Original articles

I Ferreira D, Kostakos V, Rogstadius J & Venkatanathan J (2010) Design, Prototyping and Evaluation of Ambient Media: Lessons Learned from the Ambient Notifier. Adj Proc of MobileHCI'10, Int WS on Tool-Support for Mobile and Pervasive Application Development (TSMPAD 2010), September 7–10, Lisbon, Portugal.

II Venkatanathan J, Ferreira D, Benisch M, Lin J, Karapanos E, Kostakos V, Sadeh N & Toch E (2011) Improving Users' Consistency When Recalling Location Sharing Preferences. Proc INTERACT '11: 380–387, Springer-Verlag.

III Dey AK, Wac K, Ferreira D, Tassini K, Hong J-H & Rojas J (2011) Getting Closer: An Empirical Investigation on the Proximity of User to Their Smart Phones. Proc UbiComp '11: 163–172, Beijing, China, ACM Press.

IV Ferreira D, Dey AK & Kostakos V (2011) Understanding Human-Smartphone Concerns: A Study of Battery Life. Proc Pervasive '11: 19–33, San Francisco, USA, Springer-Verlag.

V Ferreira D, Dey AK & Kostakos V (2011) Lessons Learned From Large-Scale User Studies: Using Android Market as a Source of Data. Int J Mobile Human-Computer Interaction (IJMHCI) (4): 28–43, IGI Global.

Original publications are not included in the electronic version of the dissertation.

442. Sorsa, Aki (2013) Prediction of material properties based on non-destructive Barkhausen noise measurement

443. Sangi, Pekka (2013) Object motion estimation using block matching with uncertainty analysis

444. Duan, Guoyong (2013) Three-dimensional effects and surface breakdown addressing efficiency and reliability problems in avalanche bipolar junction transistors

445. Hirvonen-Kantola, Sari (2013) Eheyttäminen, kestävä kehittely ja yhteyttäminen : integroivaa kaupunkikehitystyötä Vantaalla

446. Hannu, Jari (2013) Embedded mixed-signal testing on board and system level

447. Sonkki, Marko (2013) Wideband and multi-element antennas for mobile applications

448. Saarinen, Tuomas (2013) Temporal and spatial variation in the status of acid rivers and potential prevention methods of AS soil-related leaching in peatland forestry

449. Pirinen, Rauno (2013) Towards regional development by Higher Education Institutions : an empirical study of a University of Applied Sciences

450. Mäkinen, Liisa (2013) Improvement of resource efficiency in deinked pulp mill

451. Guo, Yimo (2013) Image and video analysis by local descriptors and deformable image registration

452. Pantisano, Francesco (2013) Cooperative interference and radio resource management in self-organizing small cell networks

453. Ojanperä, Tiia (2013) Cross-layer optimized video streaming in heterogeneous wireless networks

454. Pietikäinen, Martti (2013) Metalli- ja elektroniikkateollisuus Oulun eteläisen alueella : kehitys koulutuksen ja teknologian näkökulmasta

455. Pitkääho, Satu (2013) Catalytic oxidation of chlorinated volatile organic compounds, dichloromethane and perchloroethylene : new knowledge for the industrial CVOC emission abatement

456. Morais de Lima, Carlos Héracles (2013) Opportunistic resource and network management in autonomous packet access systems

457. Nardelli, Pedro Henrique Juliano (2013) Analysis of the spatial throughput in interference networks

Book orders:
Granum: Virtual book store
http://granum.uta.fi/granum/

UNIVERSITY OF OULU P.O. Box 8000 FI-90014 UNIVERSITY OF OULU FINLAND

ACTA UNIVERSITATIS OULUENSIS

ISBN 978-952-62-0189-4 (Paperback)
ISBN 978-952-62-0190-0 (PDF)
ISSN 0355-3213 (Print)
ISSN 1796-2226 (Online)

UNIVERSITY of OULU
OULUN YLIOPISTO