UNIVERSITY OF TRENTO

DOCTORAL THESIS

Bridging Sensor Data Streams and Human Knowledge

Author:
Mattia ZENI

Supervisor:
Prof. Fausto GIUNCHIGLIA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Knowdive Group
Department or Information Engineering and Computer Science

November 2, 2017

Declaration of Authorship

I, Mattia ZENI, declare that this thesis titled, "Bridging Sensor Data Streams and Human Knowledge" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: November 2, 2017

"I think that's the single best piece of advice: constantly think about how you could be doing things better and questioning yourself."

Elon Musk

Abstract

Bridging Sensor Data Streams and Human Knowledge

by Mattia ZENI

Generating useful *knowledge* out of personal big data in form of sensor streams is a difficult task that presents multiple challenges due to the intrinsic characteristics of these type of data, namely their volume, velocity, variety and noisiness. This problem is a well-known long standing problem in computer science called the Semantic Gap Problem. It was originally defined in the research area of image processing as "... the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation..." [Smeulders et al., 2000]. In the context of this work, the lack of coincidence is between low-level raw *streaming sensor data* collected by sensors in a machine-readable format and higher-level *semantic knowledge* that can be generated from these data and that only humans can understand thanks to their intelligence, habits and routines.

This thesis addresses the semantic gap problem in the context above, proposing an interdisciplinary approach able to **generate human level knowledge from streaming sensor data in open domains**. It leverages on two different research fields: one regarding the collection, management and analysis of big data and the field of semantic computing, focused on ontologies, which respectively map to the two elements of the semantic gap mentioned above.

The contributions of this thesis are:

- *The definition of a methodology* based on the idea that the user and the world surrounding him can be modeled, defining most of the elements of her context as entities (locations, people, objects, among other, and the relations among them) in addition with the attributes for all of them. The modeling aspects of this ontology are outside of the scope of this work. Having such a structure, the task of bridging the semantic gap is divided in many, less complex, modular and compositional micro-tasks that are which consist in mapping the streaming sensor data using contextual information to the attribute values of the corresponding entities. In this way we can create a structure out of the unstructured, noisy and highly variable sensor data that can then be used by the machine to provide personalized, context-aware services to the final user;
- *The definition of a reference architecture* that applies the methodology above and addresses the semantic gap problem in streaming sensor data;
- *The instantiation of the architecture* above in the **Stream Base System (SB)**, resulting in the implementation of its main components using state-of-the-art software solutions and technologies;
- *The adoption* of the Stream Base System in four use cases that have very different objectives one respect to the other, proving that it works in open domains.

Keywords: Big Data, Ubiquitous Computing, Pervasive Computing, Context Aware Systems, Computational Humanism, Semantic Gap, Sensor Data, Knowledge

Acknowledgements

I feel that this section is the best opportunity I have to thank all of the people who have helped me throughout my graduate career, and probably the last good opportunity to express my gratitude, in writing, to the many individuals who have supported me in this long and perilous journey.

Firstly, I would like to express my sincere gratitude to my advisor Prof. Fausto Giunchiglia for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study. He also taught me many more things than simply scholarly matters, which this section is far too short to list in their entirety. In the end, we had fun together.

I would like to thank all the Knowdive members for their devices to test my applications and more importantly for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years.

Another important thank goes to my dear colleague Enrico, who helped me many times in different situations, both from an academic but also a personal point of view. You're a good friend and we've been through a lot. I hope to continue working with you and do great things together.

On a more personal level, I must thank my patient and understanding girlfriend Valentina who supported me from the beginning. She has not only accepted my self afflicted impoverishment but has fed and clothed me on occasion. Most importantly, I have never had difficulty in leaving work in the office as coming home to Valentina is coming home to the most important girl in the world. Her support in my life outside and inside academic life has been and is invaluable and cannot be properly expressed here in few lines. She helped me in very difficult moments of my life and most likely I won't be here without her.

Mattia Zeni
University of Trento
December 2017

The work compiled in this thesis has been partially supported by:

- *the European Union's Horizon 2020 (H2020) research and innovation programme under grant agreement n. 732194, QROWD - Because Big Data Integration is Humanly Possible* `http://www.qrowd-project.eu/`
- *the European Union's Seventh Framework Program (FP7) under grant agreement 600584, Smart Society - Hybrid and Diversity-aware Collective Adaptive Systems: Where People Meet Machines to Build Smarter Societies*
  `http://www.smart-society-project.eu/`

Contents

List of Figures

List of Tables

List of Abbreviations

3G 3rd Generation. 38, 39

A Attribute. 21, 22, 24

AD Attribute Definition. 21, 22, 24, 60

AI Artificial Intelligence. 5, 9, 13, 17, 20

API Application Programming Interface. 34

AT Attribute Type. 22

AV Attribute Value. 60

C Component. 43

CA Container Architecture. 168

CPU Central Processing Unit. 38

CSV Comma Separated Value. 37

E Entity. 21, 24

EB Entity Base System. 6, 7, 23, 42, 47–50, 52, 61, 62

ET Entity Type. 19–22, 24, 47, 52

EU European Union. 30

FIFO First In First Out. 38

GDPR General Data Protection Regulation. 11, 13, 33, 45, 62, 63

GPS Global Positioning System. 9, 33, 37, 48, 51, 169

H High. 37

KB Knowledge Base System. 6, 7, 23, 47

L Low. 37

LTE Long Term Evolution. 38, 39

M Medium. 37

N None. 37

P Procedure. 60

S Streams. 43

SB Stream Base System. vii, xv, xvi, xix, 6–8, 11, 12, 15–18, 20, 23, 24, 27, 29, 33, 35, 38, 40–42, 46, 47, 50, 54, 56, 59, 61–63, 67–73, 77–79, 81–87, 90–95, 97–99, 101–103, 105–108, 111, 114–116, 118, 119, 123, 135, 136, 168, 170

SHA-1 Secure Hash Algorithm 1. 45

SMS Short Messaging System. 35

SPR Smartphone Penetration Rate. 4

SURI Semantic Universal Resource Identifiers. 20, 21, 24

SURL Semantic Uniform Resource Locators. 20, 21, 24

SUS Smartphone Usage Statistics. 4

URI Universal Resource Identifiers. 20, 21

URL Uniform Resource Locators. 20

URN Uniform Resource Names. 20

UTC Coordinated Universal Time. 43

Weak AI Weak Artificial Intelligence. 61

Author's Contributions

What follows is the list of publications of the author that are relevant to the work presented in this thesis.

Bahle Gernot Gruenerbl Agnes, Lukowicz Paul Bignotti Enrico Zeni Mattia and Giunchiglia Fausto (2014). "Recognizing hospital care activities with a coat pocket worn smartphone". In: *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*. IEEE, 175–181 **BEST PAPER AWARD**.

Caprini, Carlo et al. (2013). "TinyBox: Social, local, mobile content sharing". In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. IEEE, pp. 300–302.

Giunchiglia, Fausto, Enrico Bignotti, and Mattia Zeni (2017). "Human-like context sensing for robot surveillance". In: *INTERNATIONAL JOURNAL OF SEMANTIC COMPUTING* 2017.

Giunchiglia, Fausto et al. (2017a). "Analyzing the impact of students' time allocation on academic performance via smartphone and time diaries". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **SUBMITTED TO**.

— (2017b). "Mobile Social Media and Academic Performance". In: *International Conference on Social Informatics*. Springer, pp. 3–13.

Giunchiglia, Fausto et al. (2017c). "Mobile Social Media Usage and Academic Performance". In: *Computers in Human Behavior*, **SUBMITTED TO**.

— (2018). "Personal Context Recognition via Reliable Human-Machine Collaboration". In: *Pervasive Computing and Communications (PerCom), 2018 IEEE International Conference on*. IEEE, **SUBMITTED TO**.

Giunchiglia Fausto, Bignotti Enrico and Zeni Mattia (2017a). "Human-like context modelling for robot surveillance". In: *Semantic Computing (ICSC), 2017 IEEE 11th International Conference on*. IEEE, pp. 360–365.

— (2017b). "Personal context modelling and annotation". In: *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*. IEEE, pp. 117–122.

Zeni, M. and K. Weldemariam (2017). "Extracting Information from Newspaper Archives in Africa". In: *IBM Journal of Research and Development* 61.6.

Zeni, Mattia, Daniele Miorandi, and Francesco De Pellegrini (2013). "YOUStatAnalyzer: a tool for analysing the dynamics of YouTube content popularity". In:

Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 286–289.

Zeni, Mattia, Daniele Miorandi, and Francesco De Pellegrini (2016). "Understanding the Diffusion of YouTube Videos". In: *Proceedings of ECCS 2014*. Springer, pp. 309–319.

Zeni, Mattia, Ilya Zaihrayeu, and Fausto Giunchiglia (2014). "Multi-device activity logging". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. ACM, pp. 299–302.

Zeni Mattia Ondula Elizabeth, Mbitiru Reagan Nyambura Agnes Samuel Lianna Fleming Kala and Weldemariam Komminist (2015). "Low-power low-cost wireless sensors for real-time plant stress detection". In: *Proceedings of the 2015 Annual Symposium on Computing for Development*. ACM, pp. 51–59.

Dedicated to the only person who has been with me
all the time, me, myself, and I…

Part I

General Notions

Chapter 1

Introduction

This Chapter is intended to introduce the work presented in this Ph. D. thesis. We start by defining a simple motivating example that will facilitate the understanding of the methodologies and theories presented in this work. We then present the context according to which this work should be looked at. Then, we present which are the problems this thesis wants to tackle, starting from the motivating example and the context. Finally, we draft the solution to these problems, that is composed by a methodology, a reference architecture and an instantiation of this architecture in a working system used in four different use cases.

1.1 Motivating Example

Fausto has an appointment in the Trento city center today at lunch and he cannot afford to be late (again). The i-Log application installed on his smartphone knows about this because Fausto gave it permissions to check the agenda in addition to other personal information. At 11:30AM the application notifies Fausto that he has to leave for his appointment and proposes him the best route to cover to reach the parking lot. This apparently simple service that notifies about an appointment involves a lot of knowledge the system has to be aware of. First of all, the locations about where Fausto is now and where he has to be for lunch. Knowing the locations, the system is able to calculate the time necessary for reaching the destination considering additional information such as real-time traffic conditions in the area. Not only this, the system is aware that Fausto has a specific goal referred to the physical activity he has to perform every day, being it a total of 10.000 steps. Since today he didn't move much, the system wants to incentive Fausto to walk more and proposes a parking lot that is a bit further away. For this reason, the departure time from the office takes into account also this additional element. Additionally, the system has learned from previous experiences with Fausto that he always tends to be late and then adapted the departure time accordingly, anticipating it by 10 minutes. Fausto is having an amazing time at lunch and didn't realize that the time is going on. In fact, he has a very important meeting with a Professor coming from another university. Since the meeting is scheduled at 1PM but at 12:40 Fausto is still at the restaurant, the system notifies him with the same service used in the morning, but this time Fausto decides he wants to finish the conversation an ignores the notification. At this point the system elaborates a strategy, based on a past experience, that involves sending an email to Mattia, one of Fausto's postdoc, communicating about his delay. One month before in fact, Fausto was in a similar situation and decided to send an email to inform about the delay and advice that the meeting could start without him.

In such a scenario described above, which is common with many users, the AI system must have knowledge of the user preferences and habits among other heterogeneous information such as about his surroundings in order to provide the appropriate service at the right time. To have such knowledge, and most importantly, to structure it in a way that makes it easy to retrieve and use it, is not an easy task.

1.2 The Context

Understanding what people are doing in a context-aware manner and react accordingly is an active field both from a research but also an industrial point of view. The final goal of this task is to provide services that are highly personalized on the user and that ultimately will improve her quality of life. This has become a field that is gaining interest especially thanks to the increasing number of electronic devices that surround the user in every aspect of his life. This devices are so pervasive and powerful that can enable completely new possibilities that few years ago were not even foreseeable. As an example we can mention IoT devices in general and smartphones with wearables. Smartphones in particular are devices that are more intertwined with our lives. There are two dimensions of this phenomena:

- **The Smartphone Penetration Rate (SPR)** which is the measure of the number of users that own and use at least one smartphone. There are yearly conducted surveys that show an amazing 30,9%[1] of worldwide smartphone adoption in 2017, that corresponds to 2.5 Billion people. This value increases if we consider a country such the United States, with a total penetration of 81%[2] in December 2016. These numbers are seeing an incredible growth, for example in developing countries, where they moved from a 21%[3] penetration rate in 2013 to a 37%[3] two years later.

- **The Smartphone Usage Statistics (SUS)** that refers to a measurement of the time spent using smartphone in specific situations. A person uses the smartphone for an average of 3 hours[4] per day, that excluding the sleeping time, it corresponds to almost 19% of the time available during the day. In addition to the time spent using the phone, there are other interesting dimensions that should be considered. For example, 52%[5] of UK owners look at the device within 15 minutes after wake up in the morning, which increases to 86%[5] within one hour. Similar values can be found considering the time interval between when the user looks at the phone and when he goes to sleep, with a 43% within 15 minutes that raise up to 77%[5] within one hour. Also in the middle of the night, 34%[5] of the users claimed they check at least one time their smartphones for different purposes. Finally, these numbers are even more significant if we consider only a subset of the population, for example filtering by age. Millennials are the one that use more these devices, with 87%[6] of them saying that they never leave their smartphones during the day and 80%[6] saying that they are the first thing they look at when they wake up in the morning.

These statistics support our intuition that smartphones are the devices that are more intertwined with our lives. Another very important element is that they are

[1] http://goo.gl/5qGJBZ
[2] http://goo.gl/7zUEmG
[3] http://goo.gl/s5MqED
[4] http://goo.gl/ctXFLv
[5] http://goo.gl/t9krFC
[6] http://goo.gl/2KJFZF

facing an exponential increase in performances and features. Even a low-end smartphone today has enough memory, computational power and sensors to run most of the applications available.

All these considerations let us understand that the smartphone is the best candidate in a system that wants to tackle the situation described in the motivating example of previous Section. In fact, a device that is always with the user and that is powerful and flexible enough to run custom application can be used to collect huge amounts of data and to provide the results of the analysis in terms of context aware personalized services.

This thesis looks at many situations where an Artificial Intelligence (AI) system can help the user in improving her everyday life.

1.3 The Problem

Starting from the motivating example of Section 1.1 and within the context described in Section 1.2 the problem this thesis addresses is part of a long standing problem in computer science defined as *the semantic gap problem* [Smeulders et al., 2000]. In the context of this thesis we can refer to it as the lack of coincidence between the sensor data collected by the machine and the understanding of the situation the user has and the machine doesn't have. Thanks to her contextual information composed by habits, routines and ultimately her intelligence, the user has a completely different understanding with respect to the machine. The reason can be explained by the fact that the same sensor values can refer to multiple situations if no contextual information is provided. This is even more evident if we consider that two very similar situations can be perceived as very different to a person. If a person is very close to the door of a room, it is very different if she is inside or outside it. If she is inside, she can be attending a meeting, while if she is outside, probably she is at the copy machine one meter away from the door. From a machine point of view, the two positions can be assimilated with a unique point in space, due to the accuracy and errors in the measurement. To better explain this statement we can make an additional simple example: consider an application that collects location points about the places visited by the user. From an human perspective, the same location points collected by the machine in terms of coordinates can be interpreted very differently depending on the context. If the user has to communicate where she works to a new person met in a conference, she cannot reply with "I work in my office" but rather she will say something like "The University of Trento in Povo (TN)". As a counterexample if a user's friend asks where she is, she cannot reply using "The University of Trento in Povo (TN)" but she would prefer saying "I'm in my office". This situation shows that, depending on the context, a different output is enabled starting from the same sensor inputs, i.e., the physical coordinates of the University.

From this problem, we identified the following sub-problems:

1. **Data Acquisition and Management.** Collecting data from the people is not as easy as collecting data from other sources e.g., the web or sensor networks. When dealing with the user one must take into consideration all the *psychological and social implications* of such an operation. In fact, she must be engaged in collaborating in order to provide her data. Moreover, the privacy requirements for personal data are more strict with respect to other cases;

2. **Knowledge Generation.** Analyzing the sensor data in a context aware manner is not an easy task and this is why the semantic gap problem is still unsolved,

especially in open domain scenarios. There is the need to represent the different elements of the context involved in the analysis and create the appropriate methodologies to process sensor data;

3. **Knowledge Exploitation for Services.** Different users have different needs. This personalization aspect adds an additional challenge and must be taken into account when providing services to the user. Different solutions must be considered so that everyone receives the service she wants.

1.4 The Solution

This Ph. D. Thesis wants to find a solution to the semantic gap problem applied to personal data by giving the machine the necessary contextual information so that it can understand the user situations represented by the sensor data collected from her smartphone. In other words, the machine needs to analyze the collected data in a meaningful way for the human, ending up helping her in the every day life situations.

The contributions of this Ph. D. Thesis are:

- *The definition of a methodology* based on an interdisciplinary approach able to **generate human level knowledge from streaming sensor data in open domains**. It leverages on two different research fields: one regarding the collection, management and analysis of big data and the field of semantic computing, focused on ontologies, which respectively map to the two elements of the semantic gap mentioned above. The methodology is based on the idea that we can model the user and the world surrounding her, with an incremental method, defining most of the elements of his context as entities (locations, people, objects, among other, and the relations among them) in addition with the attributes for all of them. The modeling aspects of this ontology are outside of the scope of this work. Having such a structure, the task of bridging the semantic gap is divided in many, less complex, *modular and compoositional micro-tasks* which consist in mapping the streaming sensor data in a context aware way in the attribute values of the entities. In this way we can create a structure out of the unstructured, noisy and highly variable sensor data that can then be used by the machine to provide personalized, context-aware services to the final user;

- *The definition of a reference architecture* that implements the methodology above to collect, store, process and finally generate human level knowledge out of streaming sensor data that can be generated by a smartphone as well as any other IoT device;

- *The instantiation of the architecture* above in the SB using state-of-the-art software solutions to fulfill the requirements in terms of scalability, performance and efficiency. The system has three main goals: (i) to collect the streaming data from the users devices (smartphone and others) using a mobile application developed for this scope called i-Log [Zeni, Zaihrayeu, and Giunchiglia, 2014] and store them in a distributed database system; (ii) to apply the methodology defined in this thesis to generate human level knowledge from the sensor data using context information stored in the Entity Base System (EB) and Knowledge Base System (KB); (iii) finally, to provide a framework that makes the

human-level knowledge available to other applications that want to provide services to the final user;

- *The adoption of the SB* in four real-life use case with different goals in mind, proving that the system works in open domains.

1.5 Structure of the thesis

The remainder of this thesis is organized as follows:

- **Chapter 2** elicitates in details the different elements of the problem that are related to the task of generating high-level human knowledge out of streaming sensor data in a context aware way;
- **Chapter 3** describes the basic notions needed to fully understand the content of the thesis. In particular, those related to the representation and organization of the knowledge in the EB and KB. Moreover, since the knowledge schema is outside the scope of this work, a part of the whole schema needed for this work will be introduced in this chapter as well;
- **Chapter 4** explains the methodology we developed to solve the problems related to the data acquisition and management of personal big data from the user's smartphone;
- **Chapter 5** illustrates the methodology we developed to solve the problems related to the human knowledge generation from the personal big data in form of streams of sensor data collected and stored as described in the previous Chapter.
- **Chapter 6** describes the methodology we developed to solve the problems related to the knowledge exploitation for providing context aware services to the users that are meant to improve their quality of life;
- **Chapter 7** presents the second contribution of this thesis, the definition of a reference architecture that has been designed according to the methodology elements presented in Chapters 4, 5, and 6. We present the requirements, the logical view and a dynamic view that facilitate the reading of the next Chapters. The architecture is general in the sense that do not focus on any specific use case and do not sticks to any specific technology;
- **Chapter 8** describes in details the architectural solutions related to the acquisition and management of the users' data;
- **Chapter 9** describes in details the architectural solutions related to the generation of human knowledge from the streaming sensor data;
- **Chapter 10** describes in details the architectural solutions related to the exploitation of the generated knowledge to provide services to the users;
- **Chapter 11** describes in details how we instantiated the reference architecture into a real working prototype of the SB. It describes the technologies we used to satisfy all the strict requirements we defined;
- **Chapter 12** illustrates how we used and evaluated the implemented SB into two use cases run inside the University of Trento;

- **Chapter 13** illustrates how we used and evaluated the implemented SB into two use cases run outside a laboratory setting on the students of the University of Trento;
- **Chapter 14** presents the related work in different areas that are connected with the work presented in this thesis. Since our solution is highly interdisciplinary, in this Chapter different research communities are presented;
- **Chapter 15** wraps up the thesis, presenting the conclusions and possible research directions that will follow this work;

Chapter 2

The Problem

The goal of this thesis is to define a system able to transform huge amounts of sensor data into meaningful knowledge for the user in an open domain scenario. This knowledge will then be used by the AI to react to context changes and to provide personalized services that improve the user's quality of life.

However, this goal forces us to address a long standing problem in computer science, i.e., *the semantic gap problem*. It was originally defined in the research area of image processing as "... the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation..." [Smeulders et al., 2000]. The same issue applies within the work of this thesis, since the two sources of information that we rely on are humans and sensors. These two sources represent the world very differently, and their representations are not coherent. This makes it very hard, sometimes impossible, to understand that they actually represent the same real world. Consider for instance how both sensors and humans represent a location:

- **Sensors:** a location can be reduced to (a set of) coordinates, which may also be collected with different sensors with varying degree of granularity and noise. For instance, rooms within buildings are hard to be represented with current technologies embedded in smartphones with the same granularity of external buildings, which are easily detectable using the Global Positioning System (GPS) [Ladd et al., 2004].

- **Users:** Humans understand their surroundings via *context*, i.e., "a theory of the world which encodes an individual's subjective perspective about it" [Giunchiglia, 1993], which relates and make sense of different elements of humans' environment(s). In the case of locations, humans distinguish between different types of locations not only in terms of functions, e.g., my house vs my workplace, but also in relation to other elements such as social circles, e.g., family vs colleagues.

The misalignment between these two representations originates from the fact that the same sensor values can ideally refer to multiple situations if no further contextual information is provided. This is even more evident if we consider that for a person two very similar situations can be perceived as very different. In fact, sensors always collect position with a certain error, e.g., being in front of a door vs being inside, which for humans implies radically different situations, e.g., being in a meeting vs walking in the street.

This semantic gaps affect not only humans and machines, but also humans between themselves. For instance, if the user has to communicate where she works to a new person met in a conference, she cannot reply with "I work in my office" but rather she will say something like "The University of Trento in Povo (TN)". As

a counterexample if a user's friend asks where she is, she cannot reply using "The University of Trento in Povo (TN)" but she would prefer saying "I'm in my office". This additional layer of complexity only worsens the fact that, from the point of view of sensors, a different output is enabled starting from the same sensor inputs.

What the methodology and the system presented in this thesis wants to achieve is to give the machine these contextual information so that it will eventually be able to provide the right answer at the right time. In other words, we want to bridge the semantic gap so that the machine can analyze the collected data in a meaningful way for the human, ending up helping her in the every day life situations.

The overall problem can be split into different sub-problems that will be described in the next Sections: issues related to data acquisition and management, how to generate the knowledge and finally how to exploit the generated knowledge to provide highly personalized services to the user.

2.1 Data Acquisition and Management

The first element of the problem consists in having the data from the users that the system can use to do its analysis. This is a non-trivial issue considering its different dimensions: how to collect the data, how to store it efficiently and finally how to access these data.

Considering that the knowledge modelling aspects are outside the scope of this thesis (a summary is presented in Section 3), in this Section we refer only to the sensor data collected by the machine from the user personal devices. This task is usually called Lifelogging and is defined as "a record of a person's everyday life produced by a portable camera and/or other digital device which the person regularly carries around with them[1]". In this work [Gurrin, Smeaton, and Doherty, 2014] they refer to Lifelogging as "Personal Big Data" and we believe this definition perfectly fits with what this thesis presents. Dealing with big data adds additional challenges and this affects every element of the anayslis and the system.

2.1.1 Data Collection

Data collection is defined as the process of gathering and measuring information on targeted variables in an established systematic fashion. In Computer Science there are thousands of approaches to collect information from the multitude of sources available, the web in general, social networks in particular, sensors, among others.

Since we are dealing with personal data and then ultimately with the people, this task involves an additional challenge related to the **interdisciplinarity**. In fact, interacting with the users requires additional competences from other field of studies such as psychology, social sciences and humanities in general. The result is that all the dimensions of a typical data collection problem such as data quantity, quality and privacy are conditioned by the interdisciplinary factor.

Data Volume. Collecting data from the users is not as easy as collecting data from other sources e.g., sensor networks. The reason for this is that when dealing with the people one must take into consideration all the *psychological and social implications* of such an operation. It is in fact not enough to develop a system able to collect data in the most efficient way since if the user does not collaborate, the data

[1] http://www.macmillandictionary.com/buzzword/entries/lifelogging.html

cannot be collected. To facilitate this collaboration the user must be *engaged*.

Data Quality. While methods to ensure the proper volume of collected data vary by discipline, the goal for all of them is to capture quality evidence. Considering the problem presented in this thesis we identified two key quality metrics: *the sensor sampling rate* and *the presence of multiple sensors*. All the considerations made to design the data collection task must take into account these two dimensions, otherwise the data won't be good enough to provide meaningful results.

Data Privacy. When dealing with personal data the privacy and in general any ethical issue that can arise from their collection and exploitation are very important. The European commission will introduce (starting from May 2018) the Directive 95/46/EC called General Data Protection Regulation (GDPR)[2] that will completely change the way personal data are managed by organizations. In a system such as the SB we must take into account all the elements related to the privacy of the users. We will present them in the data collection even if we will make extensive use of them more when the data are accessed to be exploited.

2.1.2 Data Storage and Retrieval

If in the case of the data collection task there are multiple interdisciplinary factors to take into consideration, on the other hand in the case of data storage the problem is more on the technical side. The key point is to find the best solution able to store the collected data and deal with all their characteristics, the volume, the velocity at which they are generated and finally their variety of formats.

Data Volume. Personal big data can easily reach big volumes and on resource constrained devices such as the smarpthones, this constitutes a problem. Specific solutions must be designed to address this problem that are general and do not focus on a single use case of technology.

Data Velocity. The problem is even bigger considering that the memory can be filled up in a short period of time due to the speed these data are generated, usually in days.

Data Variety. The personal data collected from the smartphone are of different types that have heterogeneous formats. The best solution to store each of them is needed.

Storing the data is only half of the problem. After the information has been stored it needs to be extracted for the contextual analysis. The context in this thesis (as will be defined in Section 3.1) is the personal view of the world of the user in a specific moment in time. The rate at which the context changes varies on the situation but can be short, then for creating a context-aware system the real time aspect is crucial. To generate insights in such a short time the data must be always available and most importantly they must be retrieved very quickly from the storage systems. This is a non-trivial task because affects the choices made in the selection of the database technology, the data schema and also the hardware configurations. Details about this will be provided in Section 11.

[2] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC

2.2 Knowledge Generation

The knowledge generation task is at the core of the methodology and the SB presented in this thesis. There are different ways the system can leverage on for generating meaningful knowledge to the user, each one comes with its own problems: how to instantiate the knowledge, both in a semi-automatic or in a manual way and how to update the existing knowledge automatically by exploiting the data collected from the users' smartphone.

2.2.1 Knowledge Instantiation

They key element of transforming sensor streaming data into meaningful knowledge automatically stands in the ability of the system to map the raw data into the property values in the knowledge (we refer to Section 3 for notions required to understand the knowledge elements of this research). This involves the creation of a systematic way that is able to:

Select the Appropriate Streams of Sensor Data. Since the data collection task is costly, this phase must take into account to use the minimum required information to produce accurate results.

Select the Right Time Intervals. Retrieving the information from the database is an expensive operation if the mapping task has to be performed in semi real time. Then, there is the need to extract only the necessary data.

Select the algorithm. Select the most efficient machine learning algorithm for the task and the data in input, that is efficient enough to process the data in a short period of time to allow the real time dimension.

Select the Entity Attribute. Once the result is generated, the entity attribute value has to be updated with the one generated from the algorithms, this involves the search in the Knowledge Graph.

2.2.2 Knowledge Update

We believe the user is the best candidate to annotate her own data. The annotation task we are referring to is the one that usually is performed by a field expert when supervised machine learning algorithms are used. In a situation such the one described in this work, in an open domain scenario and where the possible context elements to be recognized are not constrained, the idea of using an external expert annotator is unfeasible. By looking at the data, even an expert cannot really understand the specific situation the user was involved in. On the other hand we believe the user can help in this task since she has her own personal representation of what's happening that is for sure meaningful to her. This of course presents some challenges:

Don't Bother the User. Interacting with the user in a systematic way can be very useful but also very dangerous. In fact, if she feels she is asked too many things too frequently, she can decide to quit.

Represents all the Answers using a Time Diary. Linked to the previous element there is the fact that short and fast to select answers must be presented to the user. This implies additional challenges in the way the time diaries are adapted.

User Unreliability. The user should be incentivized to help the machine in understanding his own context. But we cannot assume that her annotations are always realiable, since she is not an expert.

Finally, another way the human can be kept in the loop and help the machine to improve the context recognition task is through another type of interaction usually is referred to as active learning. On a non-regular basis, whenever the machine cannot decide among one possible representation of the real world to choose from as the result of the analysis, it can ask directly the user. Even if the task is different with respect to the previous one, the challenges that can arise are the same.

2.3 Knowledge Exploitation for Services

The final goal of generating meaningful knowledge out of streaming data is ultimately to generate an AI that can improve the quality of life of the user by providing useful, highly personalized services at the right time in the right place. The challenges that this task represents are the following:

User's privacy in accessing the data. We already presented some information about the privacy of the users and GDPR while explaining the data collection. In this situation, where the data is accessed for providing the service, these privacy issues become even more important. A whole security pipeline must be implemented to satisfy all the requirements in terms of authentication, anonoymization, access control.

Select which services are more suitable to satisfy the user needs. It is not obvious which are the services that can be created out of the user's data. Focus groups should be executed to understand the user needs and requirements for the different use cases. Moreover, in order to obtain the highest user satisfaction, all the services should be personalized on the user needs and configurable.

Services from different providers. Multiple providers should be able to provide services to the users since everyone of them should be expert and have information in its specific field of work. For example, a Municipality can provide services that are different from the one provided by an University or a National Health System.

How to provide the service. It is not obvious neither how to provide the services to the users, if on a desktop application or directly on the mobile device. Additionally, even if the mean is clear there are many possibilities to provide the service and all of them must be studied.

Chapter 3

Ground Knowledge

This chapter is intended to introduce the general notions used to set the ground knowledge for the content that is presented throughout this thesis. Therefore, we define the basic elements that will be used as building blocks for the SB.

We start by presenting the notions of user, smartphone, sensor, context, service, entity and entity attribute. We illustrate the semantic schema we adopt in this thesis for representing the data. Then we illustrate the instantiation process that allows to create the actual data that describes the users and their context. Finally we present the set of simplified entities the SB uses.

Acknowledgement. The knowledge modelling aspects are outside of the scope of this work. In fact, we leveraged on previous work developed by the members of the Knowdive[1] group, in particular [Fernandez and Ignacio, 2012], [Hume Llamosas, 2014] and the work of Enrico Bignotti.

3.1 Fundamental Notions

This Section explains the definitions of some key notions that will be extensively used in the SB and in the next Chapters of this thesis.

User. The user is the person who uses the SB and leverages on its services to improve her quality of life. In exchange of the services, most (or all) of them free-of-charge, the user provides her data, so that they can be analyzed and the insights can be generated. The data she provides are sensor data collected from her smartphone, referred to as personal big data and additional high-level knowledge in form of annotations using semantic labels. This knowledge is seen as an help to the AI that provides the user's personal view of what is happening.

Smartphone. The smartphone is a mobile device that performs many of the functions of a computer, typically having a touchscreen interface, Internet access, and an operating system capable of running third parties applications. Additionally in the context of this thesis such a device has also sensing capabilities. The internal sensors can sense different elements of the person, like the position, the movement, her social life, and also environmental conditions. Every device has a different set of sensors, every sensor with different characteristics like sampling frequency, sensitivity, among others.

Sensor. The most common definition of sensor states that "a sensor is an electronic component, module, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics, frequently a

[1] http://disi.unitn.it/~knowdive

computer processor. A sensor is always used with other electronics, whether as simple as a light or as complex as a computer."[2]. There are sensors for every purpose and of every size but, in general, there are three main characteristics the all sensors must have:

- It is **sensitive** enough to measure the *property* it has been designed for.
- It is **insensitive** to any other property.
- It does not **influence** the measured property.

For the sake of simplicity in the next Chapters we will refer only to sensors embedded inside smartphones. This doesn't mean the SB cannot manage other sensor sources. In fact, it has been designed to accept streaming sensor data from any device through dedicated data input pipelines.

With respect to dedicated sensing devices, using a smartphone to sense the user has both advantages and disadvantages. The former comprehend: (i) the fact that they can easily be deployed outside lab settings since they require to install only a dedicated application on the smartphone; (ii) this translates in a cheaper setup with respect to dedicated devices; (iii) finally, since the data collection is unobtrusive and does not require additional bulky devices, truthful data is generated, since the user doesn't perceive to be monitored. At the same time the disadvantages are: (i) a less accurate measurement since the internal sensors to the smartphone are not explicitly designed for this purpose and (ii) the fact that the measurement affects the battery life of the device and countermeasurements must be taken into account. We believe that the advantages are far more important then the disadvantages and this is why we are using the smartphone to sense the user.

Context. One of the first definitions of context is the one by [Giunchiglia, 1993], which defines it as "a theory of the world which encodes an individual's subjective perspective about it". A second definition that better refers to the work presented in this thesis, which belongs to the field of ubiquitous computing is "...any information that can be used to characterize the situation of an entity (person, place, physical or computational object) that is considered relevant to the interaction between the entity and application" [Abowd et al., 1999]. The core entity is the *user*; every user has her own personal view of the world, that is originated from her habits and routines and that is composed by all the elements (i.e., people, locations, objects, events) that surround her and the relations among these elements. The *context* is a part of this view that accounts for the **temporal dimension**. In other words, all the context situations we encounter compose our personal view of the world. A formal definition of this is defined by the tuple

$$MyWorld =< \{Cxt_{T0}, Cxt_{T1}, ..., Cxt_{TN}\} > \tag{3.1}$$

where *Cxt* is the (real world) context, aggregating different elements surrounding the user at a specific time instant Ti. Furthermore, the context is defined by the tuple

$$Cxt =< WA, WE, WO, WI > \tag{3.2}$$

[2] https://en.wikipedia.org/wiki/Sensor

where:

- **WE** is the Spatial context, i.e., the context generated from the question "**W**h**E**re are you?". For instance, an office or my house.
- **WA** is the Temporal context, i.e., the context generated from the question "**W**h**A**t are you doing?". For instance, being in a meeting or taking the bus.
- **WO** is the Social context, i.e., the context generated from the question "**W**h**O** are you with?". For instance, my girlfriend or my colleagues.
- **WI** is the Artifact context, i.e., the context generated from the question "**W**hat are you w**I**th?". For instance, my smartphone.

Service. A service in this thesis is an output produced by the AI that is meant to help the user in her everyday life situations. In order to provide a personalized service at the right moment in time the machine has to be aware of the user's context [Jones, 2008]. The specific nature of the service is domain and device dependent and in general all the variances are possible with the SB thanks to its modularity and adaptability. These are some of the services we already developed for our vertical solutions:

- **Quantified self**[3] **reports.** The concept Quantified Self, refers to a movement aimed at incorporating technology into data acquisition on all the aspects of a person's daily life, from the food consumed, to the quality of surrounding air, the mood, health related aspects performance, whether mental or physical. It is considered self-knowledge through self-tracking with technology. In the last years many works in the research community started focusing on it [Lupton, 2016], [Swan, 2013], [Fox and Felkey, 2016], [Swan, 2012a] and [Swan, 2012b]. Our thoughts about this are that a person lies to herself in an involuntary manner. We always believe we perform better than we actually do and the quantified self helps in this regard by showing us what we really are and allows to improve ourselves. Having a service that, on a fixed time interval basis (daily, weekly, etc), presents the user a quantified self report will really help her in being self aware and improve her quality of life.
- **Alerts.** With such a pervasiveness in our lives, the smartphone is the perfect device to provide services. One of its main characteristic is that it allows third-party applications to run and communicate in realtime. Designing a service that, based on the context, generates and presents alerts to the user can really provide an added value to her. The notifications themselves will be domain dependent and the design specifications will vary based on it.
- **Dedicated applications.** Third-party applications out of the scope of the SB can be developed as well to exploit its context-aware insights. These applications can be used as an interface with other systems that will enable additional service in a domain dependent manner. Examples regard smart homes ad smart environments in general as well as everything that is related with the internet of things.

Entity. There are multiple definitions of entity but the one we refer to is a "thing" that exists in the real world. Entities can be defined as abstract or physical

[3] http://quantifiedself.com/

objects, can be of different types (e.g., person, location, event, artifact) and are described by attributes (e.g., name, position, address, etc.) which vary based on the type of the entity but also on the domain. Different domains want different attributes for the same entity, e.g., in the educational domain is enough to provide few attributes to the entity type person, like name, address, faculty, while in a medical domain much more attributes should be provided at a much higher level of details like blood type, among others. Within the SB we formalize the notion of entities and we adopt the entity-centric approach that uses entities as the basic element of the knowledge.

Entity attribute. The attributes (also called properties) of an entity are those that characterize it. Different entity types differ one respect to the other thanks to a different set of attributes. Moreover, entities of the same entity type differ thanks to different attribute values. Even if the knowledge schema is outside of the scope of this thesis and for this reason we present it in a simplified way (Section 3.2), it is worth spending some few words on the entity attributes since they are the one the SB will more frequently interact with.

In a very simplified way, we can say that the process of bridging the semantic gap defined by our methodology and implemented in the StreamBase (SB) system consists in updating the entity attribute values based on a context aware analysis.

In addition to these definitions the following additional considerations must be taken into account:

- One user may have different representations of the same entity, we call it **intra-user** difference. In this situation, the different versions differ by the name and/or by the set of attributes they have. The set of attributes depends on the context. In fact, one representation in one context may require additional and/or different attributes with respect to another representation in another context.
- Users represent their own "versions" of the same global entity, we call it **inter-user** difference. As a consequence, different users may describe different aspects of the same entity because everyone focuses on different details.
- The user has only a partial view of the world, we called it *MyWorld* above. This is the consequence of the fact that different users may know an entity for only a subset of its attributes.

The work produced in this thesis up to now refers to the "personal world" of every user, but the methodology and system described in this work will benefit from the interoperability across different users, in terms of knowledge. In other words, sharing the entities between users can help everyone in improving each personal view of the world. For this to happen there is the need to find an agreement on the formal models that they will follow in representing the data. The entity-centric approach adopted in this thesis helps in this regard since it distinguished between the *schema* that defined this format and the *knowledge* that comprehends the actual data. In the following sections we formally present these elements.

3.2 Knowledge Schema

There are multiple definitions of schema but the one that best fits with the work presented in this thesis is the one provided by the Schema.org[4] initiative. A schema is "A set of types, each associated with a set of properties and where the types are arranged in hierarchy.". Our approach is aligned with this idea and allows the definition of templates for each type of entity defined in the system. The objective of using such templates is to restrict the set of possible attributes that can be used to describe that type of entity. In our implementation the meaning of each entity and property is further defined by mapping the single elements from the schema to concepts from a knowledge base. The word *concept* has different definitions: it is defined as "an abstract idea" in the Oxford dictionary[5] while Wikipedia[6] defines it as "An idea, something that is conceived in the human mind". In the area of knowledge representation a concept is used to formalize and represent the meaning of a word in a language independent manner.

Let's now move to the definition of the main elements our schema will be made of, namely entities and their attributes.

An **Entity Type (ET)** is formalized as a tuple

$$ET =< C, \{AD\} > \tag{3.3}$$

where,

- *C* represents a concept associated to the name of the entity type and which defines the class of entities that are described in it;
- *AD* is a non-empty set of attribute definitions denoting the type of attributes that can be used to describe an entity of the corresponding *ET*. Some of the attributes are mandatory while some others are optional and depend, for example, on the current representation the user is referring to, or in other words, on the context (intra-difference).

The notion of **Attribute Definition (AD)** is aimed to explicitly state constraints regarding how can be describe certain property of an entity. It is formally defined as the tuple

$$AD =< C, AT > \tag{3.4}$$

where,

- *C* is a concept associated to the name of the attribute, which provides a meaning for the property that an instance of the corresponding AD is describing;
- *AT* is a data type that establishes constraints on the values for the definition of the attribute. We can distinguish among those that are natively supported by the system (e.g., integer, string, float, data, relational, etc.), complex concepts from a knowledge base and the application defined *ETs*.

[4] http://schema.org/
[5] https://en.oxforddictionaries.com/definition/concept
[6] https://en.wikipedia.org/wiki/Concept

3.3 Instantiation of Knowledge

The additional step required to use the knowledge which is based on the schema presented in Section 3.2 in a real application is to instantiate it. An entity type ET is instantiated to represent a particular representation of a real world entity.

3.3.1 Entity Identifiers

Each entity is unique within a pre-defined context and its uniqueness, that allows to distinguish it from other entities, is given by its identity. The identity is defined by the intrinsic (e.g., that belongs by nature) or extrinsic (e.g., acquired from the world) characteristics of the entity [Do Van Thanh, 2007]. It is a fundamental notion when dealing with entities because it allows to understand their relations one respect to the other and with other elements of the context [Windley, 2005; Camp, 2004].

Starting from the notion of identity, the identifiers are used to identify a person, a location or, in general, any type of entity within a context. The same entity can have multiple identifiers that are used for different purposes or in different contexts. In the SB we can distinguish between two main types of identifiers: the ones used by humans, also called human-understandable identifiers and those used by the machine called machine-understandable identifiers.

Humans usually refer to someone or something (any entity) by its name because for them the name is uniquely mapped to the representation they remember of such entity [Fernandez and Ignacio, 2012]. The context dimension is implicit when dealing with humans since we always think in a context aware manner. Of course, an entity can be called by multiple names (e.g., the building in via Sommarive 14, Povo (TN) can be referred as the University of Trento, the Department of Information Engineering and Computer Science, my work place) and at the same time different entities can be referred by (called using) the same name (e.g., the university has multiple locations scattered around the city but one can call each of them the University of Trento) as a consequence of being arbitrary assigned. This makes perfect sense for a person, since she can understand the context and disambiguate the name based on the situation but makes impossible for the machine to do the same operation. For this reason, we need machine-understandable identifiers to allow the AI to refer to entities.

In the case of machine-understandable identifiers, the best choice is then not the name, but something that can be uniquely solved by the computers. Among the many standards available in the WWW for digital identifiers the most widely used are Universal Resource Identifiers (URI)[7], Uniform Resource Locators (URL)[8] and Uniform Resource Names (URN)[9]. In the SB we need to distinguish the different representations of the same real world entity (inter- or intra-difference) while still maintaining track of the global entity the representation is referring to. With the purpose of distinguishing between local and global identifiers, the work [Fernandez and Ignacio, 2012] creates two new identifiers, called Semantic Uniform Resource Locators (SURL) and Semantic Universal Resource Identifiers (SURI).

A SURL is defined as a semantic URL that represents a particular representation (from the user point of view) of a real world entity. A SURL is created for each entity being represented by the user, it is globally unique and can be dereferenced to obtain the full representation of the entity. In other words, it encodes the location of a

[7] RFC1630
[8] RFC1738
[9] RFC1737

particular representation of a real world entity. A SURI is defined as a semantic URI that represents a real world entity without attaching it to a particular representation. The same SURI is shared by different users describing the same real world entity, it is also globally unique. A SURI cannot be directly used to retrieve an entity representation, because it does not commit to one single representation and it rather includes the different points of view from which an entity is represented. Differently from other approaches from the Semantic Web that combine URIs and URLs to identify entities in the Web (e.g., OKKAM, semanticweb.org[10], www.w3.org[11]), the separation between local and global identifiers allow us to split the identification of a real world entity and its representation(s). Further, other approaches implicitly impose a representation for the real world entity when reusing the identifier, while we (by adopting the local/global identifiers) embrace diversity with regard to the point of views represented by different users.

3.3.2 Entity Instances

As described above, the knowledge we are referring to in this thesis is represented through the instantiation of the models described in Section 3.2.

An **Entity (E)** is an instantiation of an entity type ET and describes a real world entity from a specific point of view (i.e., of the person). This representation is associated with a set of attributed that characterize the entity. Additionally it is also attached with the identifiers that are used to refer to it, both the human-understandable and the machine-understandable identifiers. Formally, it is defined as a tuple

$$E =< SURL, SURI, \{N\}, ET, \{A\} > \tag{3.5}$$

where,

- *SURL* is the unique (personal) identifier;
- *SURI* is the unique identifier of the real world entity the E is describing;
- *{N}* is a set of strings representing the names used by the corresponding representation E to identify the real world entity;
- *ET* is the entity types among those allowed in the system;
- *{A}* is a non-empty set of attributes that characterize the entity.

An **Attribute (A)** instantiates an attribute definition Attribute Definition (AD) that represents a characteristic of the entity he refers to. An attribute is attached with a value that can be of different type: some attributes may have multiple values, its values can be mapped to a meaning in the Knowledge Base (i.e., a semantic value) or can refer to another entity in the system (i.e., relational attribute). Formally, it is defined as the tuple

$$A =< AD, \{V\} > \tag{3.6}$$

where,

- *AD* is an attributed definition defined in the knowledge schema for that system and is meant to constraint on the possible values the attribute can have;

[10] http://semanticweb.org/wiki/Uniform_Resource_Identifier
[11] http://www.w3.org/TR/cooluris/#semweb

- *{V}* is the set of attribute values of the type Attribute Type (AT) of the corresponding AD. If that corresponding AT is an ET then A is called a relational attribute since it defines a relation between two entities. The possible relations are many, e.g., my office is **part-of** the university building.

An example of how the knowledge schema is instantiated into two entities (for the inter-difference situation), simplified for the sake of clarity, is presented in Figure 3.1. In this case we have one ET instantiated by two different users (User1 and User2) into two personal representations that refer to the same global entity. The two instantiation both have the mandatory attributes Name, Address, Building, but they differ for the optional one. In fact, User2 in this case represents this Location with an additional attribute that is important for him: the *Reached by* attribute. This element is important because he can choose between different means of transportation. Depending on the mean of transportation he uses, for example, the time required to go to work changes and also the building entrance is different.

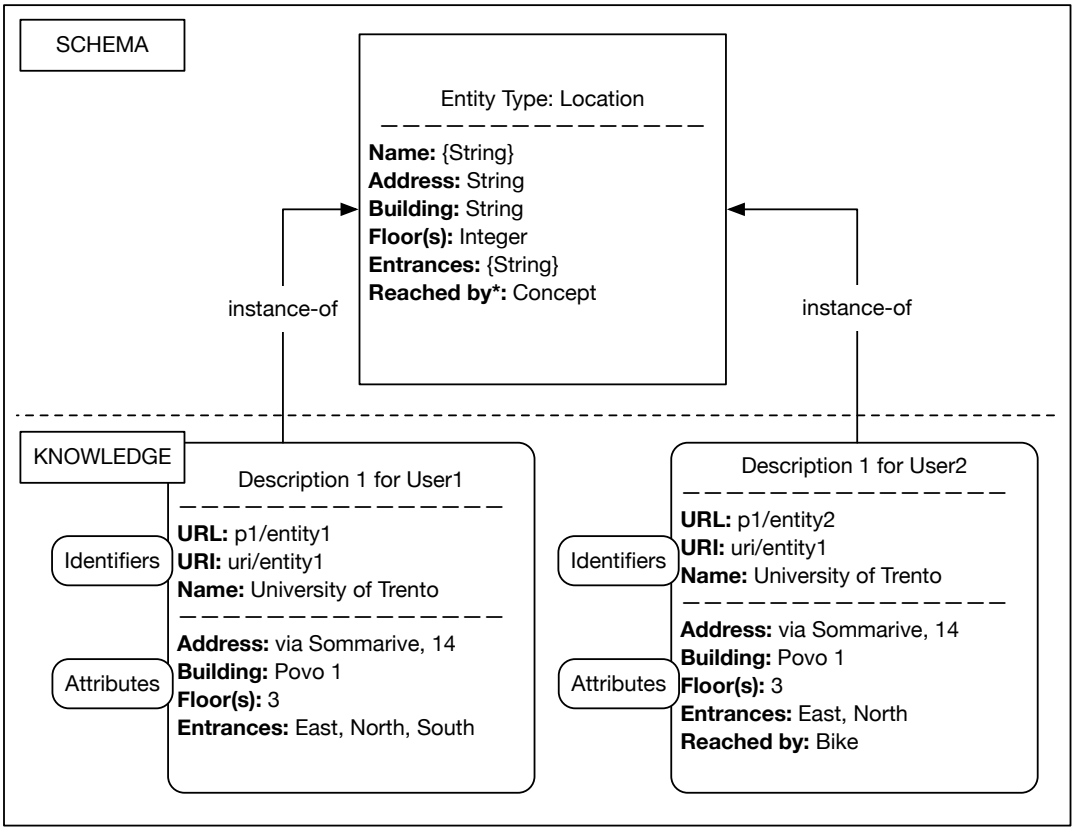


FIGURE 3.1: An example of an ET Location instantiated into two different representations (inter-difference).

The second example presented in Figure 3.2 on the other hand presents an example of instantiation with two different entities belonging to the same User1 (intra-difference). In this case the instances are the same and the only difference is their name. According to the context, the user can choose to use one with respect to the other.

For both the examples it is important to notice that the URI is always the same while the URL is different for every entity.

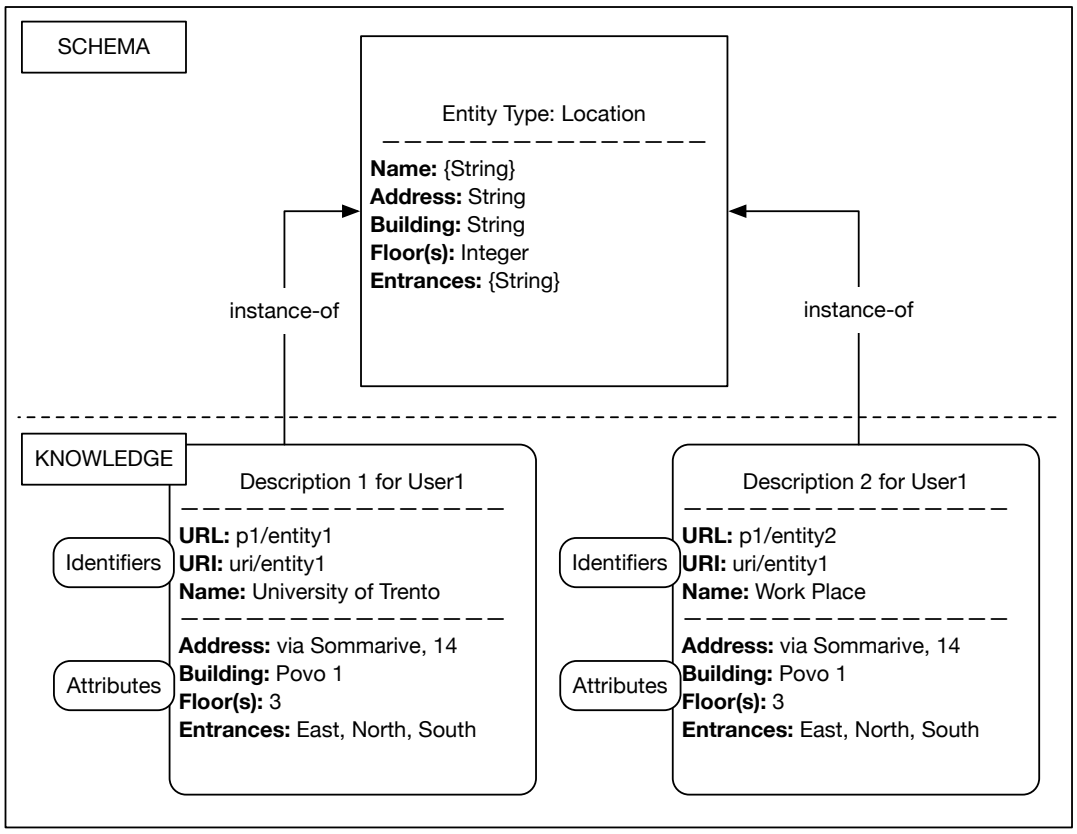


FIGURE 3.2: An example of an ET Location instantiated into two different representations (intra-difference).

3.4 Existing Systems for Knowledge Management

What presented above is the theory behind two systems developed by various members of the Knowdive[12] group in the past years, namely the KB and the EB. They are used for the various research projects of the members of the group and allow to generate, maintain and retrieve the knowledge elements of a model. For the sake of completeness we summarize in the following sections what they are, and how they work since we will be using them in the SB. We will use them to manage the contextual information about the user. **The EB in particular will contain a snapshot of the user context, the most recent one. All the other components of the system will be able to read a consistent version of this context and update it when required**.

3.5 Summary

This chapter introduced the main notions that are at the foundation of the work presented in this thesis which will be extensively used in the next chapters to build the SB. We presented our definitions of user, smartphone, sensor, context, service, entity and entity attribute. The *user* is the person using the SB who leverages on its services to improve her quality of life. In exchange of these free-of-charge services she provides her own data. The *smartphone* is the main (but not the only one) device that allows the user to interact with the SB in both directions. The *sensors* are those embedded in the smarpthone that allow to collect personal and environmental data.

[12] http://disi.unitn.it/~knowdive

The *context* is part of the user's personal view of the world, in a specific moment in time (a snapshot). This view is originated from everyone's habits and routines and is composed by all the elements (i.e., People, Locations, Objects, Events) that surround her and the relations among these elements. A *service* is the output of the SB after having analyzed the data from the user in a context-aware manner. The final goal of such a service is to improve the user's quality of life. An *entity* is a representation of any object that exists in the world, it has a type and is described by a set of *entity attributes* that are the key elements of our methodology and system. Moreover, we defined the entity-centric approach as the main representation formalism.

We then presented the schema as the reference structure for all the knowledge represented in the SB. We saw that there are ET composed by a concept and an AD. To generate knowledge out of this schema, it must be instantiated to create real world entities. The same user (intra-difference) or different users (inter-difference) can create multiple personal representations of the same real world entity in order to accommodate all the possible contexts. This notion will be the key for our methodology. We used the definitions of SURL and SURI as defined in [Fernandez and Ignacio, 2012] as a mean of identification of the entities, in particular those who are personal representations of the same real world entity. Each E is composed by a set of A that belong to a AD that characterize it and that will be the key element used by the methodology and system presented in this thesis.

Finally, since the knowledge modelling aspects are outside the scope of this thesis, we summarized two existing systems developed by the Knowdive[13] group which are meant to store the knowledge and the entities. The SB will leverage on these two components to manage the contextual information needed to generate knowledge that is meaningful for the user from the sensor data collected from her devices.

[13] http://disi.unitn.it/~knowdive

Part II

The Proposed Methodology

Chapter 4

Data Acquisition and Management

In this chapter we present the methodology developed in this thesis to tackle the data acquisition and management related issues when dealing with personal big data.

The first step consists in analyzing the data collection task and find the best solution considering the problem dimensions presented in Section 2.1.1. We start by defining some high level considerations that a data collection system must take into account when dealing with personal big data. We then present how the smartphone can be used in such a situation to collect high quality data, in the right quantity to produce meaningful results, without violating the privacy of the users. Every dimension of the process is analyzed: how the smartphone can be used to collect streams of sensor data, how this data can be efficiently synchronized for the analysis, how the collection task can be configured to be able to fast react to changes, how to engage the user in the data collection process designing a solution that is user friendly and finally, how to be transparent with the user so that he is aware of how her data is used.

The second part of the chapter describes how to manage the data collected by multiple users in the SB. Efficient solutions must be found to deal with such a huge amount of information and with such a variable work load. More in details, the analysis of the type of streams is made, with considerations also regarding the schema of the data without focusing on a single technology or use case. Lastly, considerations related to the users' privacy will be elicited.

4.1 Data Collection

The first element of the problem the SB is trying to solve consists in finding the best and most efficient solutions to collect data from the users that will be then used for the generation of the knowledge. This task can be assimilated to Lifelogging and in particular can be referred to as *Personal Big Data* [Gurrin, Smeaton, and Doherty, 2014]. Due to the characteristics of such data it is difficult to deal with them, collect, store and use them.

4.1.1 Data Collection Characteristics

The task of collecting personal big data presents additional challenges with respect to a traditional data acquisition task since it requires to interact with the user. The human factor must be taken into account with its psychological and sociological implications. We start the discussion of the methodology by defining the key characteristics the data has to satisfy: its volume, its quality and privacy related issues.

4.1.1.1 Data Volume

Unfortunately it is hard to define how much data is required for solving the semantic gap problem. The problem is twofold: there is the risk of having *not enough data* since the user can decide not to collaborate but at the same time there is also the risk of generating *so many data that we cannot even manage them*. This last aspect is particularly relevant when dealing with personal big data. In fact, as the standard definitions based on the 3 Vs [Laney, 2001] claims, big data are huge in terms of *volume* since they are generated very fast (*velocity*) and they are *various* in terms of data types. A solution that finds a good balance between the two is necessary.

When dealing with pervasive technologies such as in this work, these problems are even more difficult to tackle. The smartphone can easily generate data volumes that fill up the internal memory very quickly (within days) and the generated data are various. But collecting data from the users is not as easy as collecting data from other sources e.g., sensor networks. It is not enough do deploy the devices and wait for the data to come. The reason for this is that when dealing with the people one must take into consideration all the **psychological and social implications** of such an operation. Having a perfect smartphone application dedicated to the collection of the data is not sufficient. In fact, the user must collaborate, she needs to install this application and use it constantly in the next days. This requires the data scientist to find proper solutions in **engaging the user**. One possible solution is to incentivize the user in different ways: by providing useful services free of charge, creating a gamification mechanism, proposing a money compensation, among others. Unfortunately, it is not granted that this solution always works. In fact, if the requirements at the data collection side are too strict, the user feels like the incentives are not enough. To mention an easy example, if the data collection task impacts the energy consumption of the device resulting in only 3 hours of battery life, most likely the user won't accept any incentive; since the smartphone is used for purposes other than the data collection, this task cannot affect its normal usage. Usually a trade-off must be found between the two sides: on one side there is the need to have a lot of data and on the other there is the need of the user that does not want to be bothered and does not want her habits to be modified. In most of the situations the user wins and the data scientist must define less strict constraints. It can be the case that the new ones do not produce data good enough, but the alternative was to don't have data at all.

In this work the context is defined as a particular situation the user faces and believes is important. During a day we face many different situations that are important to us. Starting from this consideration we can define a requirement for the data collection task, that is, the system must collect data at least *on a daily basis*. As we will see this is not enough and a real time data collection scenario is foreseeable. Additional considerations must be done about how many data is needed to process each sensor stream, but we will deal with this in Section 4.1.2.

4.1.1.2 Data Quality

While methods to ensure the right volume of collected data vary by discipline, the goal for all of them is to capture quality evidence. A dataset is considered to be high quality if it "fits for [its] intended uses in operations, decision making and planning." [Redman, 2008]. This definition is broad and does not define the characteristics (also called quality metrics) a dataset must have to be considered high quality. Usually

these metrics are defined ad hoc to solve specific problems [Huang, Lee, and Wang, 1998; Laudon, 1986].

For the sensor data collected from the user's personal device we can say that if the algorithms are able to produce accurate results from them, then the quality or the original data is high. The quality metrics that affect the analysis in the context of this work are are:

- **The sensor sampling rate.** Is a parameter typical of the sensor that usually can be configured that represents how frequently one measurement is generated. Is expressed in Hz or values per second. This metric is related with the data quality in the following way: the more values we collect about a specific phenomena, the more details we have about it and the less approximations we do. Many research works go in this direction: [Lau and David, 2010] states that with $20Hz$ and $10Hz$ they are able to obtain an accuracy of up to $99,27\%$ and 95.42%, respectively. Their results are based on the task of recognizing the movements from accelerometer data. [Yamansavaşçılar and Güvensan, 2016] similarly find that with a sample rate of $20Hz$ and $5Hz$ they have an accuracy of $> 90\%$ and $> 95\%$ respectively. This metric of course is directly related with the data volume as well.

- **Sensor fusion.** The second aspect to take into consideration for data quality is the presence of multiple sensor streams to be used for the analysis. This is otherwise called sensor fusion, when multiple sensors are analyzed together to improve the final recognition accuracy [Khan et al., 2014; Johnson and Trivedi, 2011] with respect to a solution that analyses a single stream at the time. The SB allows to analyze multiple sensor streams to produce a single output, as explained in Section 5.3.

4.1.2 Data Collection From the Smartphone

The data collection in the SB has been designed so that to allow the usage of different sources of streaming data. In this work, usually we refer to the smartphone as the main source but it is important to mention that, even if it is the best one for the scope, it is not the only one and the system can accept data from any sources. However, according to our findings, the smartphone is the device that is always with the user and consequently allows to collect truthful data. We developed a solution [Zeni, Zaihrayeu, and Giunchiglia, 2014] the SB is currently using which consists in a mobile application called i-Log that takes into account all the before mentioned dimensions, i.e., Volume and Quality of the data. Multiple works in the research community have focused on Lifelogging solutions using the user's smartphone [Zhou and Gurrin, 2012; Shah et al., 2012].

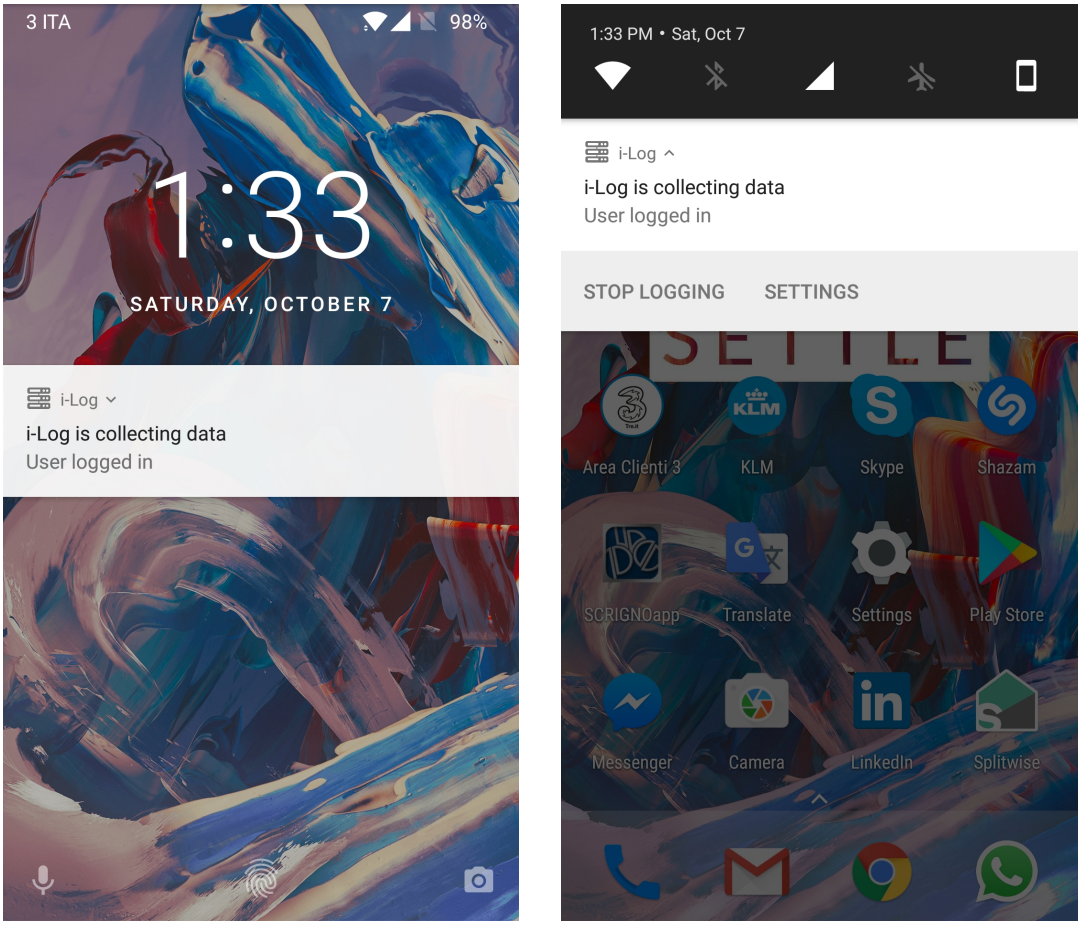
i-Log is a modular application for mobile devices that allows to simultaneously collect sensor streaming data from all the sensor embedded in the device, taking into consideration all the dimensions of the problem as explained in Section 4.1. The main key points in describing i-Log and its functionalities are presented in the next sections.

4.1.2.1 Unobtrusiveness and Transparency

One of the advantages of using the smartphone to collect personal data is that it can collect data for extended periods of time [Eagle and Pentland, 2006] and most importantly it can collect truthful data. To better explain this statement we have to

make a comparison with the other means that can be used to collect personal data, e.g., using dedicated sensing devices. In the research community, especially in the past, multiple works used these devices. They are devices that are designed specifically for collecting data. This means that the sensing hardware embedded in them has better characteristics with respect the one embedded in today's smartphones. On the other hand it has been proved that when the user uses these devices, she is aware of their presence and alters her routines only because she knows she is monitored. Using a smartphone on the other hand this effect doesn't happen because the logging process is unobtrusive to the user.

The aspects we took into consideration in designing i-Log that are related to unobtrusiveness and transparency can be summarized as follows:



(A) Notification while the screen is locked.

(B) Notification in the notification drawer.

FIGURE 4.1: i-Log notifications used to inform the user about the logging process.

Informed Logging. To be as transparent as possible with the users and to comply to the European Union (EU) regulations in terms of privacy and ethics, in i-Log we have to constantly inform the user about the logging process. This means that when the application is running and is actually collecting data, the user needs to be informed in some way. We analyzed different solutions, like periodic notifications with sound, but in the end the best solution we found is to use a visual clue. Technically speaking this corresponds with a notification always present on the user

FIGURE 4.2: How the i-Log notification is showed in the notification area.

screen. As notifications are implemented in Android[1] they are visible when *the screen is locked* and when the user *opens the notification drawer*. On the other hand, when the user uses the smartphone normally, the only way to see a notification is from the notification area at the top of the screen that shows a small icon for every notification. This perfectly balances the need to inform the user but at the same time to be as unobtrusive as possible.

The way we implemented the i-Log notification system can be seen in Figures 4.1a, 4.1b and 4.2. The first one shows how the notification appears when the screen is locked, the second one when the notification drawer is open and the last one shows the icon in the notification area.

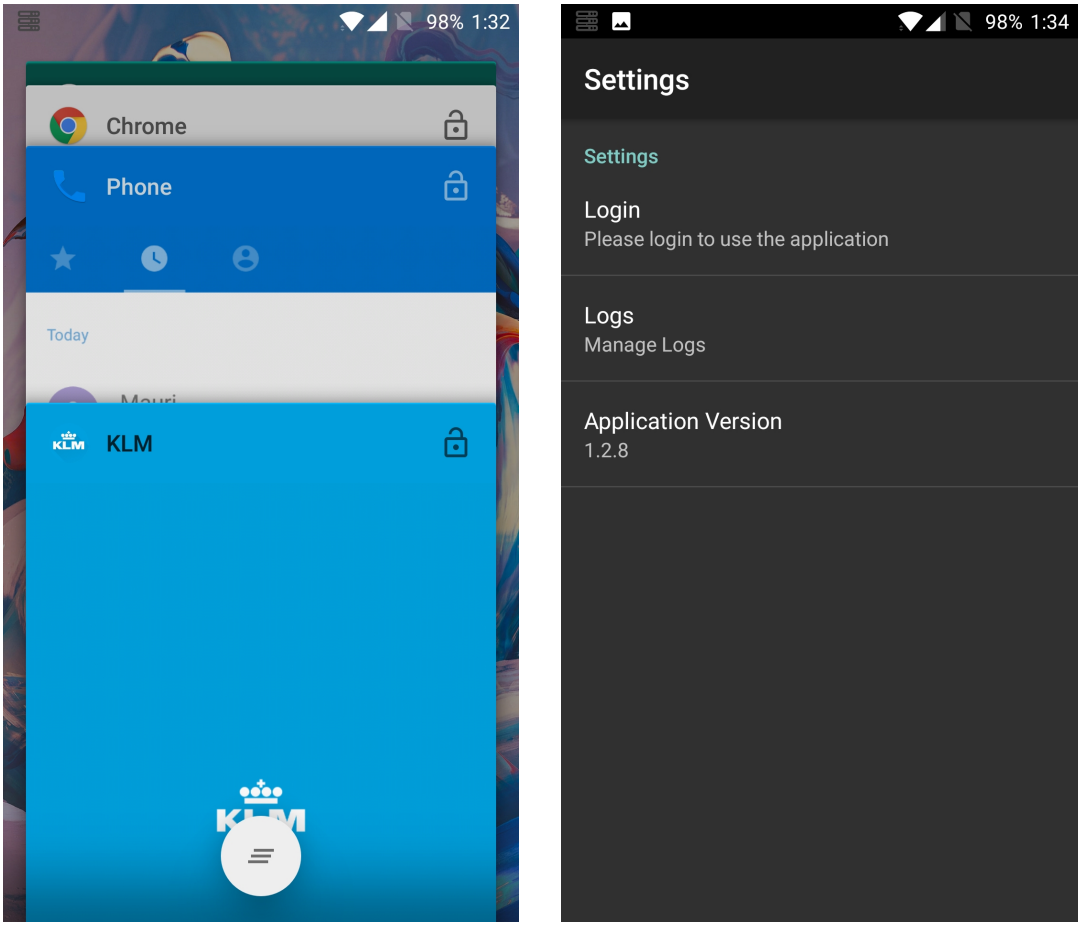
Minimal User Interface. As can be seen from the screenshots above, we kept the user interface essential. At the beginning i-Log was created as a tool for the data scientist to collect the data that later he will analyze. For this reason the user interface was composed by multiple screens that allow to control the logging process. For example it was possible to see all the collected sensor values or one could enable or disable the collection from single sensors.

Unfortunately, when we changed the target audience from the data scientist to the normal user, we discovered that these features were only creating a lot of confusion. In fact, the user not being used to the technicalities, was disoriented and some time also scared. We needed to completely re-design the user interface, removing all the views in order to simplify it as much as possible. As it if designed today, the user interface is composed by only the notification as explained before and a small setting screen. We also removed the application from the "recent application" screen (Figure 4.3a) because our experiments showed the user was annoyed by it.

The users can now interact with i-Log only via: the notification where she can stop the logging process and open the settings screen (Figure 4.3b). The settings screen is very simple, the user can only login and see how many logs are still present on the phone before the synchronization. Finally, the user can click the application icon in the Android launcher to start the application and the logging process.

Battery Efficiency. The evolution in the smartphone industry is so disruptive that every year important breakthroughs are made in regard to the computational power. Unfortunately there is still a huge limit in today's devices — the battery. The technology behind modern lithium batteries powering these devices has not

[1] https://developer.android.com/guide/topics/ui/notifiers/notifications.html



(A) i-Log is no longer present in the recently used applications list.

(B) i-Log settings window.

FIGURE 4.3: i-Log minimal user interface.

changed significantly since their first introduction decades ago[2]. A mitigation to the phenomenon (but not a solution) is given by the increase in the devices size which results in an increase in the battery size, allowing for more energy storage capacity. Additionally we start seeing fast charging solution that allow to completely charge the phone in less than one hour.

Due to this important limitation a smartphone can rarely last more than a whole day with normal usage patterns[3]. This phenomenon is even worse when the phone is used for tasks other than messaging, calling and browsing, e.g., for collecting data from the internal sensors, when the battery can barely last for few hours. In fact, as several works in the research community pointed out [Lee et al., 2012; Micallef et al., 2015; Carroll and Heiser, 2010; Zhang et al., 2010; Viredaz, Brakmo, and Hamburgen, 2003; Perrucci, Fitzek, and Widmer, 2011; Koenig, Memon, and David, 2013], there are sensors inside the smartphone that consume a huge amount of energy. The battery life problem in modern smartphone is today considered their major limitation with apparently no short-term commercial widespread solutions. In a scenario where an application is using real time sensor data to infer the user context the battery problem is crucial. A workaround to this issue is to have a clear knowledge

[2] https://goo.gl/fsPU7y
[3] https://goo.gl/hClNVH

of each sensor discharging behavior so that the application developers can take the appropriate countermeasures to mitigate the battery draining issue.

This aspect of the battery life is strictly related with the user experience and in the end can be assimilated with an obtrusive factor. Having a device that needs to be charged every 5 hours makes impossible to use it for the purpose of collecting data. The user will most likely uninstall the application after few days because, the application will prevent her from using the smartphone normally.

In i-Log we addressed this issue, finding a good solution that balances the needs of the data scientist and the user. We analyzed each sensor consumption rate under different conditions, the collection frequency and the combination of multiple sensors together. We discovered that the collection rate affects the battery consumption (as well as the data quality as explained above) and we found a balance between the two needs. The sensors that consume more energy are the radio ones, WiFi, Bluetooth and GPS. But the GPS can consume up to $3138,99$ times the energy consumed by the phone in idle state. To have a comparison for this value, the digital compass consumes $6,85$ times the energy in idle state. We then decided to adapt the collection frequencies rate, reducing the ones for the sensors that use more energy, as will be explained below.

4.1.2.2 User Informed Consent

The Directive 95/46/EC GDPR[4] imposes new strict rules on how the user data must be collected, stored and processed. An important element regards the fact that the user must explicitly give her consent. In the SB we do this in two different ways: by asking for the standard written consent and an additional consent directly on the device.

Written Consent. Usually when there are privacy related issue the user is asked to sign a piece of paper that explains under which conditions her data will be used and shared. Since this is the actual regulation we complained to this form as well. The students were asked to sign two different documents:

- The first one was a standard privacy policy statement
- The second one was a document explaining what the experiment wanted to achieve and which data we were collecting from them

Additionally the participants were asked to participate in a workshop in which the experiment was explained to them

On the Smartphone. Since the very first versions of the mobile operating systems, both iOS[5] and Android[6] there has always been a form of control over what the applications can do (or otherwise what they are forbidden to do). Historically iOS has always been the more strict with this regard but Android is adapting to this trend as well. Referring to Android, there are limitations about the data usage, the background execution, the hardware usage, the applications interaction, among others. To overcome these limitations the application developer is asked to specify which *permissions*[7] her application requires. *"To maintain security for the system and*

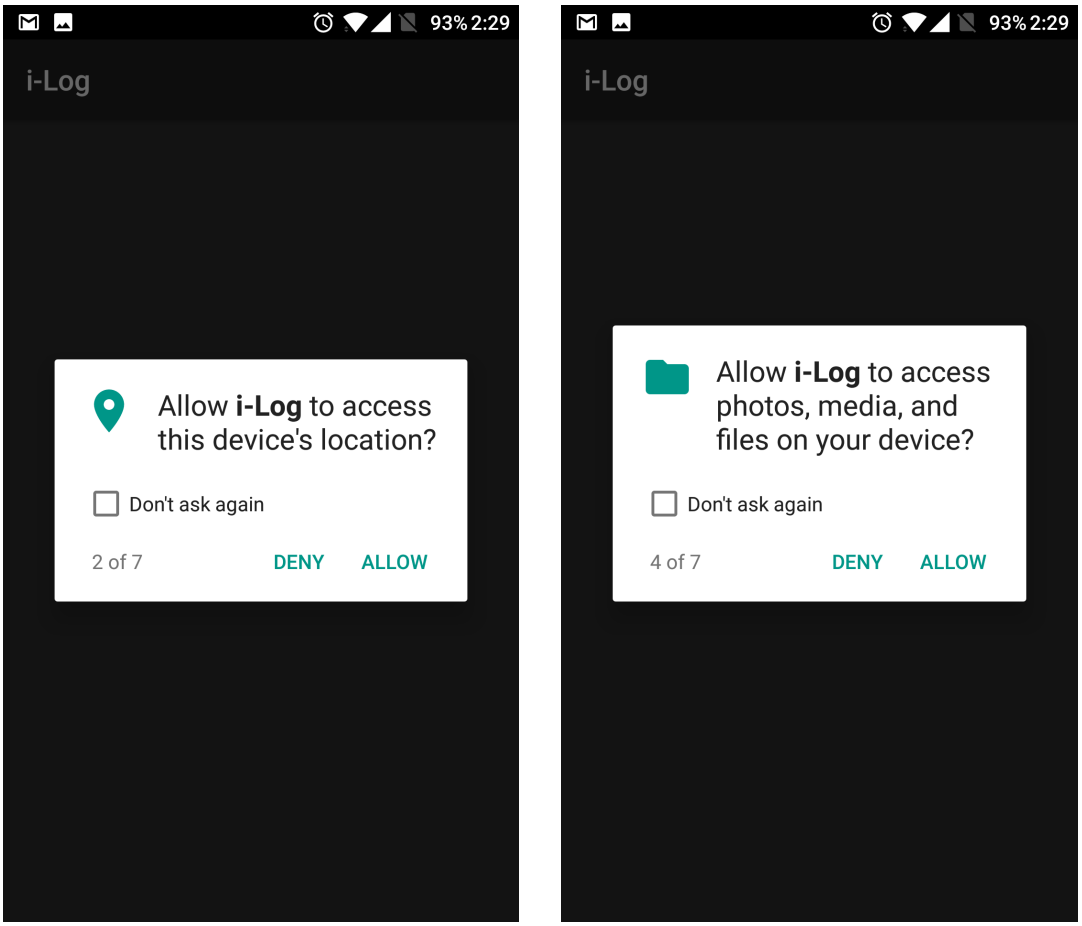
[4] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC

[5] https://www.apple.com/it/ios/ios-11/

[6] https://www.android.com/

[7] https://developer.android.com/guide/topics/permissions/index.html

users, Android requires apps to request permission before the apps can use certain system data and features. Depending on how sensitive the area is, the system may grant the permission automatically, or it may ask the user to approve the request".



(A) Permission request for device location.

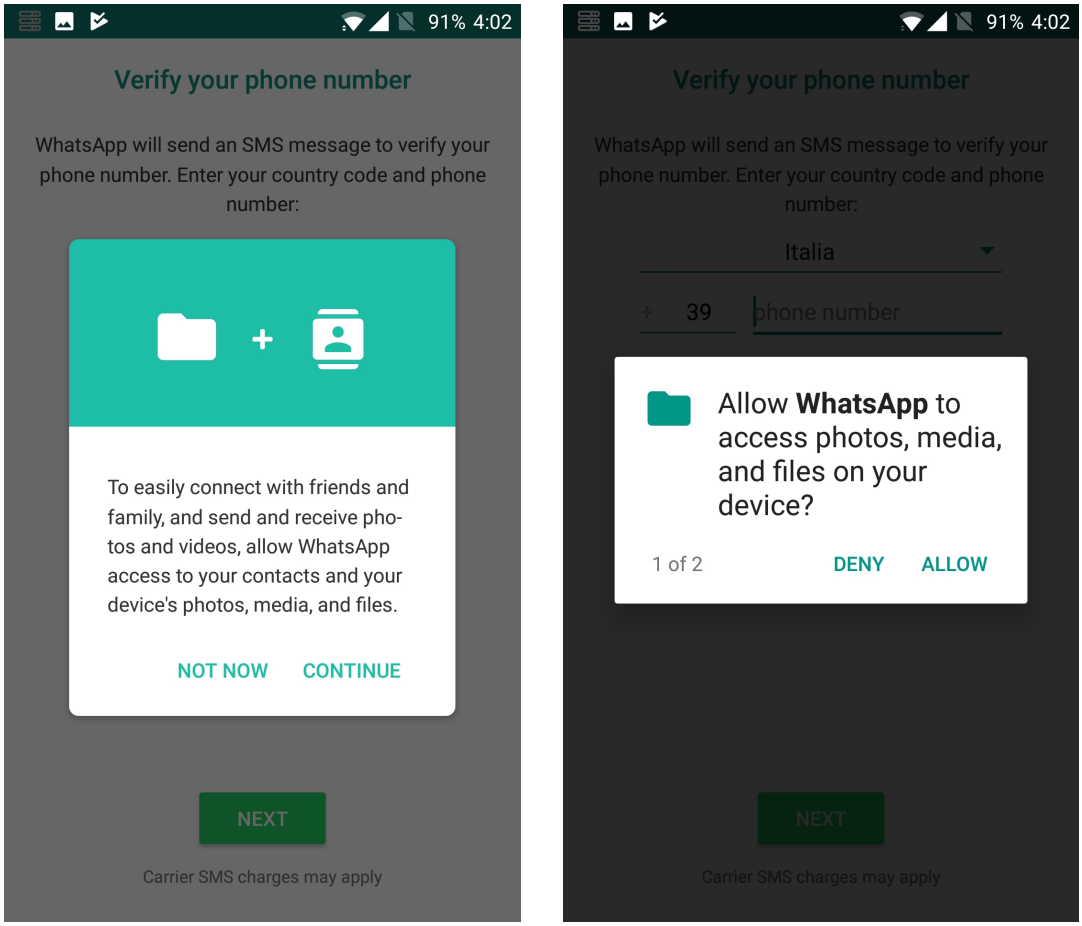
(B) Permission request for storing data on the device.

FIGURE 4.4: How Android asks the user to approve a permission request.

This last element of asking the user to approve the request is called *at run time permission request*[8] that Android introduced from Application Programming Interface (API) level 23 (Android 6.0 Marshmallow) of the operating system. Before it all the permissions were automatically granted during the installation phase: the user was presented with a list of permissions and by pressing install she was automatically accepting all of them. Now things are different and the user is asked to approve the request when the corresponding feature is used for the first time. Unfortunately, the request for approval are designed in an unfriendly way as can be seen from Figures 4.4a and 4.4b and there is no way to customize it at the moment. The most important issue with them is that they lack of details, and are not precise in describing what is requested. Consider for example Figure 4.4b, we see that the application asks to "Allow i-Log to access photos, media, and files on your device?". From this message it seems that i-Log will read the files on the device but instead

[8] https://developer.android.com/training/permissions/requesting.html

it is only trying to store its own logs containing the the collected sensor data. This procedure usually ends up in the user denying the permissions.



(A) Screen designed by Whatsapp to help the user in approving the next permission request.

(B) Permission request for storing data on the device.

FIGURE 4.5: How Whatsapp asks the user to approve a permission request.

We took this new requirement as an opportunity to further inform the user about the data she is sharing with the SB that will be used to provide her the services. We then decided to design a procedure that guides the user in the approval of the requests. It consists in showing a user-friendly screen for every permission with additional information just before the user is asked to approve it. This is similar to what other famous applications do, like Whatsapp[9] as showed in Figures 4.5a and 4.5b.

In i-Log the system requires the user to approve 10 different permissions: *to access the contacts, to access the device's location, to record audio, to access photos, media, and files on the device, to Short Messaging System (SMS), to Phone, to take pictures and record video, notification access, usage access* and finally *to run the application in background without limitations*. The only **mandatory permission** is the last one that allows to run i-Log in background, all the others are optional (but as we said before, one of

[9] https://www.whatsapp.com

the metrics to establish the quality of the data is related to the number of sensors collected. The more the sensors, the easier to infer the user context). The background limitation was introduced by Android in the last versions of the operating system to preserve the battery life. Since it has no mean of understanding how many energy an application is demanding from the battery, it superficially blocks all the applications in the background after some time if they are not whitelisted. As we said in Section 4.1.2.1 we already tackled the battery consumption issue and then we require the user to whitelist i-Log.

To ask the user to approve all these permission requests, we designed a procedure that alternates a description screen to the one that actually asks the approval, as Whatsapp does. To design these screens we followed some simple rules:

- Be transparent and honest with the user;
- Put the logos of the institution(s) the user knows: the University of Trento (and in one of the use cases also the Municipality of Trento). Seeing a familiar and official logo helps the user in approving the permissions;
- The screens were kept simple but nice to see;
- The jargon was kept colloquial to facilitate the understanding;
- Present a link to a webpage that shows more exhaustive explanation of what the permissions are and all the privacy related aspects.

4.1.2.3 Multiple Sensor Streams

Today's high end smartphones are so powerful that their hardware at some extent can be compared to the one of an average laptop, both in terms of processing power and memory. Moreover, they are often equipped with internal hardware sensors used to improve the user experience. These two elements, and the fact that it is fairly easy and inexpensive to develop ad hoc applications that can run on them, makes them the ideal device for collecting data about the user.

i-Log [Zeni, Zaihrayeu, and Giunchiglia, 2014] is able to collect multiple sensor streams from the user's smartphone and attached wearables. Every device has a different set of sensors, with different characteristics both in terms of sampling frequency but also considering their accuracy in sensing the world. All these aspects have been taken into account in designing i-Log which is able to adapt to any Android device. If a sensor is present and has been configured as enabled, the application starts collecting data from it.

But the internal hardware sensors are not the only ones that can collect data about the user. There are events generated by other components of the smartphone that if collected will be useful to the analysis. We created dedicated software components in i-Log to detect such events and collect them. We call these components *software sensors*. A nice characteristic of these sensors is that they do not affect the battery life since they do not use an hardware component on the phone. There are two main types of software sensors:

- **System Broadcast Events.** The Android operating system is designed to send broadcast messages[10] of different type and nature to the applications running on top of it. The application only needs to subscribe to the updates of these events. Example of such broadcasts are the WiFi connection/disconnection

[10] https://developer.android.com/guide/components/broadcasts.html

events, a new notification appearance/disappearance, an incoming phone call, among others. We developed single software components that can register or each of these events and listen when they are trigger. When this happens, i-Log logs the event. Because of the nature of these broadcast, there is no way to control the collection frequency but they are collected when their status *changes*.

- **Poll Events.** Some of the software sensor require to trigger the collection process instead to waiting for a change. Examples of these sensors are the one that logs the running application in foreground on the smartphone or the one that records the audio, among others. For them, we need to create a scheduler that periodically collects the value and logs it. This is done using a timer and the duration of this timer can be configured. We can assimilate this time interval to the collection frequency of the hardware sensors.

A list of all the sensors i-Log can collect data from, both hardware and software is presented in Appendix A.

Except for the System Broadcast Events Software sensors, that as explained are generated when an event is detected, all the other sensors, both hardware and software can be configured to collect data at a specific **collection frequency**. As said in Section 4.1.1.2, the collection frequency is one of the two metrics used to determine the quality of the collected data. The higher the frequency, the higher the quality. Unfortunately the frequency also affects the battery life (as explained in Section 12.1.2) and the amount of data generated, that in the case of personal big data can be easily become unmanageable. Specifically, the higher the frequency the lower the battery life and the higher the amount of data. The right strategy is to find a trade off to choose the best collection frequency for every single sensor. Table A.1 in Appendix A presents for every sensor also details about the amount of collected data and of battery consumed. The former can be High (H) or Low (L) while the latter can be H, Medium (M), L or None (N).

The i-Log application has been kept as simple as possible to reduce the battery consumption. It only collects, stores and synchronizes the data. But if needed it can also perform **smart sensing strategies** that can be designed to be context dependent. This means that a sensor can used with respect to another if certain conditions apply. The only contextual sensing feature that is actually implemented in i-Log regards the collection of location data. The standard way on a smartphone to know the position of the user is to use the GPS sensor. It is very accurate, up to 3 meters, but unfortunately has some drawbacks: it consumes a lot of energy and it works only outdoor. In an indoor situation usually the WiFi network is used to understand the user position. For this reason, it is counter-productive to keep the GPS on when inside a building: it won't produce any data and on the other hand will drain the battery very quickly. For this reason we created a smart sensing strategy that consists in disabling the GPS sensor when connected to a WiFi network and enabling it back when the connection with the WiFi terminates.

Other strategies can be easily implemented thanks to the modularity the application architecture has.

4.1.2.4 Storage

i-Log stores the collected data from the sensors as log files, formatted according to the Comma Separated Value (CSV) format. The reason for this choice is that this is the easiest and fastest way to store the data, and this corresponds to savings in terms

of energy consumption. We also evaluated to use an internal SQLite database, but it required too many Central Processing Unit (CPU) resources to insert and extract the data. Every time a new reading is generated, it is added to a First In First Out (FIFO) in memory. When the FIFO is full, it is flushed to file. In this way we reduce disk access and then the battery consumption of the whole procedure, with respect to creating an empty file and appending the content to it for every received value.

Depending on the sampling frequency and on how many sensors the smartphone collects data from, the amount of generated data varies. Our tests show that one high-end smartphone, collecting data from all the 3X sensors, can generate between 300MB and 2GB of data per day. Considering that i-Log has been designed to collect data from the user's smartphone for multiple consequent days, a good strategy for storing these data must be found. As we will present in the next section, the data needs to be synchronized with the server for the analysis but it can happen that, in some situations, the user cannot find a WiFi connection for more days. Considering the data generation rate a phone will run out of memory quickly. For this reason we implemented a feature that allows to compress the i-Log logs as they are generated. Actually they are written to disk directly in the bzip2 format [Seward, 1998]. Even if it is considered slower than others we decided to use it for its high compression rate. In fact, for the small log files we have the time required to compress them is slow.

4.1.2.5 Synchronization

In the SB all the analysis is done in the backend system while the smartphone is used only to collect the data and to provide the services generated from the results of the analysis to the user. This shows the necessity of synchronizing such collected data with the server at least on a daily basis, ideally multiple times per day. The main reason for this is to meet the desired responsiveness for the service. As we illustrated in Section 3.1 the context changes multiple times per day. Then, for a context-aware application to work it needs to adapt to these changes when they happen. Having said this, there is the need to design a synchronization mechanism for the collected data.

Data synchronization can happen in two ways: (i) with a wired or (ii) wireless connection. Following the idea that the collection process must be as unobtrusive and transparent as possible for the user, the wired solution cannot be applied since we do not want to ask the user to plug the cable in his smartphone. The other possibility is to use a wireless communication channel. Today's smarpthones are embedded with multiple wireless communication capabilities: Bluetooth[11], WiFi[12], 3rd Generation (3G)/Long Term Evolution (LTE) radios. All of them have very specific characteristics that make it hard to select the best solution a priori. Depending on the use case one should be selected with respect to the other. What follows is a summary of what are the main characteristics of each of them in order to explain the choice made in our application:

- **Bluetooth:** This connection type is very diffused in mobile devices such as a smartphone and is usually dedicated to short range, slow speed and small size file transfers. The energy consumption is low in the latest versions.

[11] https://www.bluetooth.com/
[12] https://www.wi-fi.org/

- **WiFi:** This connection is the standard for wireless communications, it allows to move huge files at high speed in mid-range situations with a moderate energy usage.
- **3G/LTE:** Very diffused in smarpthones is the main long-range wireless communication methodology that allows to reach high speeds comparable to the one of a WiFi connection. From an energy perspective point of view this is the one that requires more energy, in particular in those situations where the signal strength is poor.

For the synchronization task we evaluated all of them ending up in choosing the **WiFi** as the final wireless communication standard. We discarded the Bluetooth since its short range couldn't enable the synchronization with the server. The best candidate would have been the 3G/LTE since he had all the necessary characteristics but unfortunately the limited amount of data that could be sent and received constituted an important bottleneck. Standard data plans are limited to few GBs per month, between 2GB and 5GB (in Italy). Having the necessity to synchronize up to 300MB on a daily basis forced us to discard this solution and choose instead the WiFi.

The WiFi has some limitations as well, in particular the mid-range connection capability. To be able to synchronize the data the device needs to be connected to a WiFi network. Fortunately, today there are multiple freely available WiFi hotspot scattered around our cities. The application has been designed to *opportunistically exploit* this connection opportunities. This means that, whenever a connection event is detected, the application starts synchronizing the available data. If the connection window is too short to synchronize all the data, then some of them is sent and the remaining one is shared at the next opportunity.

With this approach we are able to synchronize the data multiple times per day meeting the requirement defined above.

4.1.2.6 Configurability

The user interaction is a critical aspect when designing mobile applications. Requesting the user to perform an operation is always considered dangerous for multiple reasons. First of all, one must understand when is the right moment to ask the user to do such an operation. In fact, asking in the wrong moment can imply loosing the user. Secondly, even if the user is willing to contribute, it is not obvious that he knows how to do it. For all these reasons and for the sake of simplicity, we decided to leave to the user the possibility to set only few configuration parameters directly inside the application while we implemented a functionality that allows to remotely configure the application. This functionality allows us to improve the collection process (and ultimately the user experience) without involving the user directly. The parameters that can be configured are the one the can affect the quality of the data, i.e., the *sensor sampling rate* and the *simultaneous collection from multiple sensors* as reported in Section 2.1.1. Remotely we can enable or disable the collection from single sensors inside the device and additionally we can change the individual rate at which these sensors collect the data.

To implement this functionality we leveraged on an existing service provided by Google to its developers called Firebase[13]. It provides a simplified way to access the users' devices by sending them messages. Creating the appropriate messages and

[13] https://firebase.google.com/

creating a dedicated module in the mobile application that receives these messages we were able to exploit this solution to remotely configure the data collection. The reasons why we chose this solutions are the following:

- **Security:** all the communications take place through the Google servers. We developed a specific server side module that sends request to the Google server and then they deliver the message to the devices.
- **Energy efficient:** using the Google Firebase service we are able to obtain an higher energy efficiency in delivering these messages. The reason for this is that Google collects messages from multiple applications and uses a single short window to deliver all of them, waking up the phone only one time. Consider the phone on the desk, with the screen off. In this situation most likely the phone is in a sleep state if the user didn't use it in the last minutes. By sending it a message it will exit this state (if he can receive the message at all), resulting in an energy waste. On the other hand, Google Firebase collects some messages from different applications like Email, Facebook, Whatsapp and then finds the best moment to deliver all of them in a single moment.
- **Semi real time delivery:** the considerations done in the previous point indicate that the delivery of the message, i.e., the configuration command, can be delayed by Google. This is not a problem in this situation since there is no need to apply the new configuration in real time.
- **Lack of connectivity:** the adopted solution helps also in the case in which the phone is not connected to the internet. In fact, Firebase is designed so that when the smartphone connects back, all the messages are delivered.

We developed a dedicated web application so that the data scientist was able to interact with the Google Firebase service and sens the configuration messages to the device. This configuration can be done by user or by topic. It is possible to create groups of users by making them subscribe to a specific topic. The strategies we followed for grouping the users were related to some of their characteristic that could affect the data collection: device, demographics (i.e., age, sex, occupation) and by the services they used.

4.2 Data Storage and Retrieval

A second very important aspect of the problem the SB is trying to solve consists in finding the best solution to efficiently store and later access the streaming information. The solution must be able to tackle the standard problems linked with the management of big data, namely their *volume*, *velocity* and *variety*.

Volume. We already tackled the problem of the size of personal big data when explaining how these data are collected from the smartphones. Since all the user send their data to a backend server for the analysis, this problem becomes even more relevant for the storage phase. In fact, thousands of users need to synchronize huge amounts of data on a daily basis. Of course, a server has much more computational power and storage capabilities with respect to a smartphone, but if we consider the amount of users in the system, the problem remains relevant. In fact, having an arbitrary big number of users generating up to 2GB of data per day, will quickly saturate the server resources. This corresponds to a situation in which we collect

everything from the users. In future implementations, once the system is stable enough and once the algorithms start getting more complex, we can decide to collect only the necessary data and reduce the requirements in this regard.

In one of our use-case we aim at reaching 1000 people that will use i-Log for few weeks. These users will generate 2TB of data per day that must be stored in a database system, using a schema, so that the retrieval will be possible and most importantly fast.

This last aspect is the key: there are no issues in just storing all the comma separated values generated from the smartphones in a standard filesystem, but at that point it will be nearly impossible to retrieve the data. On the other hand, since real time analysis must be done from these data, an **efficient storage system must be used**. Most importantly, the appropriate *schema* for the data must be used.

Velocity. The collected data come in huge quantities in short periods of time. This is due to different reasons: (i) the way big data are generated, (ii) the number of users in the system and finally (iii) the way the data are synchronized. This last aspect is the one that most affects the speed at which the data need to be stored into the internal database. In fact, as we said in Section 4.1.2.5 the data is synchronized following an opportunistic connection pattern. More in details, whenever the smartphone is connected to a WiFi network, it starts uploading the data. Some users are always connected to a WiFi router, both at work and also at home. For these people we do not see particular issues since basically the data will be synchronize as they are created. In the other hand, for some specific categories of people, this is not the case. They can have a WiFi connection only at home, or only at work. In both situations the phone stores the collected data in the internal memory and synchronizes them later when possible. These two situations show an issue that is related to the speed the data will need to be inserted into the database. In fact, it will be the case that a lot of people arrive at home/work at the same time, and all their smartphones will require to send the data at the same time. Then the system needs to handle these burst in the load and the database need dto support very fast inserts.

Variety. The data generated from the smartphones is modeled as time series. A time series can be defined as a sequence of data points that have a timestamp associated with every value. The data points can be of any kind, numbers, strings, complex structures and this is where the variety of the data comes from. Most of the values in the database will be float numbers that represent the real world, but we store also files (pictures, audio, etc), text and objects such as the Point for the location. The data type, in general, affects the queries. These arguments depend on the technology used to store the datam i.e., the database system. Since here we are referring to a solution that must be general, we don't go into further details and we will explain it in Section 11.2.

4.2.0.1 Types of Streaming Data

A stream of data is defined as a continuous flow of information that is made available over time. In contrast with non-streaming data, new values are always produced and must be added to the ones that have already been generated.

In the SB there are two types of steaming data which can be modelled using the same schema, but that have different characteristics:

- **Sensor streams.** The streams of sensor data collected from the user's smartphone. These are the personal big data that have a huge volume, are generated

at high speed and have different formats. For this type of streams the choice of the database technology will be critical since not all the database can handle them.

- **Attribute values streams.** Attribute values keep changing depending on the user context. This means that we can refer to them as streams. However, the way we store the context in the EB does not allow to store the older values. Being it a snapshot, is only memorizes the last values, overwriting the old one. However we believe these values are of key importance in the SB and decided to keep track of the older values too. They are stored in the streaming storage system with the sensor streams collected form the user devices as a stream of attribute values.

We created this distinction because even if they can be both referred to as streams, they have very different characteristics and this affects the way the are stored and used.

4.2.0.2 Schema considerations

In a database system the data schema usually refers to the structure according to which the data is stored. More in details, it refers to the organization of data as a blueprint of how the database is constructed. The data schema is associated with the database technology used since different databases require to build schema differently.

In this Section we want to give a general description of how the data must be structured in a scenario like the one the SB is trying to tackle. Moreover, we refer only to the **streaming data** since the defining the schema of knowledge data is outside the scope of this thesis. Specific considerations are technology dependent and are left to the final sections of this thesis, where an instantiation of the reference architecture that applies this methodology is presented.

There are different dimensions of the methodology with respect to how the data need to be stored:

Users' Data Separation. The first element to be considered about the data schema refers to the separation of the user data. For privacy reasons, the system must isolate the users' data so that one user can access only her own data. This separation can be obtained in two ways:

- **Physical separation:** we can allocate one database instance for every user. In this way the data is separated since one instance can only access his own data. The separation can happen on the same machine or even on different physical machines. This is the solution that guarantees the best isolation; unfortunately, having N database instances requires to allocate N times the resources of a single instance;

- **Logical separation:** the second solution to this aspect of the problem is to create a schema that allows to store the data by user so that one user can access only her own data. This solution allows to optimize the hardware requirements but the effect of the isolation is reduced.

From the point of view of the privacy both solutions are possible and is up to the technology define which one is the best. There is the need to balance the privacy

requirements with the technical requirements.

Schema for Time Series. In the context of this thesis we formalize a Streams (S) as a set of Elements (E_T)

$$S =< \{E_T\} > \tag{4.1}$$

where an Element (E_T) of a stream is defined as a tuple

$$E_T =< T, \{C\} > \tag{4.2}$$

where

- T is the time moment at which the Element (E_T) was generated. The timestamp is stored as a *long* value, using the *Unix time*[14] convention. It is defined as the number of milliseconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC) which corresponds to Thursday, 1 January 1970;
- *{C}* is a set of non-empty components. In the case of a sensor stream they are the components of the collected value. In the case of attribute values streams these are the attributes of an entity.

A Component (C) is then defined by a tuple

$$C =< N, V > \tag{4.3}$$

where

- N is a label representing the name of the component;
- V is the value of the component.

This schema is valid for both the **sensor streams** but also the **attribute values streams**. What follows is an example of the schema maps to both of them:

- **Sensor streams.** In this example we consider the stream of data that stores the information collected from the accelerometer sensor of the smartphone. The sensed measure is the acceleration, defined as the rate of change of velocity of an object with respect to time. It is the net result of any and all forces acting on the object, as described by Newton's Second Law. In the International System of Unit it is measured in $[m/s^2]$. The acceleration is a vector quantity, meaning that it has a *magnitude* and a *direction*. In fact, the accelerometer collects the acceleration of the device as three values among the different axes: X, Y and X. A body that is at rest is subjected to a total acceleration equal to the gravitational acceleration $g = 9,81m/s^2$, that, depending on its balance, should be split across the three axes. In a smartphone usually the reference system if the one showed in Figure 4.6. This means that, if a smartphone is on a desk, with the screen up, all the accelerations the smartphone is subjected to is the gravitational acceleration over the Z axes and in particular the measured value should be negative. Each of these values is mapped to one C in the schema above.

  Formally, the stream of accelerometer values is defined as:

$$S_A =< \{E_T^A\} > \tag{4.4}$$

[14] ISO 8601

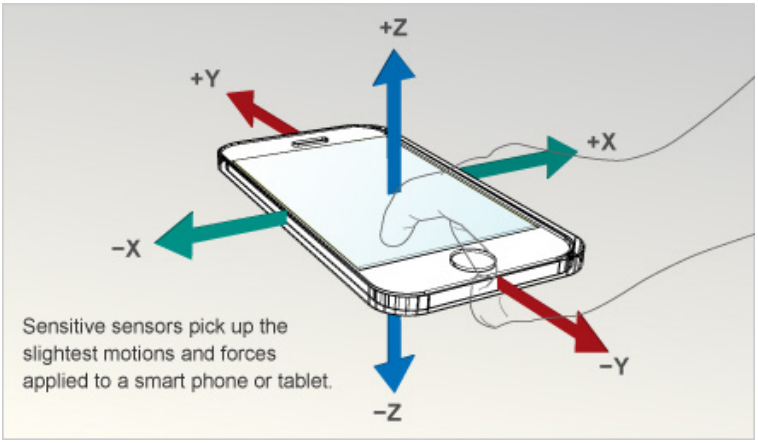


FIGURE 4.6: Representation of the reference system for the accelerometer sensor on a smartphone.

where an Element (E_T^A) of the stream is defined as a tuple

$$E_T^A =< 1507618087, \{C_A\} > \tag{4.5}$$

where

- *1507618087* is the time moment at which the Element (E_T^A) was collected on the device, this long number corresponds to Tue, 10 Oct 2017 06:48:07 GMT;
- $\{C_A\}$ is the set of components of the acceleration, X, Y and Z with their corresponding values:

$$\{C_A\} = \{< X, 0,001 >, < Y, 0,010 >, < Z, -9,81 >\} \tag{4.6}$$

- **Attribute values streams.** Consider the example of Section 5.3.1.1. The objective of that knowledge update task is to update the attribute speed of the Person by leveraging on both sensor data and contextual information referred to her car. Mapping on the above schema, we will have two streams, one for the entity Vehicle (e42b47a699d3a) and one for the entity Person (cf507j3fert16). The former has one attribute, *Speed* of type float, while the latter has two attributes, the *Speed* and the *UserPresence*, respectively of types float and boolean. Formally this situation can be described as:

$$S_{e42b47a} =< \{E_T^{e42b47a}\} > \tag{4.7}$$

$$S_{cf507j3} =< \{E_T^{cf507j3}\} > \tag{4.8}$$

where

$$E_T^{e42b47a} =< 1507621075, \{C_A^{e42b47a}\} > \tag{4.9}$$

and

$$E_T^{cf507j3} =< 1507631081, \{C_A^{cf507j3}\} > \tag{4.10}$$

Finally

$$\{C_A^{e42b47a}\} = \{< Speed, 8.2 >\} \tag{4.11}$$

and

$$\{C_A^{cf507j3}\} = \{< Speed, 8.2 >, < UserPresence, true >\} \tag{4.12}$$

4.2.0.3 Data Privacy

A key element a system must take into account when dealing with the data is the privacy and any ethical issue that can arise from their collection and exploitation. In the case of personal data this aspect is even more crucial.

Any European organization or institution that deals with personal data at any level has to comply with the European rules[15] about protection of personal data. This applies also to organizations based outside the European union that collect data about European citizens. The European Commission is in the process of proposing a completely new regulation that will start being active from May 2018, the Directive 95/46/EC called GDPR[16]. This regulation is much more strict with respect to the old one and has been designed to give individuals better control over their personal data held by organizations. The most salient elements introduced by it are:

- **Measures to protect the data**. Organizations need to implement appropriate measures to protect the users' data and avoid leaks;
- **Require the consent**. Organizations need consent from the user if they want to collect data from her;
- **Data access**. Users should always be able to access all their data, see it and decide to delete it totally or partially.

The system presented in this work tackles the privacy issue by anonymizing the collected data to store them and additionally gives the possibility to the user to have full control on her data. The anonymization consists in storing the user data and associate it to a unique identifier instead to the username. This unique identifier is created randomly by the machine when the user creates an account in the system. Specifically it is a **salt**, that in cryptography refers to a casual sequence of bits, usually used in combination with a password and an hash function. The salt is generated using a methodology based on the Secure Hash Algorithm 1 (SHA-1) which is a cryptography hash function designed by the United States National that generates a 40bytes string. This string is the one used to refer to the data in all the databases of the system. Of course, since the data is received from the users that are logged in with their accounts (with a standard username and password authentication mechanism) there is the need to link the unique identifier and the users' accounts. This is done through an encrypted disambiguation table that is small, and contains only one row per user, where the relation between the account and the unique identifier is make explicit.

[15] http://ec.europa.eu/justice/data-protection/

[16] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC

4.3 Summary

In this section we presented the methodology we developed to collect personal big data from the user. The data are at the core of the SB and then the data collection and management task is crucial.

We presented the different dimensions of the problem, i.e., collecting enough quality data to be used in the solution of the semantic gap problem. These data could be collected from different sources but we decided to focus our attention on the smartphone, since it is the best device to continuously collect truthful data from the user. We presented i-Log, the application we developed to collect huge amounts of data from the user in an unobtrusive and privacy aware way from multiple sensors inside the device. We described how the data is stored locally and synchronized with the server where the analysis occurs. We explained how the mobile application can be remotely configured to improve the data collection phase.

Finally, we explained how the streaming data should be stored inside a system that is able to deal with personal big data, also accounting for the privacy of the users.

Chapter 5

Knowledge Generation

In this Chapter we present the methodology behind the knowledge generation task performed by the SB, which is at the core of the problem we are solving within this thesis.

We start by precisely defining what is the context within this work from a technical point of view, starting from the formal definition given in Section 3.1.

We then discuss how the knowledge can be instantiated, automatically by the machine with the help of the user or manually by the user herself. Moreover, we present how the already instantiated knowledge can be updated using the streaming data collected from the user smartphone using previously designed algorithms that leverage both on the streaming data and on the status of other entity attribute values in the user context.

5.1 The Context as a Snapshot of the Personal World

In Section 3.1 we defined the context as any information that can be used to characterize a situation of the user that is considered relevant for the interactions with other entities in any specific moment. It can be seen as the personal view of the world of the user at a specific time T that is dynamic.

To enable highly personalized services we need to have access to a consistent snapshot of the user context, in real time. For this reason we decided to represent it following the entity-centric approach which describes the context as a series of entities, characterized by attributes and accounting also for the relations among them.

Among all the possible ET, we decided the fundamental ones to describe any possible user context are the Locations, the People, the Events and finally the Artifacts (objects). The definition of the schema of such Entity Types is outside the scope of this thesis and we leveraged on the work done by the other members of the Knowdive[1] group. In order to use this context as described above, the first step is to instantiate all the entities composing it, as described in Section 3.3. While the schema is defined inside the KB, the instantiated entities are stored into the EB, as explained in Section 3.4.

The EB system and its content is the core of the personal representation of the world of every user. From now on we will refer to this representation as **the knowledge of the user**. The ***context*** *is the part of this knowledge that is relevant in a specific moment, or in other words, those entities that are enabled among the one available in the user knowledge.*

Every component of the SB has access to the contextual information of the user from the EB, and all of them have a consistent view of it. At the same time, at the

[1] http://disi.unitn.it/~knowdive

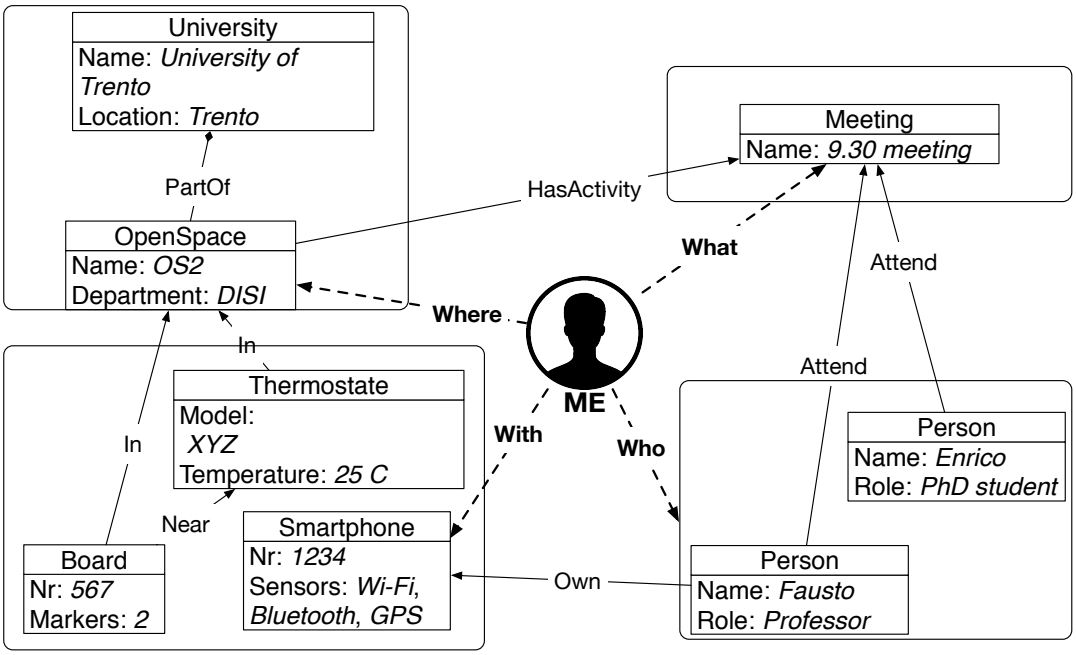


FIGURE 5.1: The user context represented with the entity-centric approach.

moment when the context varies, the dedicated component of the system can modify the corresponding information in the EB so that everyone will be aware of this change. An example of how the entity centric approach represents the context of the user and ultimately how it is stored into the EB system is shown in Figure 5.1. We see the user at the core, with four sample instantiated entities, one per type, that are part of her context.

There are two possible ways of generating user knowledge:

- **Knowledge instantiation**. This refers to the ability of *adding* or *removing* entity instances from the user knowledge database. Since this represents the personal representation the user has of her own world, the contribution of the user is important and required. At the beginning when the EB is empty, the user must create her own instances, for example defining places or people that are important. After a while the machine can start analyzing the available data and consequently help the user in the instantiation phase. An example of this can be a Location that is frequently visited but that is not saved as an important location in the EB. The machine, starting from the GPS coordinates, can detect this location and since it cannot find a match in the user knowledge among the important places, it can ask the user herself to identify it (see Section 5.2.2). At this point the user can decide to help the machine, i.e., saying that the location i s her new "Home" location or she can decide that it is not a place worth remembering. In the first situation, the reference to the old home location is deleted while the new one is added. In the second case no action is taken.

- **Knowledge update**. Once the personal view of the world of the user is full of entity instances, the system must keep them updated so that to reflect the contextual information of the user. As said before, the context is a snapshot of the user knowledge that the user believes is important for the situation. This means that the entity instances composing the user knowledge are enabled or

disabled depending on the situation she is facing. A part from enabling or disabling the instances, some of them will also need to update their attribute values, i.e., the user position. This can be automatically done by the machine using the streams of sensor data collected by the user smartphone (see Section 5.3). The analysis is done exploiting the contextual information stored in EB so that to produce meaningful results for the user.

The concepts mentioned in this Section are fundamental for the rest of this section and what follows is a summarization for the reader's convenience:

- The user has a personal view of the world surrounding him, we call it **knowledge**;
- This knowledge is represented using the **entity-centric** approach and is stored into the EB that allows to manage it;
- The knowledge is composed by **entities**, characterized by **attributes** and linked one to the other by **relations**;
- The context is a **snapshot** of this knowledge in a specific moment in time in which all the entities that are relevant for the user in that situation are **enabled**;
- When the context changes, the relations among the entities composing the user knowledge change, this means that some of them are **disabled** while others are **enabled** and for those enabled, their **attribute values** can change as well;
- The fact of enabling or disabling one entity depends on the links they have one with respect to the other. Entities that are linked with enabled entities become enabled;
- The core entity of the system is the **user** which is considered **always enabled**;
- The same process works also for **different representations** of the same entity that can be enabled or disabled.

5.2 Knowledge Instantiation

The entity-centric approach used in this thesis to represent the context requires to instantiate the knowledge schema in order to generate the real entities that compose the user personal view of the world, as explained in Section 3.3. Since this view is composed by any information that can be used to characterize a situation that is considered *relevant* by the user, the main contributor in its instantiation must be the user herself. The machine can eventually trigger the generation of the instances by knowing other elements composing the user's view of the world, but this needs to be approved by the user in the first place.

5.2.1 User Defined Knowledge Instantiation

The user is at the core of the methodology and system presented in this thesis and her help is particularly important in any phase of the knowledge instantiation process for the reasons presented above. There are three main situations in which we foresee the user can instantiate her own knowledge manually:

Profile creation. The first step the user has to perform when she decides to use the SB is to create an account on the platform. This is when the EB is created, with no entities inside. Proceeding in the profile creation phase, some key information are asked so that few entities can be already instantiated. In general the information asked are of three possible categories: (i) personal, (ii) location information she believes are important and eventually (iii) some information related to the services that will be provided by the SB. The registration process happens directly on the smartphone when the user executes for the first time the i-Log application. More in details:

- **Personal information.** These information are the most important because allow to create the entity at the core of the whole knowledge: the user. Some of the main identifying attributes will be asked at this step.
- **Location information.** We believe the Location component of the context is the most important among the four possible ones, for many reason. First of all it is the easiest one to be computed and additionally, is the one that most can help in inferring the other one. A dedicated procedure has been designed so that the user can interact with a map, by placing pins on it and defining names for the locations. In this way we give an interactive tool to the user to instantiate the location entities.
- **Service related information.** The service related information are use case dependent. To mention an example, if one of the services provided to the user is related to her physical activities, most likely some information about her physical situation will be asked, such as height, weight, among others.

The idea is to keep this first step as lightweight as possible and ask only the strictly necessary information so that to not frighten or discourage the user, that will immediately uninstall the application.

Knowledge import. Another approach for manual knowledge instantiation consists in importing the entities from other external repositories. Example of such repositories are the agenda for the entity of type Person, a calendar for the Event entities, among others. The import operation presents some challenges due to the differences in the schema between the original source and the one of the entity-centric approach used in the EB. An additional step is needed to match the information so that the correct attributes are assigned to the new entities. This process is done automatically by the system which leverages of existing solutions that we do not tackle directly in this thesis. In the whole process the user is required only to select which elements to import into the contextual knowledge of the SB choosing among the ones he believes are more relevant.

Manual creation. The last possibility for the user to create his own knowledge is to manually generate the instances using a dedicated tool. This is something similar to what she does during the registration phase but in this case is up to her to decide which ones and how many entities to create. Of course, the more the generated instances the more accurate the system will be in the analysis. Since this can be a time consuming task, we left it as optional. We have a specific place in the mobile application that allows to perform this task. Entities of the four types can be instantiated using an user interface that facilitates the process.

5.2.2 Machine Triggered Knowledge Instantiation

As we said in the previous section, there are different possibilities for the user to create the instances that compose her own representation of the world. Unfortunately all of them involve the user contribution at different levels. A standard rule in human machine interaction says that all the aspects related to the user interacting with the machine must be carefully designed because the user can easily get bored and leave. In order to alleviate this phenomena, since we cannot avoid asking the user, we gave the machine the ability to instantiate the knowledge, and periodically the user can be asked to confirm the machine findings.

There are two possibilities our system allows at the moment:

User Feedback. By analyzing the streaming data the machine can find patterns or repetitive situations that can lead to knowledge instantiation. An example is the one presented in Section 5.3.2.1 where an unknown location is found from the GPS points of the user smartphone. If the machine finds this location for more consequent days this can mean that it is a meaningful location for the user (see Section 5.3.2.2 for details about the procedure to find these locations). For this reason, we decided to create a procedure to ask the user to give her feedback about possible new instances that can be added to his representation of the world. Usually the machine instantiates the knowledge using as attributes the values computed from the sensor streams while the user is asked to give the generated entity only the name. In fact, as explained in Section 3 the schema can be instantiated in an entity only if this entity is given a *name*, that ultimately differentiates it from the other entities. Of course, a meaningful name can be provided only by the user because she is the only one that knows what the specific entity represents in her context.

We designed and implemented a procedure on the user smartphone that at specific moments in time shows a notification that, when clicked, opens a window that allows the user to select a name for the specific instance. As well as all the other aspects of the system, this one as well is configurable and can be adapted depending on the use cases. So far, we have mainly focused in the design of how to ask the names of the Location types (since this was the requirements of our use cases) and the first results can be shown in Figure 5.2a.

The automatic instantiation of entities by the system can be performed only when some entities have already been inserted by the user. At the beginning, without any prior knowledge, this task is unfeasible.

User Live Annotation. Another aspect related to having the user helping the machine to automatically instantiate her knowledge is the one that is based on the annotations made by the user. In our application, the user is annotating her own sensor streaming data using the smartphone. We believe that a standard solution seeing an external expert annotator cannot be used since in this situation the task will require a huge amount of work but most importantly the external human annotator does not have the same understanding of the situation the user has. We do this by administering the user a questionnaire at a fixes time interval designed as a time diary built out of the user knowledge. Time diaries are standard tool sociologist use to ask the user to report her time usage during the day. We extended this concept and adapted them to be used through smartphones [Giunchiglia et al., 2017; Giunchiglia Fausto and Mattia, 2017]. With this new approach we are able to collect answers to very specific questions in terms of labels. The questions are three and regard the main elements of the context, the people, the locations and the events as shown in

Figure 5.2b. These labels can then be used by the machine to create new knowledge. For example, all the locations where the user replied to be at home can be analyzed and a single "Home" location can be found the same can be done for people and activities.

Since the user is not a field expert, we cannot assume that she will be able to generate always truthful annotations. She is incentivated in doing so, but we cannot take it for granted. Our methodology allows us to discover those labels that are more keen to be unreliable through two time parameters formalized as $\mathbf{\Delta_{QA}}$ and $\mathbf{\Delta_{A(X,Y)}}$, where:

1. $\mathbf{\Delta_{QA}}$ refers to memory bias, defined as the inadequate recall of respondents when reporting their time use. The main reason is that the answers are often given with a delay after being carried out [Tourangeau, Rips, and Rasinski, 2000]. The formalized parameter is the time interval (in minutes) from when the question is presented to respondents to when the answer is given.

2. $\mathbf{\Delta_{A(X,Y)}}$ instead refers to carelessness, that is, a set of behaviours that lead to hurriedness when reporting time use. The formalized parameter is defined as the time interval (in seconds) elapsed from when the user starts answering one question of the time diary entry X and answers another question Y, where $X >= 1$, $Y <= Z$, and Z is the total number of questions and $X < Y$. The higher the value, the better in terms of reliability.

A detailed explanation about the two time parameters our methodology allows to identify and use will be presented in Section 13.1.4.2, where we associate them with the use cases referred to the students of the University of Trento.

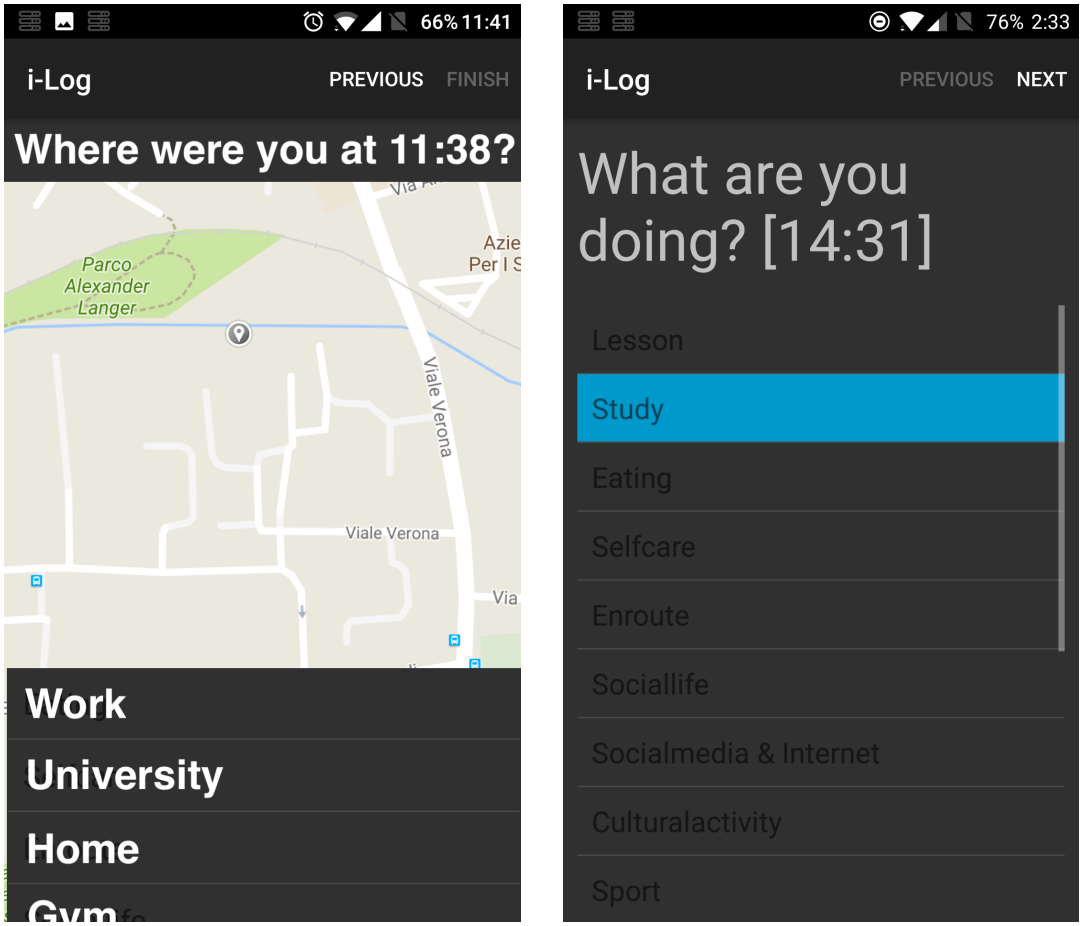
5.3 Knowledge Update

The other fundamental aspect at the core of our methodology is to allow the machine to automatically updating the contextual information about the user. The machine is able to do so by updating the Attribute Values {V} in the corresponding entities that compose the context, stored in the EB. For this task we assume the EB to already contain the entities composing the user context.

As explained in Section 3.2, the schema of the entity-centric approach we are using defines an ET composed by a Concept C and a set of Attributed Definitions $\{AD\}$. In other words, what characterizes an entity type i.e., person, location etc, is the set of attributes they have. A location entity type has a set of attribute that are different with respect to the ones of the person since their intrinsic characteristics are very different. Additionally we defined an Attribute Definition (AD) as a notion used to constrain an attribute composed by a Concept C and an Attribute Type (AT). AT can be of different types, divided into to categories *numeric* or *semantic* data types. The former is composed by boolean, integer, long and float data types that are used to quantify. The latter on the other hand are the so called *relational attributes* that create the link from the origin entity and other Concepts or Entities.

Updating one with respect to the other has different meanings in the context of this thesis:

- **Numeric:** the update of a numeric attribute value is necessary to quantify the entity that then can be aggregated together to infer higher level contextual elements. These updates have little or no meaning for the user while they are necessary for the machine.



(A) Screenshot representing how the i-Log application asks the user to name an entity instance of type Location.

(B) Screenshots presenting how the three questions of the time diary are showed to the user.

FIGURE 5.2: Screenshots of how the system leverages on i-Log to ask the user to help in instantiating new knowledge.

- **Semantic:** by updating a relational attribute the entity is linked with another one. If the newly linked entity was disabled in the current snapshot of the context, it becomes automatically enabled in the new one. An example of this is the attribute Location of the User (that we assume to be always enabled). If this attribute was set to point to the Entity "Work" and the algorithm changes it to "Home" I *disable* the "Work" entity from the current snapshot of the context and I consequently *enable* "Home". This update works for the different representations of the same entity too.

The core of both update methodologies consists in the fact that the sensor data collected by the users are exploited, in combination with the contextual information to update the attribute values of the entities.

5.3.1 Numeric Attribute Update

All the elements in the real world have some characteristics that are numeric values. A person has an height of $1.78m$ and she moves at a speed of $4Km/h$, etc. The former is a characteristic that *identifies* the person while the second one *quantifies* her motion (in this situation). Usually the identifying values are the ones that are static or change very infrequently. On the other hand, the quantifying attributes change more often and are the one we would like to update with the methodology presented in this section.

5.3.1.1 Motivating Example

For the sake of clarity we decided to keep the examples as simple as possible so that one can understand the process underneath it.

Suppose the system needs to know how fast the user is moving because it needs to compute the required time to reach a destination. Of course, the person can use multiple transportation means: walking, using the bus, using the car, among others. Unfortunately we do not have any direct way of inferring the user's speed because we don't have a dedicated sensor on her smartphone. On the other hand we are able to detect the user's car speed because of an IoT device that produces these data. We also have the information about when the user is in the car. By merging these two pieces of information, we are able to infer the required user speed that can then be used to make other analysis.

This very simple example perfectly maps with the objective of the *Numeric Attribute Update* feature of the SB. In our context, there are two entities involved, the user and the car, respectively of Entity Type (ET) **Person** and **Vehicle (is-a Artifact)**. They have different Attributes (A) that characterize the entity themselves. For the sake of simplicity, here we present only the ones required to explain the example. For the **Person** we have:

- **Speed:** this is the Attribute (A) we need to update leveraging on sensor streams and contextual information. It is defined as

$$P_{SPEED} =<< Speed, FLOAT >, 0.0 > \tag{5.1}$$

while for the **Vehicle** we have,

- **Speed:** this is the Attribute (A) collected as a **stream of sensor data** from an external device. It is defined as

$$V_{SPEED} =<< Speed, FLOAT >, 48.0 > \tag{5.2}$$

- **UserPresence:** the car is provided with a sensor that detects the user presence which generates a boolean value

$$V_{USERPRESENCE} =<< Presence, BOOLEAN >, true > \tag{5.3}$$

In this situation the attribute *speed* of the person is the one we want to update using other entities in the context, in this case the *vehicle*. For this, a small computation task must be performed so that the different elements can be merged to produce the result used to update the value.

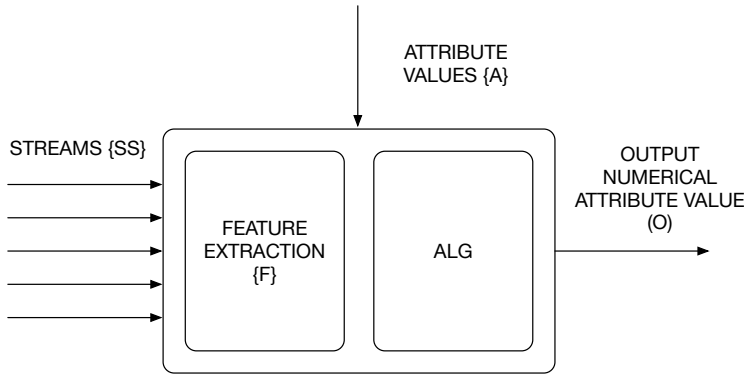


FIGURE 5.3: Schematic of the elements composing a Numeric Attribute Update procedure.

5.3.1.2 Numeric Update Procedure

In order to solve the situation presented in the motivating example above, we present the following procedure that is able to update those attributes that have an Attribute Type (AT) of one of the allowed numeric types (i.e., integer, long, float, boolean).

A **Procedure (P)** is defined by a field expert for each Attribute Definition (AD) in the schema. This procedure will be then applied by the machine anytime an update is requested. As shown in the schematic of Figure 5.3, it is composed by one or more input sensor stream $\{SS\}$, one or more attribute values of other entities $\{A\}$ (optional), a set of features $\{F\}$ to be extracted from the data and an algorithm that processes all such inputs to generate the output of the same format of the Attribute Type (AT) of the specific Attribute Definition (AD). Formally speaking it is defined as a tuple

$$P =< AT, \{SS\}, \{F\}, \{A\}, O, ALG > \tag{5.4}$$

where,

- *AT* is the data type of the Attribute Definition (AD) the procedure is updating;
- *{SS}* is a non-empty set of sensor streams the attribute value has to be computed from. They can be mandatory and optional. The optional ones, if present, can be used to reinforce the output;
- *{F}* is a set of features that need to be analyzed by the algorithm to generate the output. These features are computed from the input sensor streams {SS} by a dedicated Feature Extraction component in the procedure;
- *{A}* is a non-empty set of attributes belonging to the same or other entities whose values has to be considered in the analysis. They can be mandatory and optional. The optional ones, if present, can be used to reinforce the output;
- *O* is the output Attribute of the entity of type AT the procedure has to update;
- *ALG* is the algorithm that generates the numeric value that updates the entity attribute. It takes into account both the mandatory inputs and the optional ones if present and the reinforcement strategy in the latter case should be defined.

Let us illustrate how the procedure is defined for the use case in the motivating example presented in Section 5.3.1.1:

- The situation illustrates the need of creating one Procedure (P) for updating the Value (V) of the Attribute (A) speed of the Entity (E) person.
- The data type AT is a FLOAT
- The input stream is only the vehicle speed, V_{SPEED}
- The feature that needs to be used is the average of the speed values so that to remove unwanted fast changes
- The contextual information input is the presence of the user in the car that maps to the corresponding attribute $V_{USERPRESENCE}$
- The attribute where to update is P_{SPEED}
- The algorithm ALG in this case is straightforward and does not require particular machine learning techniques.

The resulting procedure is represented as follows:

$$P =< FLOAT, V_{SPEED}, AVERAGE, V_{USERPRESENCE}, P_{SPEED}, ALG > \quad (5.5)$$

5.3.2 Semantic Attribute Update

The second type of attributes an entity can have is the semantic type, otherwise called relational attributes. They allow to create links between entities that ultimately create the graph of the context of the person. An example of this attribute is the position of the user: it always refers to an existing location on earth, i.e., Home, Work, etc. In this situation the process of updating the attribute value is a bit more complex and requires additional interaction with the user knowledge database EB system. The process of updating one relational attribute corresponds to the action of enabling or disabling one entity. In fact, when the old attribute value (the reference to an entity) is replaced by a new one, disables the old entity since it is no more involved in the active contest of the user. The process works for different entities but also for different representations of the same entity, as explained in Section 3.3.1.

5.3.2.1 Motivating Example

A smart home environment needs to know when the user arrives at home in order to perform some task, i.e., turning on the lights of the living room. Instead of having dedicated sensors deployed in the environment the system leverages on the data collected from the user smartphone. The smartphone collects multiple sensor streams and few of them allow to determine the user position: the *GPS coordinates* or the *WiFi network the user is connected to*. Both have advantages and disadvantages. The GPS is the most accurate but works only for outdoor and since he drains the battery very quickly, the sampling frequency is set very low. On the other hand, the WiFi is cheaper in terms of battery consumption and, in this particular scenario, is very accurate if the system previously mapped the fact of being connected to the network "home-1234" as being at home. For both sensor streams, an additional step is required to translate the raw data to the higher level situation of "being at home". This step uses the raw data in combination with some contextual information to change the Position attribute of the Person to "Home".

This example, still remaining very simple, shows an additional dimension in the *Semantic Attribute Update* process of the SB, which is, it requires to compare some

results of the analysis of the sensor streams with data in the user knowledge and then return a reference to one of the elements of this knowledge, i.e., "Home". This element, will then become enabled in the active context while the previous relation will be *disabled*.

In this example there are three entities involved (actually two plus one which was not mentioned in the example). These entities are: the User, the Home and the Office. The first one is of Entity Type (ET) **Person** while the other two are **Location**. They have different Attributes (A) and to keep things simple, with present only the ones directly related with this example.

For the **Person** we have:

- **Position:** this is the most recent known position of the user. It is represented as a relational attribute that links the user with another entity that pertains to his knowledge (view of the world). It is defined as

$$P_{POSITION} =<< Position, ENTITY >, E_{SURI} > \tag{5.6}$$

  where E_{SURI} is the identifier (as explained in Section 3.3.1) of the representation the user has of a certain location represented by entity E.

while for the **Office** we have,

- **Position:** this is the position of the user's office in the real world, expressed with an object defined as **Coordinates**

$$O_{POSITION} =<< Position, OBJECT >, coordinates > \tag{5.7}$$

  where *Coordinates* is composed by three numeric float values: *latitude*, *longitude*, *altitude*

$$Coordinates =< latitude, longitude, altitude > \tag{5.8}$$

and for **Home**:

- **Position:** this is the position of the user's office in the real world, expressed with an object defined as **Coordinates**

$$H_{POSITION} =<< Position, OBJECT >, coordinates > \tag{5.9}$$

  where *Coordinates* is composed by three numeric float values: *latitude*, *longitude*, *altitude*

$$Coordinates =< latitude, longitude, altitude > \tag{5.10}$$

- **WifiNetworkAddress:** this is the MAC address of the WiFi router present in the user home. Is is defined as:

$$H_{WIFINETWORKADDRESS} =<< Address, STRING >, "address" > \tag{5.11}$$

5.3.2.2 Semantic Update Procedure

In order to solve the situation presented in the motivating example above, we present the following procedure that allows to update those attributes that have a Semantic Attribute Type (AT).

A **Procedure (P)** is defined by a field expert for each Attribute Definition (AD) in the schema. This procedure will be then applied by the machine anytime an update is requested. With respect to the procedure for the Numeric Attribute Types, this one needs to account for the variability of the attributes of the entities and then is more complex. As shown in the schematic of Figure 5.4, it is composed by one or more input sensor stream $\{SS\}$, one or more attribute values of other entities $\{A\}$ (optional), a set of features $\{F\}$ to be extracted from the data, an algorithm ALG that processes all such inputs and searches for a match $\{SR\}$ in the user knowledge that will be then provided as output to the attribute O. Formally speaking it is defined as a tuple

$$P =< \{SS\}, \{F\}, \{A\}, \{SR\}, O, ALG > \tag{5.12}$$

where,

- *{SS}* is a non-empty set of sensor streams the attribute value has to be computed from. They can be mandatory and optional. The optional ones, if present, can be used to reinforce the output;
- *{F}* is a set of features that need to be analyzed by the algorithm to generate the output. These features are computed from the input sensor streams {SS} by a dedicated Feature Extraction component in the procedure;
- *{A}* is a set of attributes belonging to the same or other entities whose values has to be considered in the analysis. They can be mandatory and optional. The optional ones, if present, can be used to reinforce the output;
- *{SR}* is a set of entity SURI that match the query performed by the algorithm ALG based on the result of the computation of the input streams {SS} based on the attribute values {A}, if any;
- *O* is the output Attribute of the entity the procedure has to update;
- *ALG* is the algorithm that generates the numeric value that updates the entity attribute. It takes into account both the mandatory inputs and the optional ones if present and the reinforcement strategy in the latter case should be defined.

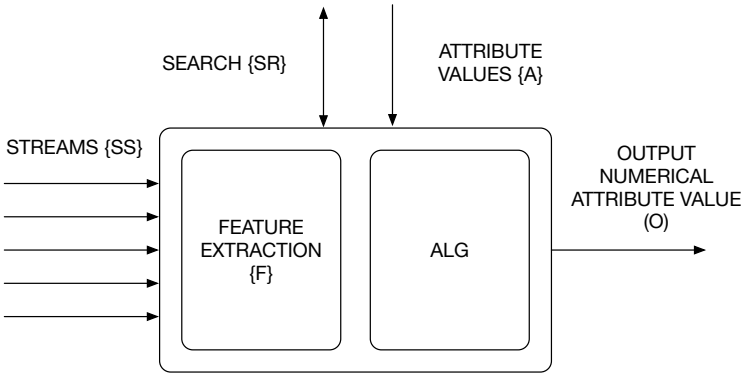


FIGURE 5.4: Schematic of the elements composing a Semantic Attribute Update procedure.

What follows is the description of how the procedure is defined for the use case presented in Section 5.3.2.1:

- The situation illustrates the need of creating one Procedure (P) for updating the Value (V) of the Attribute (A) **Position** of the Entity (E) Person.
- There are two input streams, the *GPS coordinates* and the *WiFi network the smartphone is connected to*;
- The feature that needs to be computed for the GPS data is a median point that averages the last X collected points in a short period of time in order to balance the uncertainly. This is done with a clustering algorithm. For the WiFi network name, nothing is done;
- There is no contextual information input $\{A\}$
- The attribute where to update is $P_{POSITION}$
- The algorithm ALG needs to perform different tasks:
  - Take the point generated from the feature extraction component and, if any, perform a query to the knowledge database of the user, the EB system, for entities of type Location which Position attribute matches the point;
  - Or, if the stream of GPS points is empty, use the second stream, the WiFi network address the phone is connected to. In this situation will search for the entities of type Location which WifiNetworkAddress attribute matches the WiFi address;
  - The SURI of the found entity will be then sent to the output O of the Procedure.

The resulting procedure is represented as follows:

$$P =< \{GPS, WiFi\}, \{CLUSTER, -\}, -, U_{POSITION}, ALG > \tag{5.13}$$

Since the output of the Semantic Attribute Update Procedure (P) is a reference (SURI) to an existing Entity in the user personal knowledge resulting from a search process, it can be that the search does not produce any match. If no match are present the algorithm generates an **Unknown Entity**. To characterize this entity the user feedback is needed and he can decide to help the machine in creating this new entity, by giving it a name, since he believes this entity is important for his context. Otherwise, the user can discard the newly created unknown entity. For details about Entity creation we remind to Section 5.2.

5.4 Summary

In this Section we illustrated the methodology we developed to create meaningful knowledge for the user out of the noise personal big data generated from her smartphone. This methodology is at the core of the SB and will be extensively used in it.

We started by describing what is the user context, that is, a snapshot of the personal world of the user. We modelled it using the entity-centric approach. It is composed by entities with their attribute and relations that interact one with respect to the other. There are four main
glset that describe the user context: People, Locations, Events and Artifacts.

We then described how the knowledge can be generated, dividing this phase into two sub-phases: the knowledge instantiation and the knowledge update. The former is used to create entity instances that are not present in the user personal view of the world but that she believes are relevant. The latter on the other hand is needed to update the existing entities so that to adapt to the context changes.

For the knowledge instantiation we said that the user is fundamental, since only her is aware of what are the relevant entities. The user can instantiate instances manually or can be helped by the machine. The machine can analyze the streaming data and find patterns. Then, it needs to ask for the user feedback to corroborate the findings, ending up in a new entity created.

For the knowledge update, instead, the process is automatic. The field expert must define one Procedure (P) for every AD of the entities so that the system is able to use it, when needed, to update the Attribute Value (AV). There are two types of values that can be updated, numerical values or semantic values.

Chapter 6

Knowledge Exploitation

This section explains how the knowledge generated and maintained in the SB can be exploited to help the user in her day life situations, which is the ultimate goal of the system presented in this thesis.

At some extent the SB can be assimilated to a Weak Artificial Intelligence (Weak AI) considering the following definition[1] "Weak AI is machine intelligence that is limited to a specific or narrow area. Weak AI simulates human cognition and benefits mankind by automating time-consuming tasks and by analyzing data in ways that humans sometimes can't". This is exactly what the methodology and system defined in this PhD thesis do: the machine is able to analyze huge amounts of sensor information collected from the user and the world in a way that is meaningful for the user herself and that otherwise she alone could not analyze. The definition also mentions what the AI is intended for, "to benefit mankind by automating time-consuming tasks...". The way we see this automation task is through a set of **services** that help the user in her day life situations.

6.1 Personalized Services

Generally speaking a service is effective when satisfy the user, in every dimension. It is almost impossible to create a single, unique service that satisfies all the people using it since everyone has very different needs. To solve this issue we propose to use highly personalized services that are built on the user preference. The personalization comes at different levels:

- Among all the available services, the user is free to decide which one to use;
- Once the services has been selected, the user can decide when to enable it. All the services are configurable;
- The content of each service is based on the user knowledge.

In order to have such personalized services the system needs to know all the aspects of the user's life, in other words it needs to know the user context. In the SB, the EB contains exactly this, a snapshot of the user context. Each service needs to monitor changes to the involved entities in the EB and react accordingly. The service can leverage only on the already present information in the EB but it can also generate new information that can be saved back to the user knowledge and then proposed directly to the user in its various forms: quantified self reports, notifications, among others.

[1] http://www.investopedia.com/terms/w/weak-ai.asp

The services are strictly related to the use cases and to the users. In the use cases we worked so far for example we focused on the citizens of a city interested in mobility issues and on university students interested in understanding how their habits influence the academic performances. For every use case there is the need to understand the user needs, that usually depend on their demographic characteristics, sex, age, occupation, etc.

We identified the following procedure to create a service in the SB:

- **Understanding the user needs**. A way we used to understand these needs is through **focus groups**, namely a group of people representing the sample of the final users using the system for the specific use case. These people can be asked to reply to some precise questions the scientist have or to propose their own ideas of a possible service.
- **Service design**. The scientist translates the high level user requirements identified in the focus groups and design the service, defining the outcome, the process and the input to use.
- **Schema design**. It can be that some new entities need to be created because not present in the system. This is done by a field expert. Once the schema is created, these newly created entities can be instantiated by the single users in their knowledge.
- **Update procedures design**. Once the instances are created they need to be used by the service to produce its output. Similarly to the others entities, this is done by updating their attribute values. This task requires a second field expert to design the update procedures as explained in Section 5.3.
- **Service knowledge**. Some services requires to keep track of their operation and have their own representation of the world at a global level outside each user's knowledge. Then, an EB can be instantiated for the service to store its information, again following the entity-centric approach.

To design a service, these steps have to be performed. After a test phase, the service is made available to the users that can subscribe and start using it. The subscription involves the user to give explicit consent to the usage of her data according to the latest privacy regulations. The way we implement service is through a **publish-subscribe** mechanism where multiple publishers can propose their own services. We remind to Section 10 for details.

6.2 User Privacy

In a system such as the one described in this thesis that has to deal with personal data, all the privacy related aspects must be carefully taken into account. We already mentioned the GDPR in Section 4.2.0.3 related to the data collection phase. It is a set of regulations that will start to be applied in May 2018 to all the companies and institution that collect personal data about European citizens. The main novelty point is that the user should always be in control of her own data.

In the SB the privacy of the user is guaranteed by design. Since in this section we illustrate how the user generated knowledge can be exploited to provide useful services, this is where the privacy is more important. In fact, every service must have access to the user data to elaborate them and provide the desired output to the user.

The way we made the SB compliant to GDPR is thanks to these elements:

- **Authentication.** Every operation performed in the SB needs to be authenticated. This is done to guarantee the separation of the data among the users. Once the user register to the platform, she is given a username and a password that must be used on the smartphone to collect the data and, in general, in the user interfaces she interacts to, both on the desktop and on the mobile;

- **Anonymization.** One important point GDPR impose to the companies is to take all the possible precautions to protect the user data and prevent data leaks. This can be guaranteed by adopting software and or hardware solutions to isolate the data from ill-intentioned. However, it is possible that even if the best solutions have been adopted, some data can be accessd from outside the company. To prevent this, we decided to completely anonymize the data stored in our systems. Once the data is sent from the smartphone to the server, and before storing it, it is anonymized so that, even if someone access it, he will not be able to refer back to the user who generated it;

- **Access control.** A part from the authentication, another practice we adopt in the SB is to implement access control policies based on roles. Each roles can access different elements of the system and can perform different tasks. However, **only the user can access her own data**. Neither the administrator, nor developers or others can access the user data stored in the databases if the user didn't express her explicit consent;

- **Data Subscription.** The last element that characterizes the privacy related aspects in the SB is the fact that the way the services are provide is through a publish/subscribe mechanism. The service is made available from one producer (multiple producers are allowed in the system) and the user has to subscribe to it. The subscription process comprehends a series of steps that involve the user that has to explicitly give the consent to allow the usage of her own data from the service. The subscription always has a duration and after that the service has to ask the user again.

More details about these elements will be explained in the reference architecture presented in Section 10.

6.3 Summary

In this section we described how the knowledge internally generated by the system from the personal big data collecting from the user can be exploited to provide services that ultimately will improve the user's quality of life.

We described which are the main characteristics of the highly personalized and context-aware services provided by the SB. We then explained how one service can be designed, starting from the user needs.

We also explained that in this phase the privacy of the user is crucial, when the system has to access her data. The SB is GDPR compliant by design. In fact, the user is always in control of her own data and decides which one in the system can access them and for how long it can do it. The data is stored in an anonymized way so that it is not possible to lead back to the user who generated it.

Part III

The Reference Architecture and System

Chapter 7

Reference Architecture

In this chapter we present the reference architecture for the SB which integrates different components that interact with each other in a complex manner. We present the different roles of these components and how they can work together to address the problem tackled in this Ph. D. thesis. The architecture itself is one of the contributions of the thesis.

The first step when defining an architecture is to elicitate the high-level requirements (Section 7.1). For the SB we analyzed the sub-problems composing the main problem we are addressing in the thesis and from there we defined the requirements. We organized them around some dimensions that are considered of key importance in the design of such a complex system: the data collection, the data storage, the user participation, the services, the privacy and the performances.

Starting from these requirements we designed the system architecture keeping in mind that the problem needed to be tackled in a general way. The architecture must be abstracted from any application specific constraint in order to be used in any possible scenario and context and with any possible technology.

The resulting architecture is composed by multiple components that can be grouped into three categories defined by their purpose: Data Acquisition and Management Subsystem, Knowledge Generation Subsystem and Knowledge Exploitation Subsystem. Each of these subsystems and their components will be explained in details in Chapters 8, 9 and 10 respectively.

Finally, we described how the components communicate one with the other to work in symbiosis to solve the main problem at the core of this thesis. In details, we present a dynamic view of the system that highlights how the different components interact in the three main situation that can arise: the data is collected from the user, the knowledge is generated out of the streaming sensor data and finally how this knowledge is used to provide services to the user.

7.1 Requirements

The requirement analysis for the architecture of the SB starts from the problems defined in Chapter 2. In details, we focused on defining the requirements starting from the different sub problems identified in the three macro areas: Data Acquisition and Management, Knowledge Generation and Knowledge Exploitation. Even if the three areas are very different, the requirements are shared among them and we will present them to address the global problem. What follows is the results of this analysis.

Data Collection. The system architecture described in this thesis deals with sensor streaming data about the user. The requirements are described as follows:

- The data can be collected from any source that is able to generate streams of sensor data. The smartphone is the best candidate but others are allowed as well and should be integrated into the system through dedicated data import pipelines;
- The data collection from the smartphone is done through a mobile application. A device that allows to install third-parties application is required. i-Log is currently supported on Android and the iOS version is under development;
- The installation process of such application and the creation of the user profile must be easy and fast to be done;
- The data collection process from the smartphone must not affect the device usage under any circumstance. Using the app we expect a little increase in the battery consumption, a small percentage of the internal storage to be occupied, no performance degradation;
- The data need to be periodically synchronized with the backend server for the analysis. The device should be connected to a WiFi network at least once per day if the user does not want to use her 3G network and incur in additional costs.

Data Storage. In this thesis we have two different types of data that are very different one respect to the other but that must be used together to solve the problem. The first one is modeled according to the entity-centric approach, that uses the notion of **entities** to represent the data, i.e., the knowledge of the user as described in Chapter 3. These kind of data is stored in the so called Entity Base system. How to model these data and how the EB works are elements outside the scope of this thesis. The second one are the **streams** of sensor data collected from the user smartphone and represented as time series. These data are at the core of the work presented in this thesis. These data types have direct impact on the requirements of the SB. In details, for the **entities** we have this set of requirements:

- Each user has one personal Entity Base (EB) system so that the information are stored separately from the other users. Since we leverage on others work for this component of the system, we assume that the separation can be either physical (multiple instances of the Entity Base) or logical (one instance of the Entity Base and the user's knowledges are logically separated);
- The knowledge schema must be shared among the users so that to simplify the design process and to being able to reuse the knowledge generation algorithms. Therefore, the system itself should provide a point of reference to get a basic and extendable schema as well as general purpose knowledge composed by entities of general interest like the geography of the territory, etc.

Considering on the other hand the **streams**:

- The data are logically separated one respect to the other due to privacy issues. There is one big storage system that collects all the steaming data of the users;
- We assume that the raw data referring to the past will be eventually deleted when the analysis have been performed, so that to free up space for new data;

- Since the amount of data is huge, we assume that the database system must be distributed and can scale up (and down) depending on the number of users using the system.

User Participation. The user is the core of the SB.

- In order to get the services back, she needs to share with the system the collected sensor data;
- The user should be as honest as possible when is asked to collaborate in the knowledge instantiation. If she cheats, the system can be misled;
- All the tasks that require the interaction of the user should be very easy to perform in order to be able of being executed by any user.

Services. Through the SB it should be possible to access different services allowing the user to exploit her personal data to improve her quality of life. This means that:

- The system (or external entities) should provide these services so that the user perceives the SB as useful;
- Different services should be designed and implemented in order to satisfy the different preferences of all the users;
- The services should exploit the personal streaming sensor data, the user's knowledge and if needed also external knowledge.

Privacy. The SB deals with personal information that can be considered sensitive, i.e., the user's location. This may raise privacy concerns that needs to be taken into account. New regulations will start applying soon and the system described in this work can tackle most of their new elements:

- The data should be stored anonymized in the databases. This means that, is someone can violate the security measures and come into possession of the data, he cannot associate the data with the real person who generated it. A unique identifier is assigned to a newly created user and this identifier will be used to store the data. One single table called "mapping table" should contain the mapping between the identifier and the user and this table should be saved in a safe place;
- The data should not be shared with other entities outside the SB if the user did not allow this;
- The user should always be in control of her data, she can access and delete them when she believes this is needed;
- The control over the data should be extended also over those elements of the system that use the data. This means the user should explicitly grant access to her data from a component of the system.

Performance. The system is expected to be designed to handle an arbitrary large number of users, collecting data from them, generating their knowledge and providing them services. This is mapped to these architectural requirements:

- The system should be able to scale in the number of users while still providing acceptable performances in all its tasks;
- The system should be able to scale with respect to the number of sensors and entities per each user it can manage;
- The system should be able to scale and handle different users, in different situations, with different needs while still providing good quality of service.

7.2 System Logical View

A logical view of the reference architecture is presented in Figure 7.1. This view shows the different components that are part of the SB and how they are connected together. The logical division in the three areas is also showed: Data Acquisition and Management Subsystem, Knowledge Generation Subsystem and Knowledge Exploitation Subsystem. Each subsystem will be presented in the next Chapters, 8, 9 and 10 respectively.

7.3 System Dynamic View

The dynamic view of the system outlines the interactions among the different system components in performing a task. These interactions can be very complex and involve multiple components that work on parallel threads. However, a simplified view of such interactions can help in understanding the overall system behaviour for three of the most important tasks the system has to deal with: the data collection from the users, the generation of the knowledge and the knowledge exploitation for the services.

7.3.1 User Data Collection

Figure 7.2 shows the sequence diagram of the operations needed to synchronize the sensor streaming data from the users smartphone to the server.

When the smartphone synchronizes the data with the backend, the first operation is to reach out the API where to upload the compressed log file. Together with the file, the user credentials are sent for authentication that is performed by the dedicated component in the SB. If the authentication succeeds, the procedure can start. The API generates in output an array of compressed bytes that is sent to the unzip component. It uncompresses the data and generates a byte array of uncompressed data which is sent to the pre-processing stage. The pre-processing adapts and restores the original format of the data that was modified by the smartphone. These data are then matched to the schema of the destination database, mapping each sensor variable. At this point, the data can be inserted in the database but to do so, the component must know where to store the data. As we will explain in Section 10.2, the data are stored anonymized in order to preserve the user privacy. The anonymization is obtained by storing the data with a UUID that is a random generated string of 40bytes. To store the data then, the Data Insert component needs to know this UUID for the authenticated user. This information is stored in the anonymization component that, after having determined that the request is legit, sends the UUID through a secure connection. Finally, the data can be stored in the appropriate tables under the user database.

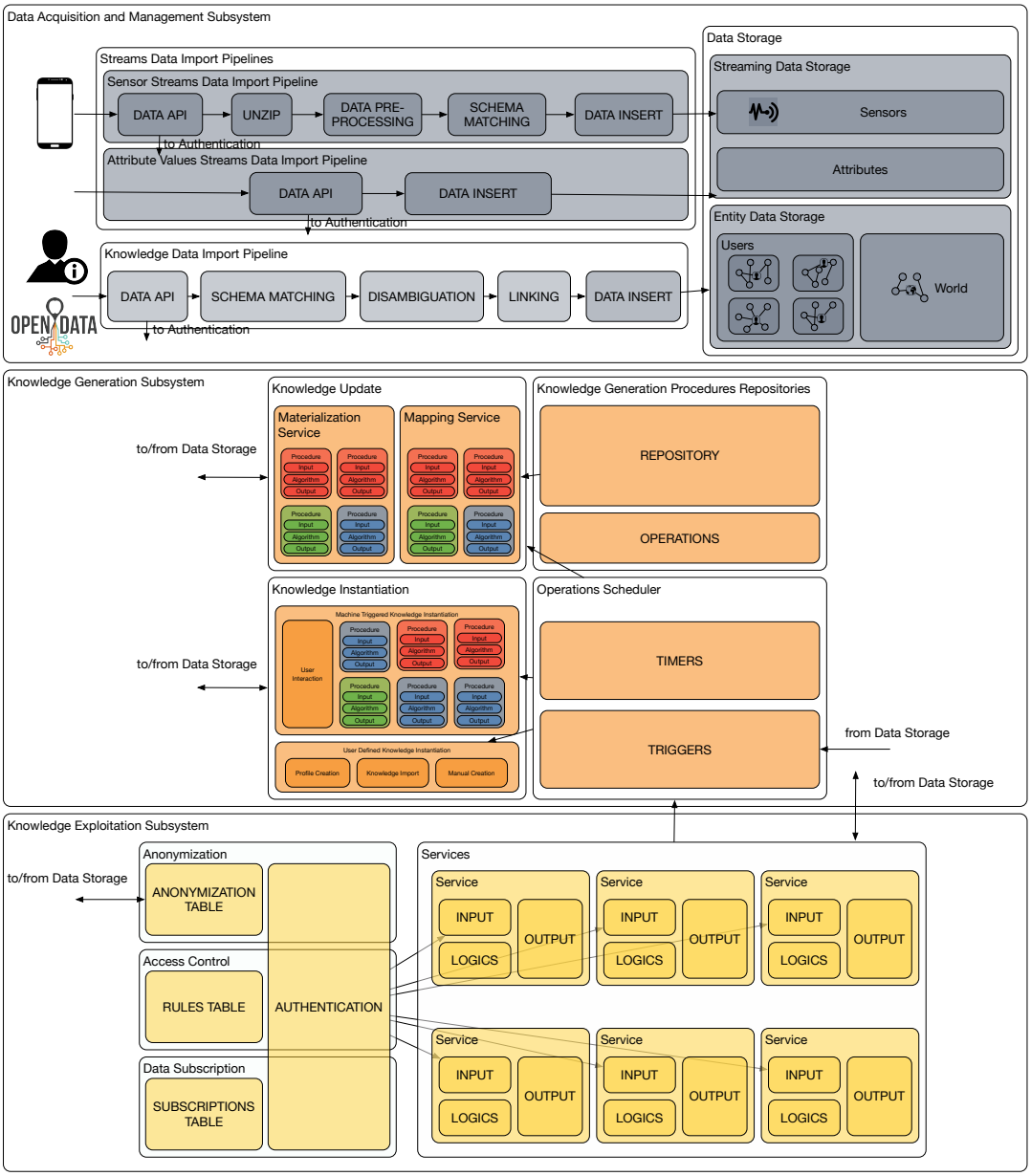


FIGURE 7.1: Schematic of the reference architecture of the SB with the three main subsystems: Data Acquisition and Management, Knowledge Generation and Knowledge Exploitation.

7.3.2 Knowledge Generation

Figure 7.3 shows the sequence diagram of the operations needed to automatically generate knowledge from the user collected data. The knowledge generation task we are referring to here and we took as an exmaple, is the Numeric Attribute Update as defined in the example illustrated in Section 5.3.1.1. The example is about the system that needs to know how fast the user is moving, without having any data that refers to it. It needs then to compute this value using a Procedure defined by the field expert, that uses the high level information of the user being in her car plus the speed of the car itself. By merging these two, the system can automatically update the user speed attribute value with the speed of the vehicle.

The process starts with the user manually creating one entity she believes is relevant for her context. Once the entity is created, the system searches in the Knowledge Generation Procedures Repository if a procedure that automatically updates the values of the attributes of such an entity has already been defined by the field expert. For the sake of simplicity, in the schematic of Figure 7.3, this is shown only for one of the attributes but in the final system this happens for all the attributes of the newly instantiated entity. In this case, the procedure is already present and then is instantiated in the Knowledge Update component of the user. In particular, the instantiation is in the mapping module. From now on, this procedure will be available and able to automatically update the values of the specified attribute, according to events i.e., changes in the context, or on a time bases. In this situation we considered a timer that schedules the update periodically. Every time the timer triggers the knowledge update, it calls the procedure that performs the following operations:

- It requests and obtains the contextual information (if needed for this attribute) to the corresponding Entity Data Storage component.
- It requests and obtains the streaming data (if needed for this attribute) to the corresponding Streaming Data Storage component.
- It then performs the computation defined by the algorithm implemented in the procedure.
- Finally, the result of the computation is mapped to the Streaming Data Storage in the corresponding stream of the attribute values.

We said previously that an update is composed by two tasks: mapping and materialization. The mapping computes the new attribute value and stores it in the Streaming Data Storage while the materialization takes this value and updates it in the Entity Data Storage to update the context of the user. In this specific example, the materialization happens together with the mapping. Then, the newly generated attribute value is read from the Steaming Data Storage and is copied as is in the Entity Data Storage.

7.3.3 Knowledge Exploitation

The last example we present for describing the interactions among the different system components refers to the task of exploiting the knowledge to provide services to the user. This situation is presented in Figure 7.4.

As explained in the methodology, the user must subscribe to benefit from one service. This is done by contacting the Publish/Subscribe system service of the Knowledge Exploitation Subsystem. The user requires and receives the list of services, and then requests for subscribing to one of them. The service replies with a request for permissions to the user. This request comprehends information about which data the service needs and for how long it needs it. Once the user grants these permissions, a new entry is inserted in the Data Subscription table in the corresponding component of the SB. From that moment on, the service is instantiated and starts operating. To operate, it needs data, that are stored anonymized to preserve the privacy of the user. To de-anonymize the data the service needs the UUID associated with the user, which id stored in the Anonymization component of the system. Once the service receives the key, it can query the Entity Data Storage and retrieve the data it needs. After some processing of various form, e.g., visualization, aggregation, among others, the service can be provided to the final user who requested it.

7.4 Summary

In this Section we illustrated the reference architecture for the SB that implements the methodology previously presented.

We started by defining the high-level requirements the system has to satisfy in terms of data collection, data storage, user participation, services, privacy and finally performances.

We then illustrated a logical view that consists in a schematic showing all the components of the SB. They can be grouped into three sub-systems: Data Acquisition and Management Subsystem, Knowledge Generation Subsystem, Knowledge Exploitation Subsystem. Everyone of them will be presented in details in the next Chapters, describing what are their objectives and their components.

Finally, we presented a dynamic view of the system that illustrates how the different components interact one with respect to the other to provide three main functionalities: data collection from the users, knowledge generation from such data and exploitation of such knowledge to provide services to the user that will improve her quality of life.

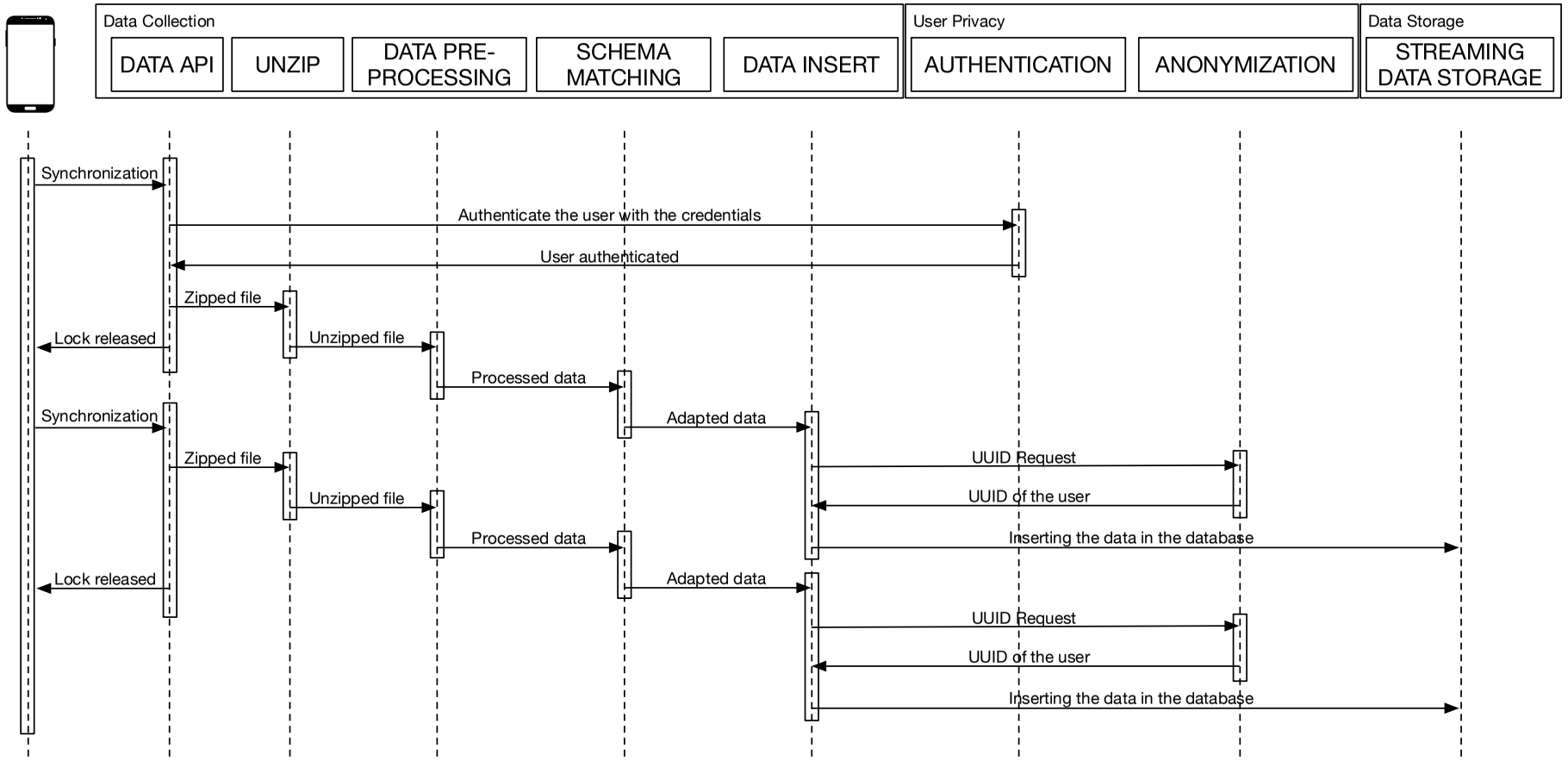


FIGURE 7.2: Sequence diagram of the user synchronizing log files of streaming data collected by her smartphone.

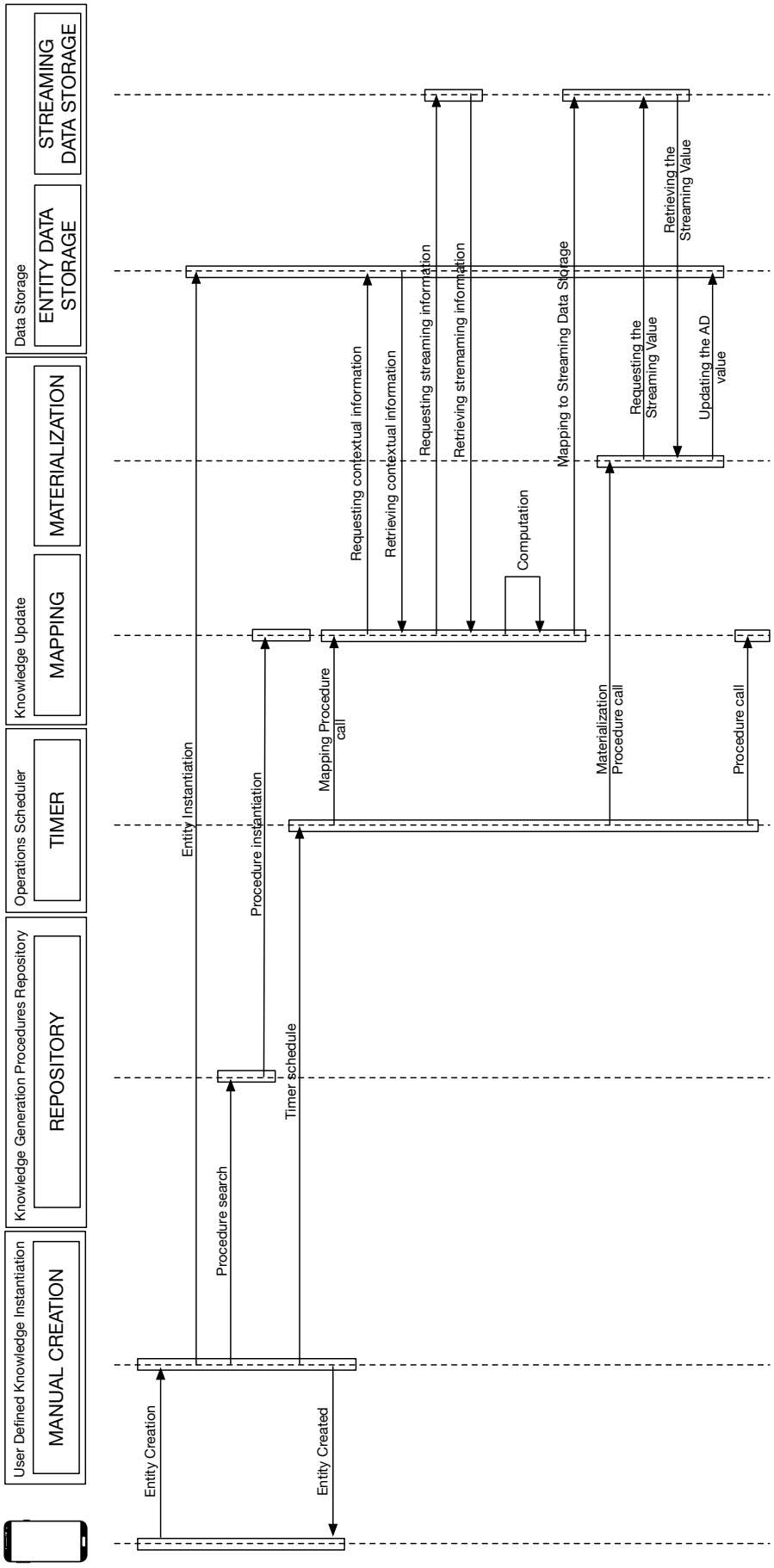


FIGURE 7.3: Sequence diagram of the knowledge generation phase. The user manually instantiates an entity and the system starts to automatically update one of its attributes.

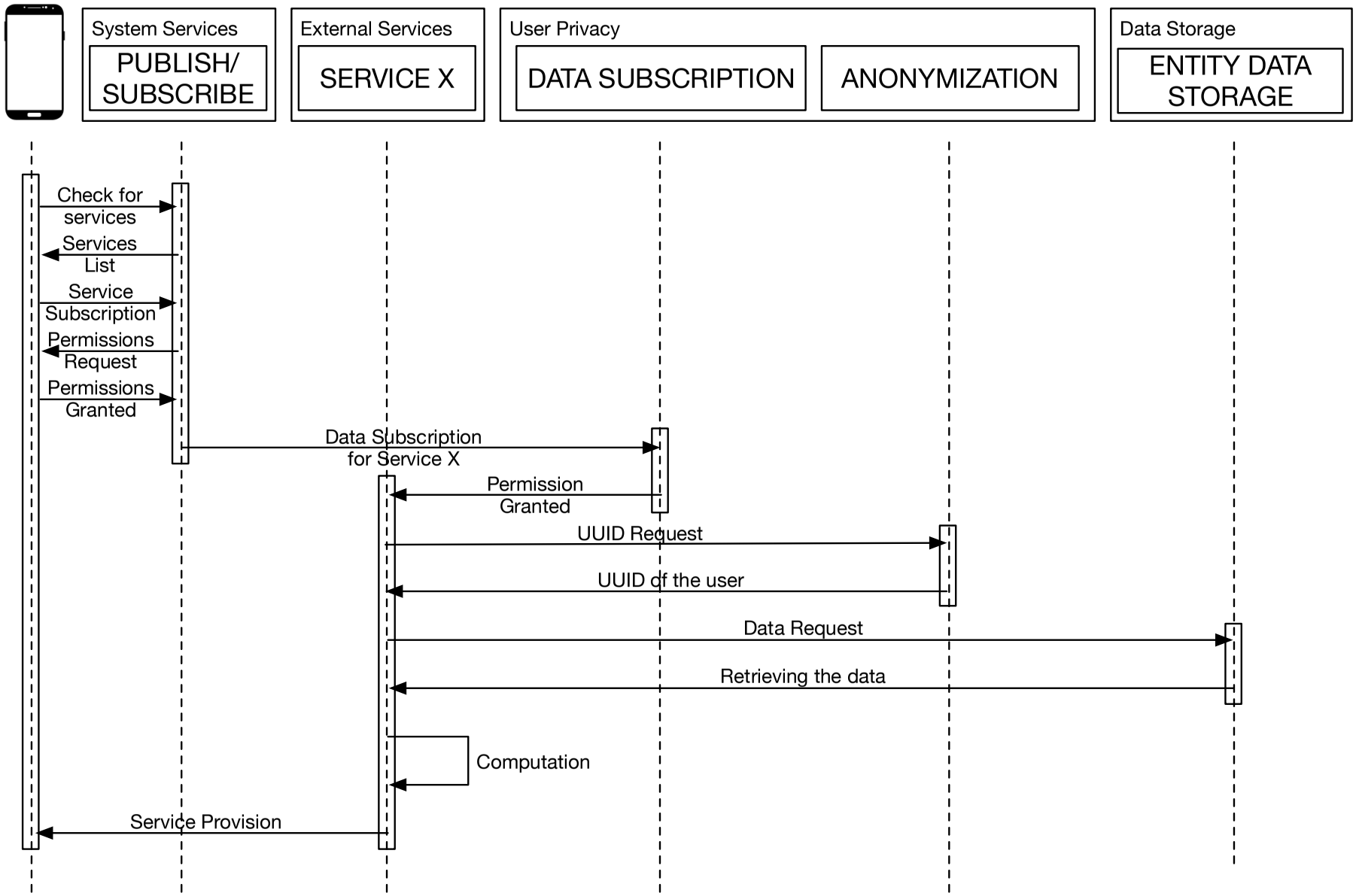


FIGURE 7.4: Sequence diagram of the system exploiting the user generated knowledge to provide a service.

Chapter 8

Data Acquisition and Management Subsystem

The user generated data, also called personal big data are at the core of the SB. The analysis of such data, through the knowledge generation phase, enables the highly personalized and context aware services that the user will ultimately use. Without any data, the system cannot perform its tasks and ultimately solve the problem at the base of this thesis.

The data acquisition and management task becomes crucial and a potential point of failure for the whole system. To guarantee the collection of the data we meticulously followed the requirements designed in Section 7.1 in addressing the problems presented in Section 2. We then designed the subsystem to be made of a series of components that work in symbiosis to efficiently collect the data from the user's smartphone. A schematic representation of the logical components of the Data Acquisition and Management Subsystem is presented in Figure 8.1.

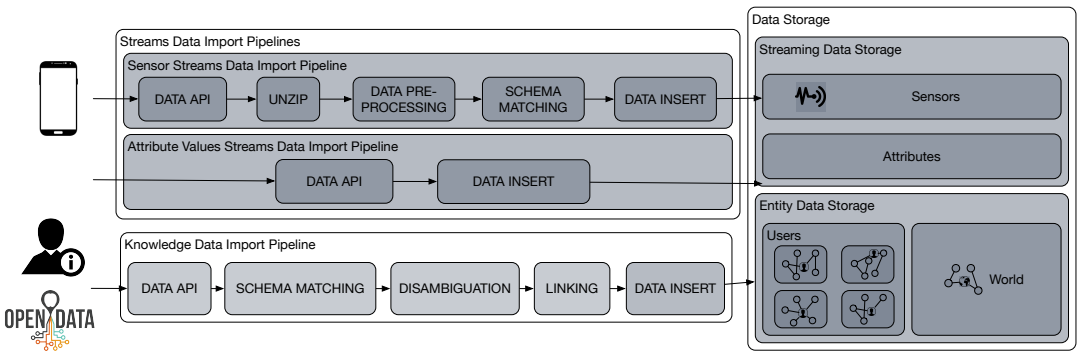


FIGURE 8.1: Schematic of the components of the Data Acquisition and Management Subsystem.

We can logically split the subsystem into three main groups of components: the data sources, the importing pipelines and the data storage systems. Each of them has specific tasks that can be described as follows:

- **Data Sources.** The system deals with different types of data that are generated from different sources. On the one hand there is the user, where her smartphone can be used to generate stream of sensor data, but also high level knowledge. The knowledge is generated in terms of answers to specific questions i.e., through feedbacks. The other source of information do not regards the user, and regard any other form of data referring to the world, composed by streets, buildings, institutions, etc.

- **Data Import.** The data are generated from different sources, namely the user or external actors. The data import component allows create data pipelines that adapt the incoming data to a format that allows to be stored in the two data storage components of the SB. There are then two different input strategies: one about streaming data and one about the knowledge. This last element will only be superficially described since is not part of this PhD thesis.

- **Data storage.** In this thesis we have two different types of data: the streaming data and the knowledge data. For each of them we have a separate storage solution. The reason why we separated the data is that they have very different characteristics and requirements and then they are modeled differently: one are time series data while the other follows the entity-centric approach. The latter is outside the scope of this thesis, and then we used a solution developed by the other members of the Knowdive Group[1] as presented in Section 3. For this reason we will only present some key elements in the following sections about it while we will focus our attention on the storage for the streaming data.

In this Chapter we are presenting one subsystem of the reference architecture of the whole SB. As we said before, the reference architecture it must address the problem in a general way without focusing on specific solutions or technologies. Here we present only the high level considerations we made to design such components, leaving to Chapter 11 the description of the technical solutions adopted for the implementation of the reference architecture in the working SB.

8.1 Data Sources

The system collects all its data from external sources that will be then used internally to generate the user knowledge. The most of the information comes from the user which is at the core of the system. Additionally a small part can come from other sources that allow to better represent and understand the surrounding of the user. The main distinction the system does according to the data it manages regards their type:

- **Streaming Data.** The stream data sources mainly refer to the user generated streams from the sensors embedded in her smartphone. As we said before, they can easily generate huge amounts of data that must be analyzed by the system. However, the system allows to collect, store and use also other streaming data sources: these data are streams become from IoT devices or any other sensor embedded in an object that can be used to represent the world. In a smart city application for examples it is possible to have sensors embedded on the public transportation cars so that to have a clear understanding of where they are and how they move. The two sources can be then used in combination to create a more immersive user experience.

- **Knowledge Data.** The other data this system deals with are the knowledge information. A we said, they are modelled using the entity-centric approach and stored in the Entity Base (EB) system. The main source of these type of data is the user, who can generate them helped or not, by the machine. Additionally, the system is able to deal with other sources of knowledge information.

[1] http://disi.unitn.it/~knowdive

These sources can be various and the data can represent many things. Referring back to the example of the smart city, an example of knowledge can be the list of buses and routes and timetables[2] they follow. Other examples are the data provided by the institutions as Open Data. Having such information and integrating it with the streaming data coming directly from the vehicles and from the user can really create useful services that add value.

8.2 Data Import

Once the data have been collected they must be imported into the storage components of the system, both the Entity Data Storage (EB) and the Streaming Data Storage. In this Section we will describe in details how the import of the streaming data works while just describing at high level what are the main steps of the knowledge data import, since the EB is outside of the scope of this thesis.

8.2.1 Streams Data Import Pipeline

Figure 8.2 shows a detailed schematic of what are the components of the streams data import pipeline we designed. As already said, the streaming data are composed by two different types of data that must be imported: the sensor streams and the attribute values streams. For the latter the procedure is easier since the data is generated inside the SB, and no pre-processing has to be performed. The former on the other hand requires additional steps to adapt the external data acquired from the sensing devices to the internal schema where the data is stored as time series.

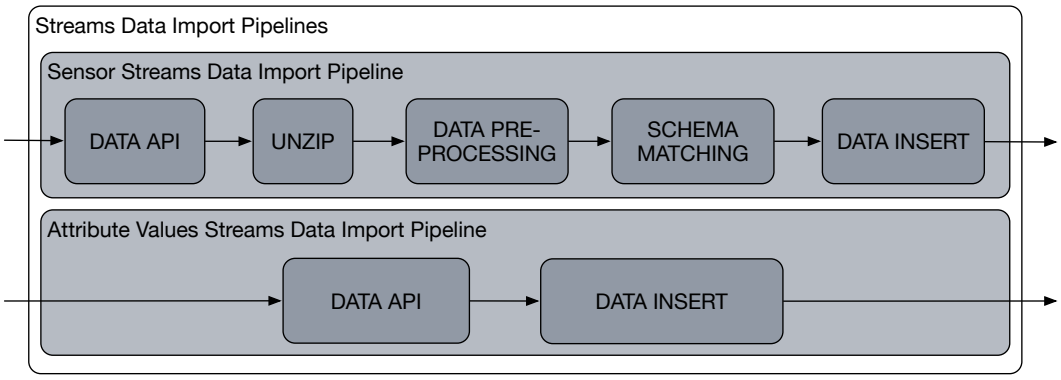


FIGURE 8.2: Schematic presenting the two pipelines used for importing the streaming data into the system database.

8.2.1.1 Sensors Streams

The sensor streams are generated from sources that are external to the SB. The are no constraint about which device and which format to use since this pipeline is developed with the purpose of adapting the data. If the input devices are different, multiple pipelines can be developed and instantiated. The only requirement on the input device is that it can send the data over the internet to the SB. In this work we are focusing on personal data and we defined that the user's smartphone is the best candidate as data collection device in such a scenario.

[2] https://developers.google.com/transit/gtfs/

The sensor streams data import pipeline is composed by five elements that are used in series, meaning that the output of the previous one is sent as input of the next one. The components are:

- **Data Api.** The entry element of the pipeline is a component that implements a RESTsul API. The smartphone when synchronizes the data, sends them to this endpoint. An authentication mechanism is implemented so that the data is recognized and associated to the user who generated them. The username and password are sent with the request for this purpose. The output of this component is a stream of compressed bytes. This is the only component of the pipeline which is blocking. This means, till the upload operation is finished the phone cannot upload another file. From this step on, everything is done exploiting the multi-threading capabilities of the processor using concurrent threads.

- **Unzip.** Since the smartphone has limited resources, in particular in terms of internal storage and amount of data it can synchronize (if not on WiFi), it has been decided to compress the generated data. This process is explained in details in Section 4.1.2. To summarize, the device generates comma separated value (csv) files, where each row consists in one sensor measurement. When a certain number of rows are added to the file, it is compressed and stored, while a new one is created. These compressed logs are the one sent over the network to this pipeline. This block of the system receives the compressed stream of bytes representing the log from the previous component of the pipeline and decompressed it using the bzip2 algorithm. The uncompressed stream of bytes is let made available to the next component.

- **Data pre-processing.** In order to reduce the size of the generated data on the phone we removed useless decimal places for every collected value. Unfortunately the roundup operation turned out to be expensive in terms of CPU usage considering that we need to round thousands of values per second. The solution we found was to apply the following formula

$$OUT = (int)(IN * D) \tag{8.1}$$

  where

  - *OUT* is the output, represented as an integer;
  - *IN* is the input float value that needed to be rounded;
  - *D* is a multiple of 10, depending on the decimals that need to be preserved, i.e., 10 for 1 decimal, 100 for 2 decimals, and so on.

  This operation corresponds to converting the original float into an integer number. A float value $1,783467$ is converted into an integer value of 178 if only two decimals were enough.

  On the server these data must go through a pre-processing step that converts back the values in the log files to their real float value, by diving it by the correct multiple of 10.

- **Schema matching.** Once the values have been pre-processed, they must be mapped to the schema so that they can be stored. This process is done by a

dedicated component that maps the values from the csv file to the corresponding label in the schema. The mapping consists in taking the right element of each line, at positions 0 to N, and mapping it to its label.

- **Data insert.** The last step consists in inserting the data into the database. This component is technology dependent since different databases use different ways of inserting the data. One can use a tool to for importing some raw file, while others have a query language that allow to directly insert i.e., SQL, CQL, etc. In some situations this procedure can be even parallelized so that to obtain faster inserts i.e., with Apache Cassandra.

8.2.1.2 Attribute Values Streams

The pipeline for importing attribute values into the streaming data storage is simpler since the data is generated in the right format by the knowledge generation subsystem. There are only two steps in this case:

- **Receiving the data.** Receiving the data from a component internal to the SB is different with respect to an external one. But, the internal/external concept is non-trivial in the architecture presented in this thesis. In fact, thanks to the high modularity of the SB (as will be explained in details in Chapter 11) every component even if it is logically grouped with others, it can run on the same or different physical machines. According to this, and since we need to design an architecture that is general we have to make the right design decisions about how the different elements exchange the information.

  Going back to the problem of receiving the data from the knowledge generation subsystem, we decided to discard the possibility of sharing the data in-memory and instead going for a generic broker/messaging system;

- **Data insert.** Similarly to the previous case, the data must be loaded into the database and how to do it depends on the database technology.

8.2.2 Knowledge Data

As presented in Section 5.2.1 the user is allowed to import knowledge from external sources e.g., contacts from the agenda, events from the calendar, locations from the map, into her personal knowledge base. The process consists in selecting which elements to import and instantiating new entities using their values. The import of knowledge is a non-trivial task that requires different aspects to be considered, but since the knowledge modelling aspects are outside the scope of this thesis, we just present the main sub-problems of such a task, without going into the details. The knowledge data import pipeline is presented in Figure 8.3.

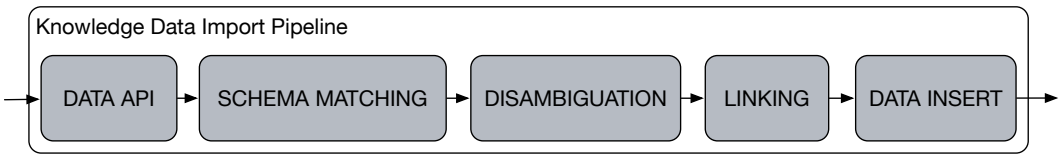


FIGURE 8.3: Schematic presenting the pipeline used for importing the knowledge data into the system database.

The main steps of the knowledge importing procedure are:

- **Receiving the data.** Usually the knowledge data is saved according to a format that allows to give a structure to the data, a csv file, an ontology, a json or an rdf. The first step of the process is to receive these data, usually from a data catalogue, and then read its content and make it available to the next component in the pipeline.
- **Schema matching.** The source data, being them complex objects, can be structured according to a different format with respect to the schema of the destination. If this is the case, the original data must me mapped to match the knowledge schema. This is done by mapping the values at the origin with the attributes at the destination.
- **Disambiguation.** Once the entities have been created some of them can have duplicates since they were instantiated from unstructured (or semi-structured) data. The disambiguation phase allows to delete duplicated entities or merge partially completed ones. For example, consider the two entities (as formalized in Section 3.3) defined as

$$E_1 =< .., .., M.Rossi, .., \{<< AGE, .. >, 29 >\} > \tag{8.2}$$

and

$$E_2 =< .., .., MarioRossi, .., \{<< HEIGHT, .. >, 1.78 >\} > \tag{8.3}$$

They refer to the same person and then are merged together:

$$E =< .., .., MarioRossi, .., \{<< AGE, .. >, 29 >, << HEIGHT, .. >, 1.78 >\} > \tag{8.4}$$

- **Linking.** Another problem that can occur regards how the entities are linked together to compose the knowledge. It is usually hard to detect the links when importing from unstructured data. The links can be reconstructured using specific techniques.
- **Data insert.** Once all the entities, their attributes and their relation have been discovered the whole knowledge can be imported into the EB and merged with the already existing knowledge.

This procedure is valid for both the user knowledge but also the global knowledge containing all the global entities like streets, buildings, among others. It the latter case the situation can be facilitated by the fact that the open data should usually be published according to a specific format but this is not always the case.

8.3 Data Storage

The data storage element of the Data Acquisition and Management Subsystem of the SB is composed by two different database systems where all the data are stored, as showed in Figure 8.4. The reason for the division lies in the different nature of the data:

- **Entity data storage.** We defined that the knowledge about the user, that is composed by all the elements she considers relevant, are represented following the entity-centric approach. In this thesis we did not focus on the aspects related to the knowledge representation since we leveraged on the previous work from the members of the Knowdive Group[3] as presented in Section 3. We said that this knowledge in terms of entities and relations among them is stored in a system called the Entity Base (EB) system. In Section 8.3.1 we will present some additional details, in particular about the interaction between this component with the other components of the SB.

- **Streaming data storage.** This is the core of the Data Acquisition and Management Subsystem. It is the place where all the streaming data referring to the user are stored. We designed it to store two different types of data: the streams of sensor data collected from the user smartphone and the streams of entity attribute values computed during the knowledge generation phase. It consists of a distributed database system with an application layer on top of it developed to insert and extract the data.

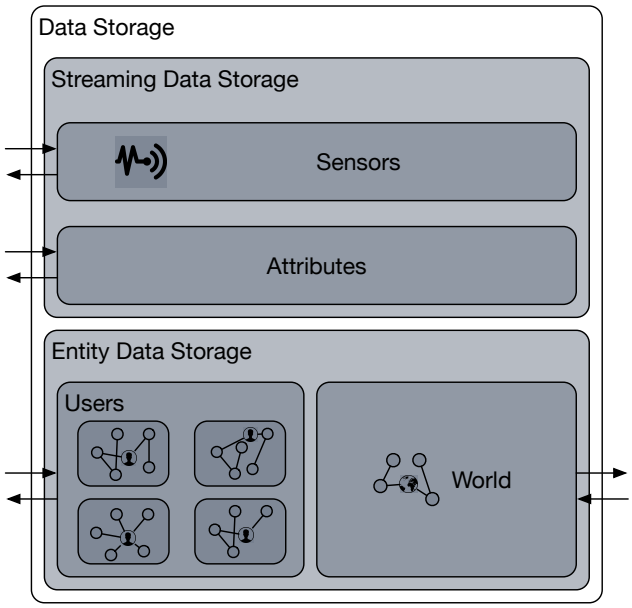


FIGURE 8.4: Schematic presenting the two storage systems of the SB.

In the next sections we describe the characteristics of the two mentioned data storages. Since the knowledge modeling and storage aspects are outside the scope of this thesis, we did not develop our own solution for the entity storage. On the other hand, we used the EB system which is a component developed by the Knowdive Group. For this reason, in Section 8.3.1 we just describe at high level how the system works, mainly focusing on its content related to the data sources. On the other hand, in Section 8.3.2 we describe the streaming data storage that we developed, integrating the information previously illustrated in the methodology. Specific considerations are technology dependent and are left to Chapter 11.

[3] http://disi.unitn.it/~knowdive

8.3.1 Entity Data Storage

As anticipated in Chapter 3, the Entity Base (EB) System is responsible for storing and maintaining the knowledge, represented using the entity-centric approach. All the elements stored in it are represented as entities with their attributes and links that allow to create a network of entities, namely the knowledge. This system has been developed by the Knowdive Group members in the past years and is currently used as reference technology to manage the knowledge by all its members in the vertical solutions.

In the context of this thesis the main piece of knowledge refers to the user personal representation of the world. Then, depending on the situations, the entities that are relevant for the user in a specific situation are enabled to compose the user context. Since the system has to deal with multiple users, every one with her own knowledge, we needed to use the EB system in a particular configuration to allow all the data to be separated for privacy reasons. Additionally, there is also the need to have some knowledge that is shared among all the users. This knowledge refers to the world and is composed by streets, buildings, public services and anything that can help the system in solving the problem at the core of this thesis.

User Entity Data Storage. Usually when designing a system, the data separation can be one of the requirements. And the data separation can happen according to different criterias: type of entities, based on the users, on groups of users, among others. Multiple reasons can be at the base of this decision but usually the most important one relates to privacy. Since the user knowledge should consists in the personal representation of the world of its owner, we can consider it sensitive data that can be accessed only by the user itself. There are two possibilities to separate the data in a storage system:

- **Physical separation.** To physically separate the data into different databases. This means, we will need N instances of the EB system, one for every user. This is the best solution from a privacy point of view since the databases are physically different, most likely even on different machines.
- **Logical separation.** To logically separate the data. In this second solution the database is unique, there is one single instance of the Eb system that contains the data of all the users.

Starting from the assumption that the system performances are the same in the two situations above, we don't have any preference for the SB, both can be used.

World Entity Data Storage. This knowledge is useful especially for computing the services, where data not referring to the users is needed. The entities inside this EB system should refer to a more objective view of the world which is not personalized on the user. They should refer to locations, services, objects, streets that are part of cities, municipalities, states. These are the entities the users that use the SB should interact with but at the same time entities that are not **relevant** for the user, so that they are not stored in their personal knowledges.

These data should come from external actors that have interest in the users using their data. A great example is Open Government Data. Recently always more institutions are starting to produce open data about their areas of competences, both at the global level (i.e., European Union[4]) but also at the local level (i.e., Italy[5] or

[4] https://data.europa.eu
[5] http://www.dati.gov.it/

the Municipality of Trento[6]). Open data are those data that are freely accessible by anyone and can be used without restrictions (except the citation of the source) for both non-commercial but also commercial uses. The core concept behind open government data is that the public administrations should be open to the citizens and transparent about the decisions they take. Moreover, these data can be of great value for the companies that can avoid investing a part of their profits in building these datasets.

If these data are imported into this knowledge base and all the users in the SB can use it, this will highly facilitate the exploitation of the knowledge for creating useful services. Consider for example a scenario in which the user wants to move around the city using public transportation. Having a detailed snapshot of what is the situation of public transport in the city can enable personalized solutions based on the user context. From the user personal knowledge the system knows she has to reach a destination for an event. From the world knowledge on the other hand the system knows all the possible means the user can use to reach this destination, considering all the variables. At this point, by merging the two, a useful service can be created. The service is useful for the user, because she can obtain what she wanted, but is useful also for the Municipality who provided the data, because it is reducing the number of cars in the city, which is a long term goal they have.

In this thesis this second knowledge will be stored in a separated EB system. All the users in the SB will be able to access its content, without any restriction. From a privacy point of view this EB will be exempted: in fact, the data stored in it are public and don;t have any privacy restriction.

8.3.2 Streaming Data Storage

A stream of data is defined as a continuous flow of information that is made available over time. In contrast with non-streaming data, new values are always produced and must be added to the ones that have already been generated. For this specific type of data, represented as described in Section 4.2.0.1, we have a dedicated component. It contains data that can be classified into two types:

- The streams of sensor data collected from the user's smartphone
- The streams of entity attributes values generated by the system out of the sensor streams

This system component is constituted by a single database system that stores both data. Since they have very different characteristics, we decided to logically separating them inside the database.

Streams of Sensor Data. Previously in this thesis we referred to this type of data as personal big data: they have huge size, they are generated very fast and they are of different types. Solutions to store such information must be addressed if we want the SB to be able to generate knowledge out of them and ultimately solve the semantic gap problem.

Not all the database technologies can deal with such huge amounts of data like the one generated by the users and managed by this component of the SB. A general consideration we needed address in the design phase was that standard SQL databases do not scale well when dealing with big data and most importantly, it

[6] http://dati.trentino.it/

is not easy to scale them up when the users increase. Then, we believe a NOSQL database technology should be used. This type of databases provide a way to store and retrieve the data that is modeled in means other than the tabular relations used in relational databases. The motivations behind this shift in the paradigm include: the simplicity of design, a simpler horizontal scaling to clusters composed by multiple machines and the possibility to tune the availability of the data base on the use case. Since the data structure is different and with less contraints, some operations are faster with respect to SQL. Many of the different NOSQL technologies compromise consistency (as defined in the CAP theorem) in favor of availability, partition tolerance, and speed. Additionally, most NoSQL stores lack of ACID transactions. On the other hand, most NoSQL databases offer a concept of "eventual consistency" in which database changes are propagated to all nodes "eventually" (typically within milliseconds) so queries for data might not return updated data immediately or might result in reading data that is not accurate. In the case of streaming data such in this thesis, represented as time series, if some of the values are lost does not create any issue.

Streams of Attribute Values. The knowledge generation process in the SB consists in updating the attribute values of the entities, using the streaming sensor data in combination with other contextual information. Since the attributes represent an abstraction of the sensor data, their size will be much smaller with respect to them. For example, one hour of streaming sensor data collected by the accelerometer from the user smartphone can be translated into the Attribute Value (V) "Walking" of the Attribute Definition (AD) "Movement". We generated knowledge out of the streaming data with the resulting knowledge mapped to a concept that is much more meaningful to the user with respect to the sensor values collected in the hour. At the next change in the movement of the user, the knowledge in the EB system will be updated and since it contains only a snapshot of the most recent situation (the context) the previous data will be lost. To avoid this and keep track of all the attribute values that can be useful for the historical analysis, we took the design decision of storing this information into the Streaming Data Storage component of the SB.

8.4 Summary

In this Chapter we went into the details of the Data Acquisition and Management Subsystem, presenting its main components, that deal with the acquisition, importing and storage of all the data represented in the system. We didn't go into the details since the application should be general and able to be adapted to all the use cases and technologies.

We presented the different sources of the data, both streaming and knowledge. The former being considered as personal big data, collected from the user's smartphone and the latter modeled using the entity-centric approach, generated by the user manually or helped by the machine. Additionally we said that other knowledge can be pushed into the system from external sources, for example to model cities, streets and other elements that will help in the description of the surroundings of the user.

We then presented the two data import pipelines that will be used to import both data sources into the SB. These pipelines present some general characteristics but is then up to the specific use case to define specific pipelines, depending on the data they have to import.

Finally, we presented the data storage systems of the SB, being them the Streaming Data Storage that has to scale well when the load increases and the Entity Data Storage, for which we used the Entity Base (EB) System, developed by the members of the Knowdive Group.

Chapter 9

Knowledge Generation Subsystem

From an architectural point of view the Knowledge Generation Subsystem is composed by four components, as shown in Figure 9.1, that are meant to apply the methodology defined in Chapter 5. The Knowledge Instantiation and the Knowledge Update are two logically separated blocks since they perform two different tasks, where the former requires the user feedback. There is then a Repository component where all the procedures defined by the field expert for the Attribute Definitions (AD) are stored. This repository is shared among all the users and when one of them has to use one of the procedures, it can get and instantiate it in his own Knowledge Generation component. Finally, all the knowledge generation tasks are triggered by a dedicated Operation Scheduler on a (i) regular basis by a timer, or (ii) according to external conditions, like an user input, a change in the context, among others.

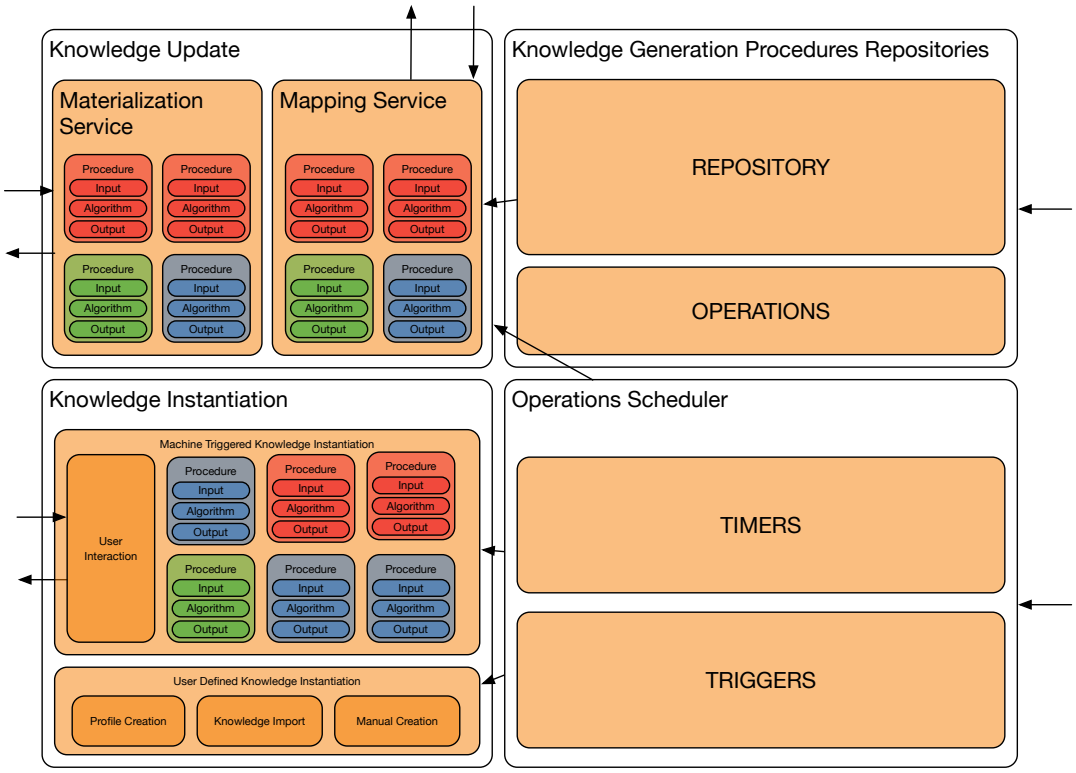


FIGURE 9.1: Schematic of the components of the Knowledge Generation Subsystem

9.1 Knowledge Generation Procedures Repository

As explained in the methodology developed in this thesis and presented in Section 5, the process of generating knowledge is divided into two tasks:

- **Knowledge Intantiation.** This steps refers to the ability of adding or removing entity instances from the use knowledge database. This database at the beginning is empty and progressively must be filled up with the entities that the user believes are relevant to represent her context.
- **Knowledge Update.** The user context is dynamic, since it is used to represent the situations in which the user is involved. There is then the need to update the entities in this snapshot of the user knowledge. Doing so corresponds to updating the attributes of these entities depending on the sensor data collected from the user's smartphone.

For both the Knowledge Instantiation and Knowledge Update tasks the machine automatically performs the analysis, thanks to the procedures developed by the field expert, according to the changes in the user context. Our methodology associates a Procedure to every Attribute Definition (AD) that needs to be updated. The procedure defines which input must be used to produce the output, at the different levels, both sensor streaming data (and their features) and the user knowledge represented as attribute values. Moreover, it defines also the algorithm that must be used to analyze the inputs to generate the desired output.

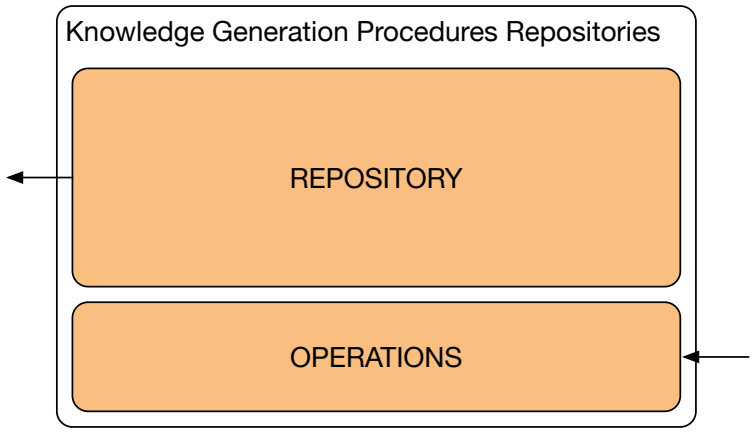


FIGURE 9.2: Schematic of the Knowledge Generation Procedures Repository component.

All the procedures, one per attribute definition, once created are stored into the Knowledge Generation Procedures Repository in the SB, which is a globally shared repository among all the users. A schematic of it can be seen in Figure 9.2. Its objectives are twofold:

- Create a unique place where all the procedures can be created, updated, tested and deployed;
- Create a way of letting the users instantiate these procedures in their respective local systems when an Entity with an Attribute Definition for which a procedure is defined is created. This allows for reusability of the procedures and helps in scaling the system and the number of users.

When the knowledge of a certain type is created, i.e., an Entity with a specific attribute associated with an attribute definition, the corresponding procedure is searched in the repository. If present, the procedure is instantiated in the SB, either in the Knowledge Update or the Knowledge Instantiation components. If not present, a field expert is required to create one. Every time the procedure must be used to update or instantiate an Entity in the user contextual information (stored in the EB system), the corresponding procedure is executed.

The repository provides different functionalities that help to *plan* the implementation of certain procedures, facilitating the collaboration between field experts. It can be the case that a procedure is fairly complex and an entire team needs to work on it, not a single person. The generated procedure can then be tested and bugs can be reported. Finally, a stable procedure can be released in production so that the users can use it and this allows also to manage updates. These are all functionalities of standard solutions, e.g., GitLab[1] we will use for this component.

The procedure creation by the field expert happens through the Operations module of the Knowledge Generation Procedures Repository. It provides some abstractions and default methods to read and access the input data from both the storage systems, the Streaming and the Entity and allows to push the results to the corresponding attribute. It provides also the authentication mechanism for the field expert using the components described in Sections 10.1 and 10.3

9.2 Knowledge Instantiation

The Knowledge Instantiation component is the one that takes care of filling up the user personal database (EB) with the entity instances the user believes are important for her context. A schematic of the component is shown in Figure 9.3.

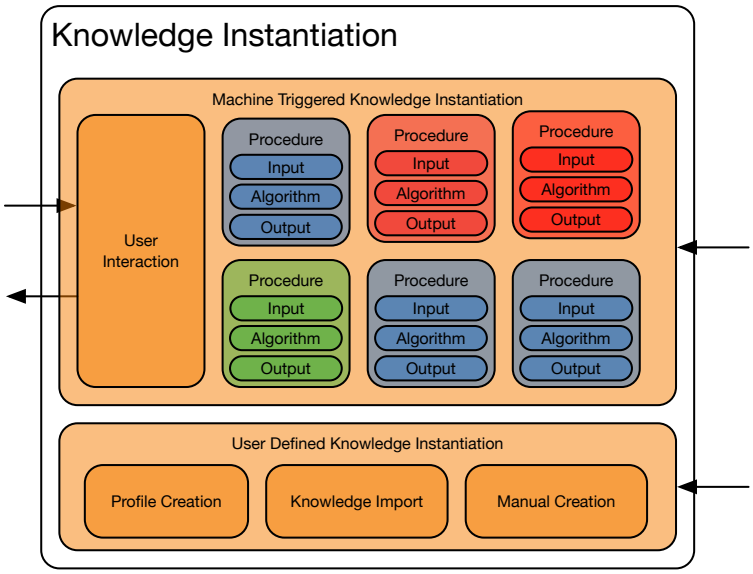


FIGURE 9.3: Schematic of the Knowledge Instantiation component.

Since every user has her own view of the world, the entities that must be instantiated are different. For this reason, this component cannot be shared among all the

[1] https://about.gitlab.com/

users, like others in the SB. There are no constraints about how to obtain the necessary separation across users, if physically, meaning that every user needs her own instance of the component, or if this can be done logically, both can be applied. In the schematic of Figure 9.3, the division is logical and every color maps to one user: user1 represented in red has two procedures instantiated, user2 in blue has three procedures and user3 has only one.

As said in Section 5.2, the instantiation of the knowledge can happen in two ways, both requiring the user interaction: the user defined knowledge instantiation and the machine triggered knowledge instantiation.

9.2.1 User Defined Knowledge Instantiation

The user can help the machine in producing more accurate results by generating her own knowledge manually. As we said in the methodology, it can be done while she *creates her profile* on the platform, providing personal information, details about her locations or additional data useful for the services she wants to use. On the other hand, the user can also *import the knowledge* from other sources like the contact list, the agenda, among others. Finally, the user can *manually create* the knowledge using a dedicated tool. In all these situations, the only procedure that needs to be applied is the conversion of the origin format to the entity-centric model. The whole pipeline in this case can be used by multiple users in parallel and the generated data will be saved in the user entity base. For each of the three methods, an application or module must be created: for the user profile generation, an online procedure must be enabled so that the user can introduce her own information and the same is for the manual creation. On the other hand, for the importing, dedicated modules must be used, one for every type of application or at least of data.

9.2.2 Machine Triggered Knowledge Instantiation

In Section 5.2 we said that the machine can trigger the knowledge generation and the user is asked only to confirm the entity instances the machine generated.

This process uses Procedures similar to the one of the knowledge update task, to analyze the streaming data collected from the user's smartphone and generate entity instances. These procedures are generated by the field expert and are stored in the Knowledge Generation Procedures Repository. The user can instantiate one or more of them, depending on the entities she wants to create. Once a Procedure is instantiated, it starts working in the background in the user "personal" Knowledge Instantiation component. It continously analyzes the streams of data collected from the user's smartphone and every time a pattern is detected, a confirmation task is generated. The task asks the user to corroborate the findings at the most appropriate moment. The separation among the users allows them to have different procedures running: a user can be most interested in locations, while another in people, and so on. In Figure 9.3 we can see that the blue user has three active procedures, while the red one has two and so on.

For example, a Procedure that infers where the user home is, continuously analysis the GPS data comeing from the user's smartphone with a clustering algorithm, i.e., DBSCAN, and generates some clustered locations made by multiple points. These locations will be then presented to the user and her feedback will be asked, through a dedicated component in the system. Once the feedback has been provided, the newly created entity is added to the user personal representation of the world. The user interaction part takes into consideration all the elements defined in

the methodology in order to define the best moment when to ask to the user so that to not bother her.

9.3 Knowledge Update

The Knowledge Update component is the one responsible for updating the attribute values of the entity instances in the user personal knowledge and context. It implements the methodology defined in Section 4.2.0.1. A schematic of this component can be seen in Figure 9.4.

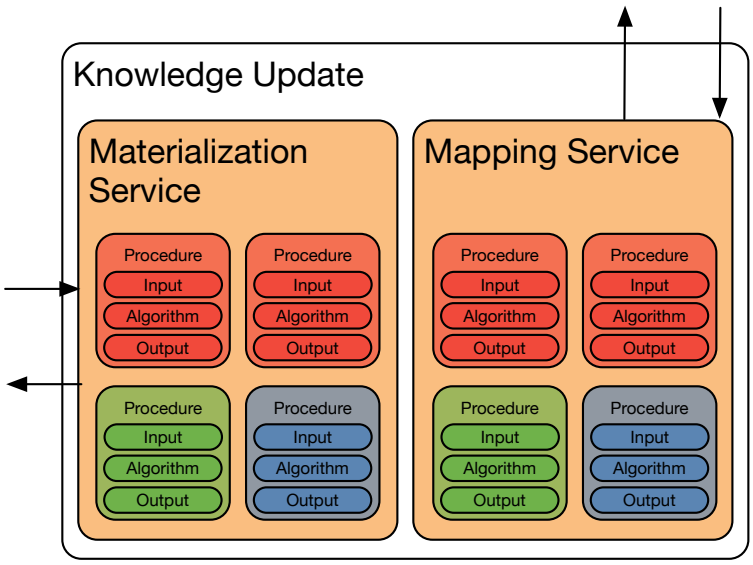


FIGURE 9.4: Schematic of the Knowledge Update component.

This component is composed by two elements, the Materialization and the Mapping services, that are very similar one with respect to the other. They are containers of instantiated Procedures taken from the repository component. The procedures are the one defined by the field expert to let the machine automatically update the attribute values of the entities. As we presented in Section 4.2.0.1 there are two types of streaming data in the SB: the first one are the data collected from the user's smarpthone, while the second one refers to the attribute values that keeps updating and then generate a stream. The entity storage (EB) contains only a snapshot of the user personal view of the world, the context, meaning that when a new attribute value is generated, the old one is overwritten. To keep track of the attribute values changes, once they are generated, they are *mapped* into the corresponding stream into the Streaming Data Storage. Then, when the update needs to be reflected to the Entity Base and then to the context, the value is *materialized*. The two operations are left separated in order to have the maximum flexibility but they can also be executed at the same time.

Also in this case as for the Knowledge Instantiation component, the user Procedures must be kept separated, either physically or logically.

9.3.1 Knowledge Mapping

The Knowledge Mapping process is the one that uses the Procedures as defined by our methodology and that actually generates the updated attribute value. The newly

generated attributes values are mapped to the corresponding stream in the Streaming Data Storage since we want to keep track of the values changes. We cannot directly store them into the Entity Base (EB) system since it contains only a snapshot of the most updated view of the world of the user. This means that the newly generated values overwrite the old ones.

This component is a container for all the active Procedure for every user and allows to run them as a process to update the attribute values when needed.

9.3.2 Knowledge Materialization

The Knowledge Materialization consists in taking the attribute values from the Streaming Data Storage and materialize them in the Entity Base (EB) system so that they can be used by the other components of the SB that need contextual information. We left the Mapping and the Materialization procedures independent so that to obtain the maximum flexibility.

Complex strategies can be defined so that to perform a smart materialization that is not simply a 1-to-1 process. The operation that are allowed at the moment are the following:

- Average;
- Maximum;
- Minimum;
- User defined operation.

All these operations can be performed on the last N attribute values generated or on the values generated in a range of time.

9.4 Operations Scheduler

The Operations Scheduler is the component that schedules all the Knowledge Instantiation and Update tasks.

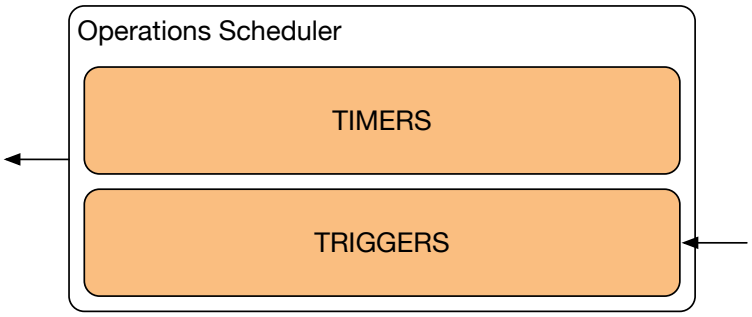


FIGURE 9.5: Schematic of the OperationScheduler component.

As can be seen form Figure 9.5, there are two ways the knowledge can be generated:

- **Timers.** The timer allows to perform operations on a regular basis the user or an external service defined. For example, a service that generates a quantified self report can be executed during the night, on the data of the previous

day, so that in the morning the user can visualize it and see information of the day before. The timer can be arbitrarily complex, allowing to do any possible combination with exceptions.

- **Triggers.** On the other hand, some external inputs can trigger the knowledge generation process, like the user input or a change to any value of the entities in the context.

9.5 Summary

In this Chapter we presented the Knowledge Generation Subsystem of the SB. We defined is as composed by four main components: the knowledge update component, the knowledge instantiation component, the knowledge generation procedures repository and the operations scheduler.

As we said in the methodology, the knowledge generation consists of two operations, i.e., the instantiation of new knowledge or the update of already instantiated one. For both, a dedicated component has been designed so that to separate the two tasks. Moreover, every instantiation and update is kept separated for the different users in the system since the context is different for every user.

We then described where all the Procedures created by the field expert can be searched and retrieved to be used to generate the knowledge.

Finally, we explained how the knowledge generation can be triggered, using a timer that schedules the operations at fixed time intervals or using external triggers.

Chapter 10

Knowledge Exploitation Subsystem

The role of the Knowledge Exploitation Subsystem is to use the data to provide services to the users in order to improve their quality of life. In this section we will describe all the components of such a system, that have to tackle the privacy related issues in retrieving the data from the storage systems. These components are shared among different subsystems in the SB, i.e., the Data Acquisition and Management Subsystem but we decided to present it here since the privacy related issues are more relevant when using the data. Since the privacy related aspects are not at the core of this thesis, we just describe them at high level, representing the most important elements. The components can be logically grouped into two groups: the first one composed by the blocks that make the data available to the services according to the different privacy dimensions while the second one composed by the blocks that actually provide the services. A schematic of the components of the Knowledge Exploitation Subsystem can be seen in Figure 10.1.

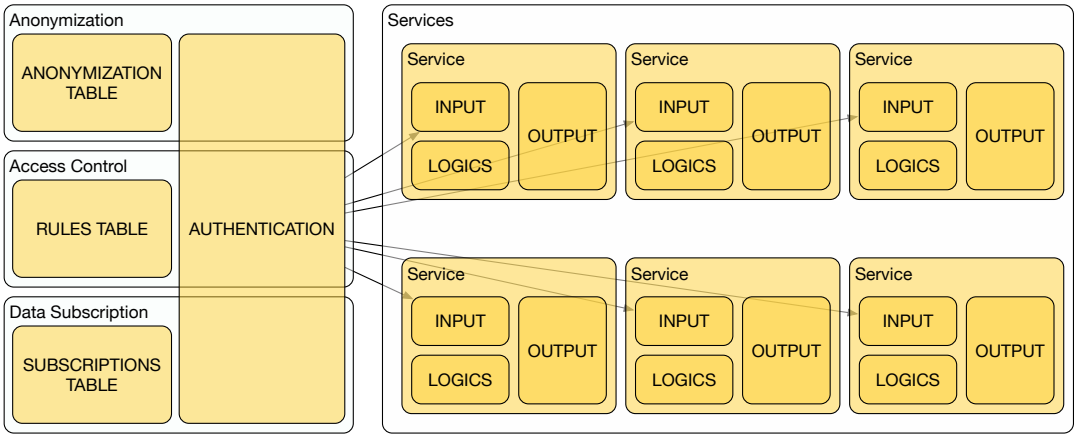


FIGURE 10.1: Schematic presenting the Knowledge Exploitation Subsystem and its components.

More in details, we have the following components: an authentication module, an anonymization module, an access control module, a data subscription module and finally the module that provides the services. The architecture, designed in this way, is compliant with the latest regulations introduced by the Directive 95/46/EC, otherwised called General Data Protection Regulation (GDPR)[1] introduced by the European Union that will start applying in May 2018.

[1] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC

10.1 Authentication

Every system requires the user to authenticate to access its services. In the SB we have an authentication module that is shared among all the components in the system. The authentication requires the user to communicate her credentials, username and password. If they match with the one present in the system, the user is considered authenticated and an acknowledgment is sent back to the user.

The authentication is the first protection mechanism in the SB to ensure data protection of the user. Used in combination with the other elements, i.e., anonymization, access control and data subscription makes it very difficult for an ill-intentioned to access the user data.

Since it is shared among different elements, it is also the one that directly makes available the other privacy related functionalities (anonymization, access control and data subscription) to the other components of the system.

From an architectural point of view there isn't much to say about authentication since the details are usually related to the technology and the specific solutions used to implement it.

10.2 Anonymization

In the SB we have to deal with user personal and sensitive information and the privacy is a key element one must take into account. Among all the privacy related aspects the most important one is to guarantee the right protection of the user data. In other words, the system has to take all the possible counter measures to prevent external actors who don't have the rights, to access and use the users' data. A part from the security measures, one way to guarantee this is to store the user data in an anonymized way. This is done to prevent others that will take possession of the user data due to a data leak to be able to use them and lead back to the users themselves.

The SB has a specific component that deals with the anonymization and deanonymization of the user data and it is shown in Figure 10.2.

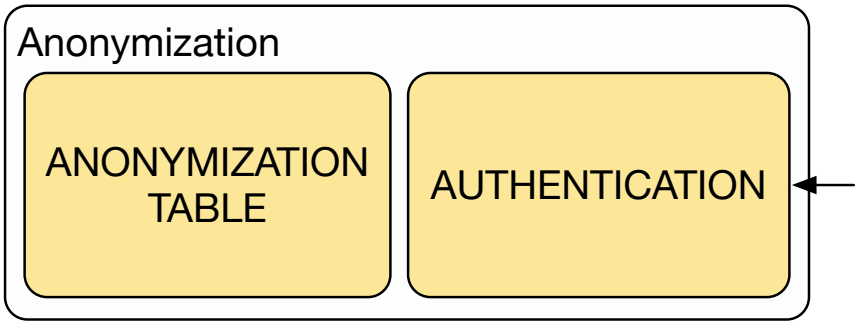


FIGURE 10.2: Schematic presenting the elements of the Anonymization component of the Knowledge Exploitation Subsystem.

The data in the SB is stored anonymously as explained in Section 4.2.0.3. The way to anonymize the user data is to store and associated them with a unique identifier that refers to the user. This identifier is a 40bytes string that is generated when the user account is created. When another component needs to store or retrieve data about one specific user, it needs this unique identifier that must be requested. When a component needs to access the users' data, it contacts the Anonymization component that, after the request has been authenticated, uses the user credentials to

lookup in the *anonymization table*. The table itself is very simple, composed by only two columns, *username* and *uniqueidentifier*. Given the importance of this table, that is the only way to lead back from the data to their owners, we decided to encrypt it using one of the strongest encryption algorithm: the Advanced Encryption Standard (AES) [Standard, 2001]. It is a standard used by the US Government to encrypt their document. In particular we choose the most secure version with the largest key, 256bits.

10.3 Access Control

Another key element in granting the users privacy are the access control policies based on roles. In the context of this thesis, there are different roles, i.e., administrator, developer, field expert, user, etc. These roles are used to access and control the different components of the system. Not all the roles can perform the same operations for example, the user cannot register other users, while the field expert can create algorithms for the knowledge generation task but cannot add new components to the system. The Access Control component allows to manage the permissions to perform such operations based on the the different roles. It is composed by a *rules table*, as shown in Figure 10.3 that associates the permissions with the corresponding roles. The roles are associated with the user credentials. When a new request comes associated with one user requiring to access some information, the user is searched in the internal table. If found, the access request is compared and if matched, the confirmation is sent back, otherwise the access is denied.

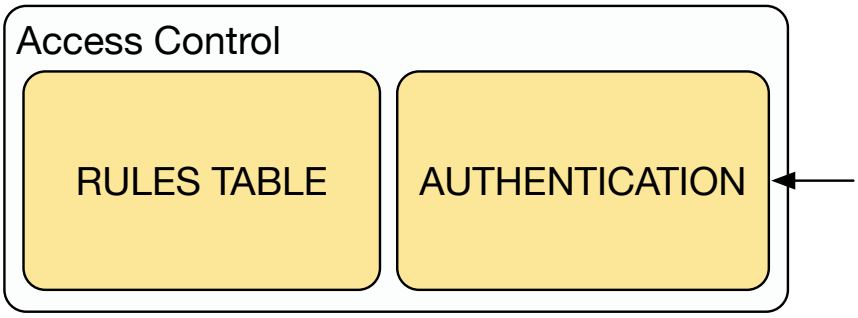


FIGURE 10.3: Schematic presenting the elements of the Access Control component of the Knowledge Exploitation Subsystem.

The access control policies **do not regard** the user data. The user data can be accessed *only by the user herself* and no other role can access them without the user permission. In particular, the management of the permissions the user grants work according to a publish/subscribe mechanism with a dedicated component in the system.

10.4 Data Subscription

As mentioned in the previous Section, only the user can access her own data, neither the administrator nor any other role can access them. Of course, in order to have personalized services, each service needs to access the user data. The access is added through a data subscription action made by the user. The are *Service Providers* that create services that can be used by the users in the platform. These providers can be internal to the SB or also external i.e., the University, the Municipality, among others.

Once the services are published, the user can enter a user interface that allows to see the services and subscribe to them. With every subscription some permissions to access the data are requested and the user must accept them.

The Data Subscription component of the Knowledge Exploitation Subsystem allows to manage the subscriptions by keeping track of all of them. A Figure of the component is shown in Figure 10.4: it shares the authentication component with the other components and then has a *subscription table* where all the subscriptions, with the expiration dates and other information, are stored.

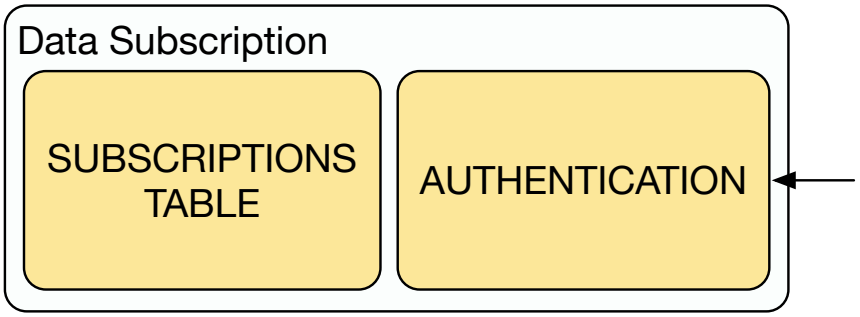


FIGURE 10.4: Schematic presenting the elements of the Data Subscription component of the Knowledge Exploitation Subsystem.

When a service needs to access some personal data, this component is queried to check if that specific service has the right to access the user data, and, if yes, which data he can query. At that point the request is forwarded to the database.

10.5 Services

The core part of the Knowledge Exploitation Subsystem if the part that provides the services to the users. As shown in Figure 10.5, multiple services can be executed in parallel.

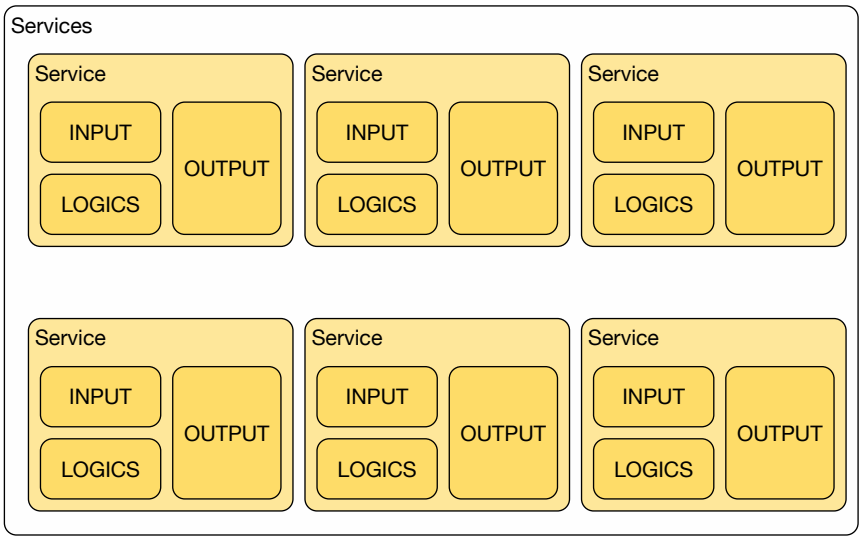


FIGURE 10.5: Schematic presenting the Services component of the Knowledge Exploitation Subsystem.

A service is composed by three main elements: the data that they must use, the logic that defines how the data is analyzed and processed and how it must be combined to show the output to the user. The input part interacts with all the other components of the Knowledge Exploitation Subsystem to have access to the data. How the interaction works is described as follows:

- When a service needs to produce an output for the user it contacts the Data Subscription module to check if it has an active subscription to read the data;
- If the first step succeeds, then the service has all the permissions to read the user data. The additional step required to do so is to have the correct unique identifier for the user so that to deanonymize her data. This information is stored in the Anonymization component;
- At this point the service can contact the storage system and read the data for the user.

The logic of each service, the algorithms and procedures that analyze the data to produce the output are use case dependent. There can be services in the context of smart cities or health related services, among others. The services are developed by the SB or by external actors that want to use the platform to reach the users. These actors can be other institutions like the University, the Municipality, the National Health System or private companies as well. The more the actors the more the services and the more the user has the chance to find the most suitable ones for her situations. Similarly, also the output of the service is use case dependent: one service can produce a notification system that alerts the user about events on her smartphone or it can create a dedicaed user interface for the web.

10.5.1 System Services

Among the different services that the actors can create to help the users there are also some services that are created by the SB to allow the user to configure different system functionalities. These services, will interact with the Access Control component. Some basic services we already developed (or are in the process to be developed) are the one that follow.

10.5.1.1 Data Control

As previously said, due to the latest EU regulations about personal data protection i.e., GDPR, the user must always be in control of her own data. To accomplish this we designed a system service that allows her to always look what is the situation of her data in the SB. This allows to:

- See how many data the system contains about her, both from the knowledge but also the streaming point of view;
- Allows to access and browse the data. This functionality is more crucial on the knowledge, since the streams of data have little meaning for the user, they will be only a lot of numbers for her;
- Delete pieces of knowledge or streaming data.

Before the user can do this she must authenticate with her credentials so that all the security checks are made, namely access control, and deanonymization.

10.5.1.2 Publish/Subscribe Mechanism

Similarly to the previous service, this one allows the user to control the publish/subscribe mechanism in the system. It allows to visualize the subscriptions, to subscribe to new ones and to delete unwanted ones. Every permission is associated with an expiration data since a permission can be granted only for a limited amount of time. With every subscription it is possible to visualize which data the service can access, how frequently it can do that and other information. A mockup of this service was developed for one of the use cases and can be seen in Figure 10.6.

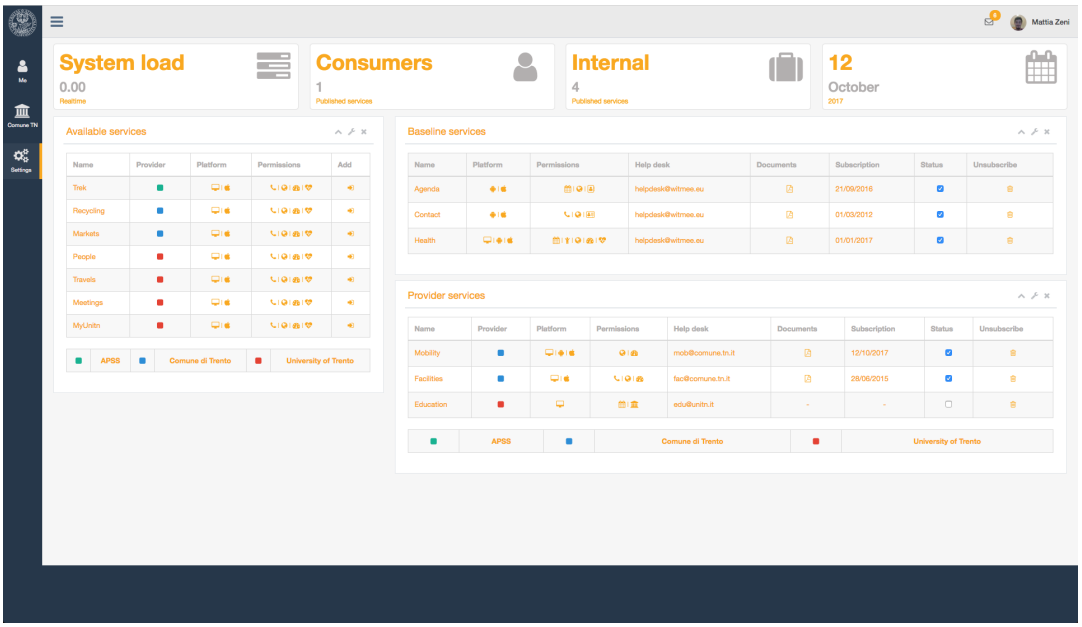


FIGURE 10.6: Mockup of the Publish/Subscribe service of the SB.

In the screenshot we can see in the left table what are the services made available by others on the platform that the user can choose from. In the two tables on the right on the other hand we can see which are the services he subscribed to.

10.5.1.3 Users Registration

The user registration service has been created to allow users to register and start using the platform autonomously. As said in Section 5.2.1, when the user registers, she creates her own profile and the Entity Base (EB) is filled up with such information. There is the need to balance the need for data with the need to keep the procedure simple. In fact, we cannot afford to loose the user because we asked too many information during the registration phase. The procedure is very simple and allows first to create username and password and later provide all the additional data. When the user registers, the components of the Knowledge Exploitation Subsystem share the data so that it is possible to create the anonymization id used to store the user data. This unique identifier is stored in the Anonymization Component with the username and is the only way the data can be anonymized when strictly necessary to provide services to the user.

10.5.2 External Services

A part from the internal system services, the way we designed the SB allows to third-party entities to produce their own services that can be used by the users. If the user decides to subscribe, her data is shared with the service provider as long as the user allows this; she is always in control and can decide to quit at any moment. These services can be distributed by institutions like the National Health System, the Municipalities or the Universities but also by private companies.

Let's consider a simple example of service that can be provided by the University of Trento to its students. With this service the university wants to understand which study rooms are more crowded than others so that to help decision makers in finding a solution to this problem. This has direct impact on the way the single student lives this aspect of the university life. In fact, if the university has knowledge on the current occupancy of its rooms, by aggregating the data coming from the single students, it can create a service that can suggest where to go to find a free spot. At the same time, if enough users use the same service, it can use the aggregated data collected form them to take decisions like changing the opening hours of the library, rather than building a new library since the once available have reached the maximum capacity.

A similar service can be provided by the Municipality of Trento to study the impact of public transportation on the citizens to re-schedule the timetables of the buses, among others.

The publish/subscribe mechanism implemented in the SB enables such a scenario and allows the services to be created by multiple actors that can really use the generated data to improve the users' quality of life in different areas. The privacy aspects have been also tackled since, by design, the user is always in control of her own data.

10.6 Summary

In this Chapter we presented how the Knowledge Exploitation Subsystem of the SB uses the internally generated knowledge to provide services to the users. It is composed by two main elements that perform two different tasks. The first one is related with the privacy of the users while the second one is the one that provides the services.

All the privacy related aspects are tackled in all the architecture, not only in this subsystem but we decided to present them here since they are more relevant when the data is accessed and used. We presented our approach based on authentication, anonymization, access control and data subscription. All of them are security measured at the backbone of the SB that make it compliant by design with the latest EU regulations, i.e., GDPR. The data subscription aspects are very important for the services, since the user must always be aware of how others use her own data.

The services can be provided by the system itself (internal services) but also from external entities like institutions or private companies. The user must subscribe to every service accepting the conditions for sharing her data. The platform allows for multiple services being provided at the same time to multiple users.

Chapter 11

The StreamBase (SB) System

In the previous chapters we have presented a methodology we have defined to solve the semantic gap problem in the context of this thesis, how to collect the data from the users, how to use them to generate useful knowledge and finally how to exploit this knowledge to provide useful services to the user. We then have presented a reference architecture for the SB that is based on that methodology.

In this chapter we present how we instantiated the reference architecture into a real working prototype of the StreamBase (SB) System, we describe the technical solutions and technologies we identified to fulfill the requirements defined in Section 7.1. Finally, we present how the SB can be quickly and easily deployed in the different use cases that will use its functionalities.

The key element of the whole SB is its modularity. As we explicitly explained during the thesis, most of the components instantiate algorithms and/or databases. To support these allocations and deallocations of resources, specific technologies must be taken into account to facilitate the integration and allow to automate this process as much as possible.

11.1 Modular Architecture Based on Microservices

To meet the specifications previously mentioned about the SB, state of the art software solutions must be used for its implementation. This is particularly true for the modularity aspect that regards the instantiation and termination of computational procedures and or databases (or part of them). Additionally, this is needed also for the highly variable workload expected for the system, that depends on the number of users connected simultaneously. This modularity should be cheap in terms of time but also costs. For this reason, is it not feasible to depend on a system architecture that runs directly on Physical Machine (PM). A computer system that runs directly on the hardware is the most efficient one since it is highly optimized, and the operating system directly runs on top of it. Then, on top of the operating system, the applications are executed. The first computers were running only in this configuration, but also today it is possible to find very performant systems that are still like this. Starting from the 1960s the first virtualization solutions start to be used and were designed to logically divide the resources provided by powerful mainframe computers between different applications. The idea behind a Virtual Machine (VM) is to virtualize and simulate the hardware and then install the operating system on top of it. This allows to install multiple instances of an operating system on the same powerful machine. This is the most common used configuration today in datacenters.

Neither Physical Machine (PM) architectures, nor Virtual Machine (VM) ones are suitable for the SB since the requirements in terms of modularity and scalability go beyond what these two solutions can provide. More in details, the modularity of the former is both time consuming and expensive in terms of money while for the latter the time required is the main bottleneck.

Recently, a new approach has started to gain consensus in the computer science community, the so called *Container Architecture* (CA) built using *Microservices*. The container architecture derives directly from the virtual machines concept of having a virtualized environment above the hardware of a computer. Differently from a virtual machine though, each container is a virtual space where to execute one application in resource-isolated processes. This means that, instead of having an entire operating system in each virtual space, each container is able to perform only few operations. This is why they are built of Microservices. Each application can be packed into easy to use building blocks, providing all its code, dependencies and configurations. With such a configuration it is easy to obtain environmental consistency, operational efficiency, developer productivity, and version control. Such containerized applications give also an additional level of control over resources with an improvement in the final efficiency. This can happen on one single physical machine but also among multiple physical machines with additional tools. This means that the scalability of this solution is virtually unlimited. It is easy to move one container from one machine to the other, without the need of reconfiguring the environment every time a new deployment is done. Moreover, the deployment of a new container can be automatized based on a set of rules. Consider a system like the one we are describing in this thesis where multiple data sources can, asynchronously, write or read huge amounts of data to a server. In this case, a containerized application can duplicate its instances automatically depending on the number of request arriving at any specific moment in time.

11.1.1 Microservices

For more then 100 years, our business markets have been about creating products and driving consumers to wanting those products. *Today, everything is about services*. Consider the following companies: Uber[1] is the number one private hire company in the world, and it has no cars; Flixbus[2] is a brand which offers intercity bus service in Europe, and it owns no buses; finally, Airbnb[3] is the worldwide leader in hospitality service, and it has no real estate. All of them provide a service to their customers, that is, connecting demand and offer through their online platforms. They build software solutions, both desktop and mobile, to create communities of users.

A Microservice Architecture (MSA) is a new approach of building software systems that decomposes business domain models into smaller, consistent blocks implemented by services [Posta, 2016]. These services are isolated and autonomous and communicate one to the other to provide the required funcitonalities. They usually have enough autonomy that each of them can change its internal implementation with minimal impact across the rest of the system. When a new microservice is designed, all the specifications to use the service must be produced as well, otherwise it is useless. They usually comprehend API and documentation, plus others.

[1] https://www.uber.com/
[2] https://www.flixbus.com/
[3] https://www.airbnb.com/

Each service must implement the right technology with the appropriate programming language to solve the problem it is designed for. The boundaries for a service can be elicited as follows:

- Understand what is the task the service has to solve without considering other concerns related to the large application;
- Quickly build the service in a local environment;
- Pick the right technology and programming language for the problem;
- Once implemented, test the service;
- Build/deploy/release at a cadence necessary for the business;
- Identify and horizontally scale parts of the architecture where needed;
- Improve resiliency of the system as a whole.

Services allow to cancel the synchronization costs. However, they have also a drawback if used like we just explained, that is, it can be more *resource intensive*.

The concept of microservices maps 1-to-1 with the SB as presented in the previous sections. Many times we referred to *instantiating a knowledge generation procedure*, *instantiating a service for the user*, *instantiating the database*. All these elements are seen as a microservice (or multiple if the final element is too complex to fit in one single microservice) that runs independently from all the rest of the architecture.

11.1.2 Docker

As said before, microservices nicely work with the Container Architecture paradigm, and Docker[4] is the world's leading software containerization platform. It allows to package an application with all of the dependencies it needs, i.e., OS, JVM, libraries, in a lightweight, layered, image format. These images are then used to run the applications inside a Linux container with isolated CPU, memory, network and disk usage. We can see these containers as a form of *application virtualization* or even *process virtualization*. This means that we can get more applications running on a single set of hardware for higher density without the overhead of an additional operating system like for virtual machines [Merkel, 2014; Turnbull, 2014].

A standard Virtual Machine (VM) emulates the hardware of the Physical Machine (PM) it is installed on. When you launch a VM and run a program that hits disk, its generally talking to a "virtual" disk. When you run a CPU-intensive task, those CPU commands need to be translated to something the host CPU understands. All these abstractions come at a cost: two disk layers, two network layers, two processor schedulers, even two whole operating systems that need to be loaded into memory. These limitations typically mean you can only run a few virtual machines on a given piece of hardware before you start to see an unpleasant amount of overhead and churn. On the other hand, you can theoretically run hundreds of Docker containers on the same host machine without issue.

The machine hosting the Docker containers has its own infrastructure and operating system. Instead of the hypervisor (like in Virtual Machines), it has the Docker Engine installed, and this is what interacts with the containers. Each container holds

[4] https://www.docker.com/

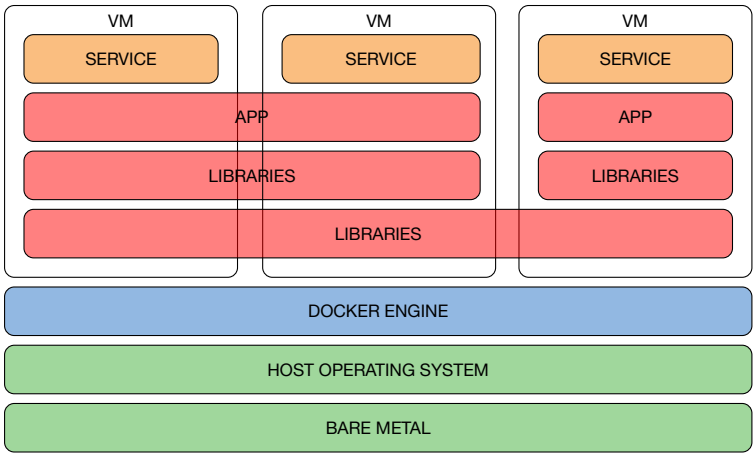


FIGURE 11.1: A representation of how Docker works.

its application and the required binaries, libraries and other dependencies, as seen in Figure 11.1. It is important to note that they don't require their own guest operating system. This allows the containers to be significantly smaller in size, and able to be distributed, deployed and started in a fraction of the time taken by virtual machines.

11.1.3 Kubernetes

Kubernetes[5] is an open-source system for automating deployment, scaling and management of containerized applications. It allows to create a cluster of physical machines on which the containers can be deployed, automatically. It was developed by Google Inc. and released as an open source project beck in 2013. Google is known to run all its applications using the container architecture paradigm at scale, since they run more that two billions container per week. Kubernetes brings a lot of functionality for running clusters of microservices inside Linux containers at scale.

Kubernetes orchestrates, schedules and manages *pods*, instead of containers. One or more containers are grouped into a *pod*. Among the different characteristics of a pod, the most important one is that all the containers inside the pod run on the same physical machine. Moreover, all containers in a pod share the same IP address and Volume. Additionally, pods, like containers, can be destroyed at any time and then it is important to make them stateless.

Since the microservices we are referring to must solve scalability issues, this means that multiple instances of the same microservice will be running at the same time. Kubernetes allows this and, most importantly, allows the containers to scale automatically, thanks to the *ReplicationController* that manages the number of replicas for a given microservice.

11.2 Distributed Database System

From a technological points of view, out of all the different databases able to manage big data (as reported in Section 14.5) we decided to adopt Apache Cassandra as main storage for the SB, specifically for the streams of data generated by the user's smartphone. As reported in [Wlodarczyk, 2012], Cassandra in a good choice for storing time series data, in particular for its elastic scalability and eventual consistency features. In this section we illustrate what is Cassandra, what are its most

[5] https://kubernetes.io/

important features and how we used it to store the streams collected from the user's smartphone.

11.2.1 What is Cassandra

Among all the available definitions we found about Cassandra, the one that we like more is the one in [Carpenter and Hewitt, 2016]:

"Apache Cassandra is an open source, distributed, decentralized, elastically scalable, highly available, fault-tolerant, tuneably consistent, column-oriented database that bases its distribution design on Amazon's Dynamo and its data model on Google's Bigtable. Created at Facebook, it is now used at some of the most popular sites on the Web."

Cassandra is a *distributed* database system, meaning that it is capable of running on multiple machines while appearing to the final user as a unique database. This means that every node stores only a part of the data and this helps when the load starts to grow since every single machine has to manage only some data. Additionally, Cassandra is also *decentralized* meaning that all these nodes are the same, there is no master-slave distinction. There are no nodes that perform organizing tasks distinct from any other and this element gives a great contribution in reliability of the whole cluster since there is no single point of failure. This aspect is called server symmetry because all the nodes are the same and there is not a special host that if fails will cause the whole cluster to fail and this is the key to Cassandra's high availability. Scalability is an architectural feature of any system that can continue to reply to a greater number of requests with little degradation in performance. There are two types of scalability:

- **Vertical:** it usually refers to systems that are not distributed and then that run on a single machine. To vertically scale means that by adding hardware to this machine, it is still able to continue serving requests without being affected by performance degradation. With hardware usually they refer to CPU and RAM, but also disk to store more data. This is the easiest solution, that can be achieved in less time but that is also very expensive and ideally has an upper bound limit.
- **Horizontal:** this is typical of distributed systems, and refers to the ability of adding smaller, less performant machines to an existing cluster so that no one machine has to bear the entire burden of serving requests. Of course, dedicated software solutions must be used to keep the data in sync and avoid conflicts.

Cassandra goes beyond these two concepts and introduces the *elastic scalability*. It is referred to as a special property of horizontal scalability and means that the cluster can scale up and back down seamlessly. When a new node is added to the cluster and starts participating, there should be a mechanism to send him a copy of some or all of the data and start serving new requests from the users without major disruption or reconfiguration of the whole cluster. All the balancing and queries are automatically managed by the cluster itself, there is no need to change the applications that use the server. When removing a node because of a failure or any other reason it works the same, a simple command can be triggered on the node and after some time the cluster balances itself to receive its data and the new node can be removed. The availability of a system is usually measured according to its ability to fulfill requests, but computers can have all sorts of failures, hardware failures, network failures, power outages, among others that will prevent the system from being

available. A system to be *highly available* and *fault tolerant* it must include multiple networked computers and of course the software running on them must be capable of operating in a cluster and recognizing failures. In this regard, Cassandra is highly available and fault tolerant because it can replace a failing node with no downtime. A database system to be consistent means that a read operation always returns the most recently written value. Of course, scaling up is not free and some trade-off must be made between **data consistency**, **node availability** and **partition tolerance**. The CAP theorem [Gilbert and Lynch, 2002] refers to the three characteristics just mentioned and says that only two of them can be achieved at the same time in a distributed system. In Cassandra the application can *tune the consistency* level, but in general it is more on the trade-off between availability and partition tolerance, meaning that the system may return inaccurate data but it is always available, replies fast and is always possible to write. We believe this is the best approach for time series data: it is important to write them all, and there is no "rush" in reading them and even if some values are skipped, it is not a big deal since we collect hundreds of values per second. Cassandra is is not relational, and it represents its data structures in sparse multidimensional hashtables where sparse means that for any given row you can have one or more columns, but each row doesn't need to have all the same columns as other rows like it. Each row has a unique key that makes the data accessible. We can think at Cassandra as an indexed, row-oriented store. Cassandra has been designed for *high performances* meaning that it can scale consistently and seamlessly to hundreds of terabytes.

Despite Cassandra's design and features, it is not the right tool for every job. First of all, it is meant to operate in large deployments. None of its qualities like high availability, tuneable consistency, peer-to-peer protocol is meaningful in a single-node cluster. Second of all, Cassandra is optimized for excellent throughput on writes with less predictable read operations. The ability to handle application workloads that require high performance at significant write volumes with many concurrent client threads is one of the primary feature of Cassandra and in particular this suites well for time series where a lot of data is continuously written.

11.2.2 How Cassandra Stores the Data

Cassandra's data model is an excellent fit for handling data in sequence regardless of datatype or size. When writing data to Cassandra, data is written sequentially to disk. When retrieving data by row key and then by range, you get a fast and efficient access pattern due to minimal disk seeks – time series data is an excellent fit for this type of pattern.

We refer to the Datastax documentation[6] to explain how time series are stored in Cassandra, to motivate the modelling choices we made in Section 11.2.4. The simplest model for storing time series data is creating a wide row of data. The timestamp of the reading will be the column name and the temperature the column value. Since each column is dynamic, the row will grow as needed to accommodate the data. In some cases, the amount of data gathered isn't practical to fit onto a single row. Cassandra can store up to 2 billion columns per row, but if we're storing data every millisecond you wouldn't even get a month's worth of data. The solution is to use a pattern called row partitioning by adding data to the row key to limit the amount of columns you get per device. Using data already available in the event, we can use

[6] https://academy.datastax.com/resources/getting-started-time-series-data-modeling

the date portion of the timestamp. This will basically produce one row per day, and an easy way to find the data. This representation is showed in Figure 11.2.

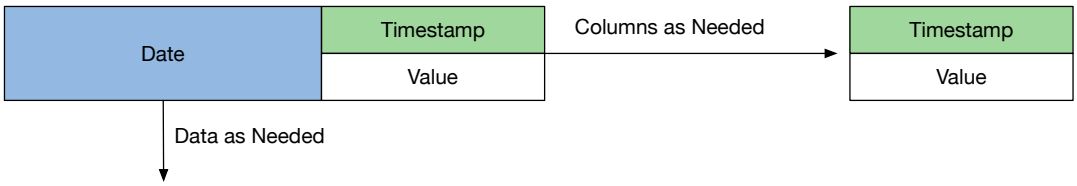


FIGURE 11.2: A representation of how Cassandra stores timeseries data.

11.2.3 Querying a Cassandra Node

Querying a Cassandra cluster is an operation that is performed using the Cassandra Query Language (CQL). The way Cassandra is structured allows to perform only certain types of query, that do not affect performances. In other words, a query is allowed only if takes a reasonable amount of time to complete and return results, otherwise is not allowed at all. The main reason is related to the very particular data model Cassandra uses and if this seems a bottleneck in using the system because requires a lot of time in the design phase, it simplifies the queries and improves the performances. Only fast queries are allowed in Cassandra and if a query is slow, then there is a mistake in the way the developer modelled the data. Queries in CQL are done using the SELECT and WHERE clauses. These constructs allow to select and filter the results of the query according to some conditions the user specifies in the query itself. Another important aspect about queries is the fact that they are strictly linked to the PRIMARY KEY which is composed by the PARTITION KEY and the CLUSTERING KEY. With these considerations in mind let's list some of the rules[7] that must be taken into considerations while querying Cassandra:

- Range queries (<=>) are not allowed on the partitioning key
- Restrictions on the partitioning keys are allowed. We must restrict all keys or none.
- An alternative to the restriction of all the keys is to restrict only some of them and use ALLOW FILTERING. This practice is a performance killer.
- In general, every query can involve only those columns that are part of the primary key.
- There is the possibility to override rule 4 in the sense that we can query on additional columns that use secondary indexes. Also in this case as for rule 3 this practice is a performance killer.

11.2.4 Cassandra Data Model for Streams

In the SB Cassandra is used as a persistence system that stores the logs collected from multiple sensors for multiple users. The amount of data and the amount of writes per second are very high and the system has to be carefully designed. Our application collects data continuously from up to 30 sensors for each user at a rate

[7] http://www.datastax.com/dev/blog/a-deep-look-to-the-cql-where-clause

of 20 readings per second (up to 350 per second). At the lowest collection rate the application can easily collect more than 30 million values per day per user. Usually the synchronization occurs at night, when the phone is plugged and there is wireless connectivity. For this reason, writes are not distributed across the whole day but are focused in a short period of time (burst) and this corresponds to more load for the server. Details about the performances of the system are presented in Section 11.2.5. With these considerations in mind let present the data model carefully designed for this system.

In order to preserve users' privacy, we adopted a complete separation between users using one keyspace per user. The keyspace name is a specific unique identifier for the user which cannot lead back to the username and in particular is a salt. In cryptography, a salt is random data that is used as an additional input to a one-way function that hashes a password or passphrase. The primary function of salts is to defend against dictionary attacks versus a list of password hashes and against pre-computed rainbow table attacks. We generate this salt when the user registers to the system using the SHA-1 algorithm. Since our application is user centric, which means that every user has access only to his data and is not aware of others' data, this division using one keyspace per user is the right choice. Moving forward in the explanation of the data model, we decided to create one table per sensor in the user keyspace. In this way we have a logical separation of the sensors in the database without affecting the performances. Each table has a specific data schema according to the values we have to store and the queries we have to perform. Although there are many sensors, some of them share the same schema because they collect the same values and then we need to perform the same queries on them. The first set of tables are the ones related to the 3-values sensors, namely those that collect data according to the 3-axes X, Y, Z. The data schema is presented in Table 11.1 and is valid for the sensors: accelerometer, gravity, gyroscope, linear acceleration, magnetic field, orientation and rotation.

TABLE 11.1: Schema for the 3-axes sensors.

Key Name	**Day**	**Timestamp**	**X**	**Y**	**Z**	**Primary Key**
Data Type	text	text	float	float	float	((day), timestamp)

The next set of tables are those related to the single-value sensors, namely those sensors that collect only one variable. These sensors are: temperature, proximity, light, pressure and humidity. As explained in the previous Sections Cassandra encourages data denormalization and duplication because usually disk space is very cheap while time during queries is very expensive. To satisfy all the queries for our application we need to duplicate these tables and create 2 tables (11.2 and 11.3) with two different data schema (primary key) for each sensor.

TABLE 11.2: Schema for the sensors that generate only one value, modeled to allow query based on the timestamp.

Key Name	**Day**	**Timestamp**	**Value**	**Primary Key**
Data Type	text	text	float	((day), timestamp)

Similarly, for the Bluetooth and Bluetooth LE sensors we have to create 3 tables according to the schemas presented in Tables 11.4, 11.5 and 11.6. This is necessary to allow, respectively, query by address, by timestamp and by signal intensity (distance).

TABLE 11.3: Schema for the sensors that generate only one value, modeled to allow query based on the value.

Key Name	Day	Timestamp	Value	Primary Key
Data Type	text	text	float	((day), value)

TABLE 11.4: Schema for the Bluetooth and Bluetooth LE, modeled to allow query based on the address.

Key Name	Day	Timest.	Name	Address	Bond State	Rssi	Primary Key
Data Type	text	text	text	text	text	float	((day, address), timestamp)

TABLE 11.5: Schema for the Bluetooth and Bluetooth LE, modeled to allow query based on the timestamp.

Key Name	Day	Timest.	Name	Address	Bond State	Rssi	Primary Key
Data Type	text	text	text	text	text	float	((day), timestamp)

TABLE 11.6: Schema for the Bluetooth and Bluetooth LE, modeled to allow query based on the signal strength (RSSI).

Key Name	Day	Timest.	Name	Address	Bond State	Rssi	Primary Key
Data Type	text	text	text	text	text	float	((day), RSSI)

The last set of tables are the one dedicated to the user location. In particular, we have two sensors on the mobile device that allow to acquire the location of the user: the network location and the GPS. The former is a way to triangulate the user that uses information from the GSM network or the WIFI network. These two approached allow to extract only the latitude and the longitude with an accuracy value that ranges from $30m$ to $3km$. The latter instead uses a dedicated GPS sensor inside the smartphone that is far more accurate than the previous method and allows to acquire other very important information. More in details, the GPS can acquire the user's latitude, longitude, altitude, speed, bearing with an accuracy that can go below to $3m$. By default, Cassandra doesn't have direct support for Geolocalization and this means that we have to design the schema to be very efficient while querying. We created a custom type in the database called Point which is composed by latitude, longitude and altitude

$$Point < latitude, longitude, altitude > \tag{11.1}$$

where this point type is the basic element for the data schema we designed. We identified one single table to store these data as shown in table 11.7. With this table, we

can natively support per timestamp queries, while we need an external tool to support geographical queries. This tool is called Stratio's Cassandra Lucene Index[8] and leverages on the indexing capabilities of Cassandra to index the location points so that geographical queries can be performed, such as "Give me all the Points stored in the database that are in a radius of X meters from the location Point<latitude, longitude, altitude>". What Stratio does is basically to provide near real time search such as ElasticSearch or Solr, including full text search capabilities and free multivariable, geospatial and bitemporal search.

TABLE 11.7: Schema for the Location data, modeled to allow query based on the timestamp.

Key Name	**Day**	**Timest.**	**Acc.**	**Bearing**	**Point**	**Provider**	**Speed**	**Primary Key**
Data Type	text	text	float	double	frozen <point>	text	float	((day), timestamp)

11.2.5 Performances

As reported in Section 11.2.1, Cassandra has been designed to take full advantage of multiprocessor/multicore machines, and run across many dozens of machines housed in multiple datacenter, always operating with high performances. In particular, the best performances are during the write phase since Cassandra is optimized for excellent throughput on writes.

In this section we present the performances result in terms of throughput and load we obtained during our test with the schema used by the SB. We have two separate test clusters:

- The first cluster is composed by 5 nodes, where every node is a virtual machine with remarkable resources: 8 2.2GHz processors, 96GB of RAM, 1Tb of disk hosted in a NAS, connected to the machine through an optical fiber connection.
- The second cluster is composed by only one node, with the same characteristics as the first one.

We used the second cluster to test the throughput we could obtain from Cassandra, using the "cassandra-stress" tool shipped with the database. We run both write and read tests, which results are respectively shown in Figures 11.3 and 11.4. In these tests, we used one single node, 913 threads writing/reading simultaneously for a total of 1 million values.

The results are summarized also in Table 11.8. As it is possible to see, the writes are 21.16% faster than the reads in terms of throughput, while the result for the latency are not clear towards one or the other.

We also tested how much data a Cassandra cluster can store without affecting the performances. For this test, we wrote random data modeled according to the schema described in Section 11.2.4 on the first test cluster. We let it run at the maximum speed for two months, writing at $80000[op/s]$, and we ended up having a total size of $1.79TBytes$ of data. With these data stored in the database, we could not see any concrete performance degradation.

[8] https://github.com/Stratio/cassandra-lucene-index

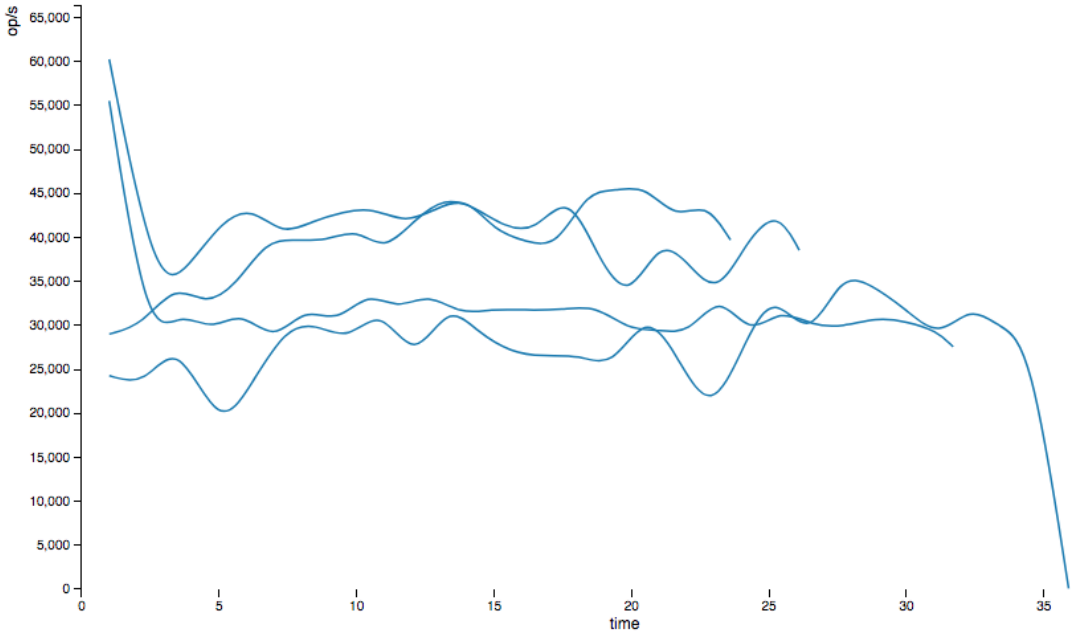


FIGURE 11.3: Graphic showing the Cassandra reading performances for the configuration used in the SB with one node, 913 threads reading 1 million values.

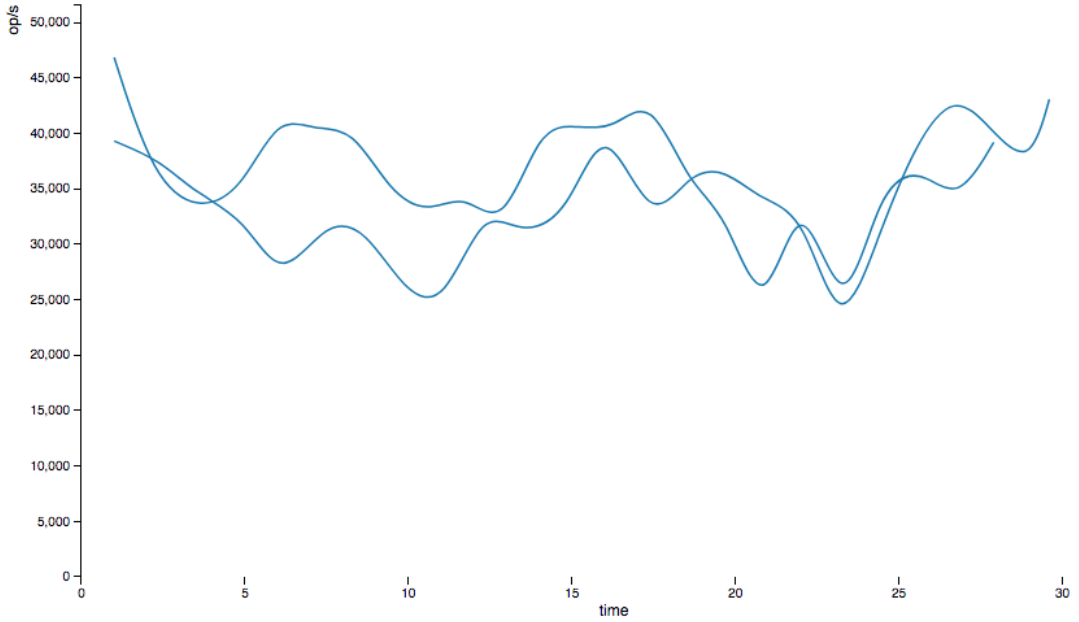


FIGURE 11.4: Graphic showing the Cassandra writing performances for the configuration used in the SB with one node, 913 threads writing 1 million values.

11.3 Framework for Distributed Computing

In Computer Science, the size of the data keeps rising and this generates the need to change the way these data are processed and managed, as individual processors clock speed evolution slowed down and system evolved to a multi-processor oriented architecture. There are already scenarios where the data size is too big to be analyzed in acceptable time by a single system, and in this cases is where Distributed

TABLE 11.8: Cassandra performances summarized for the SB deployment with one single node, 913 threads and 1 million records.

	Operations [op/s]	**Latency [ms]**					
	Mean	**Mean**	**Median**	**95th perc.**	**99th perc.**	**99.9th perc.**	**Max**
Write	38282	24.1	17.0	55.4	163.1	251.8	418.1
Read	31594	29.7	23.4	74.3	113.0	196.7	368.5

Computing Solutions are able to shine. Today there are different solutions that allow to do this, the most important and consolidated one is Apache Hadoop[9] which is designed to efficiently distribute large amounts of work and data across multiple systems. however, in this thesis we decided to adopt Apache Spark[10] which is a data parallel general-purpose batch-processing engine. The main reason of this choice is that it is much faster than Hadoop since it works in memory.

In the SB we use it for performing heavy calculations such as to speed up the data input process, or to apply machine learning techniques on the data, among others. The fact that the heavy tasks happens on the Apache Spark cluster (which usually is executed on separate machines) let us have a tighter control on the single components of the application, since they are lightweight and we can better manage the local resources.

11.3.1 What is Apache Spark

The simplest but also more complete definition we found about Apache Spark is the one in [Karau et al., 2015]:

Apache Spark is an open-source cluster-computing framework which provides an interface for programming entire clusters with implicit data parallelism, fault-tolerance, speed and generality.

Spark extends the popular MapReduce [Dean and Ghemawat, 2008] model to efficiently support more types of computations, including stream processing. One of the main feature it offers for *speed* is the ability to run computations in memory, even if it is more efficient with respect to MapReduce even on disk. On the *generality* side, Spark is designed to cover a wide range of workloads that are divided into categories such as streaming, batch applications, machine learning, and since all of them are integrated in the same engine, it is easy and inexpensive to combine different processing types which is often a requirement in production data analysis pipelines. More in details, at its core, Apache Spark is a computational engine that is responsible for scheduling, distributing, and monitoring applications consisting of multiple tasks across many workers in a computing cluster. Since, the core engine is fast and general purpose, it powers multiple higher-level components that are specialized on different tasks. These components are designed to interoperate closely so that an application can easily combine their functionalities. This tight integration brings different advantages: first of all, optimizations made in the core, are reflected in all the components; secondly, the cost of deploy, maintain, test and support stacks

[9] http://hadoop.apache.org/
[10] http://spark.apache.org/

of applications is reduced with respect to single ones; finally, when a new component is added to the stack, every organization that uses Spark will immediately be able to use it as well.

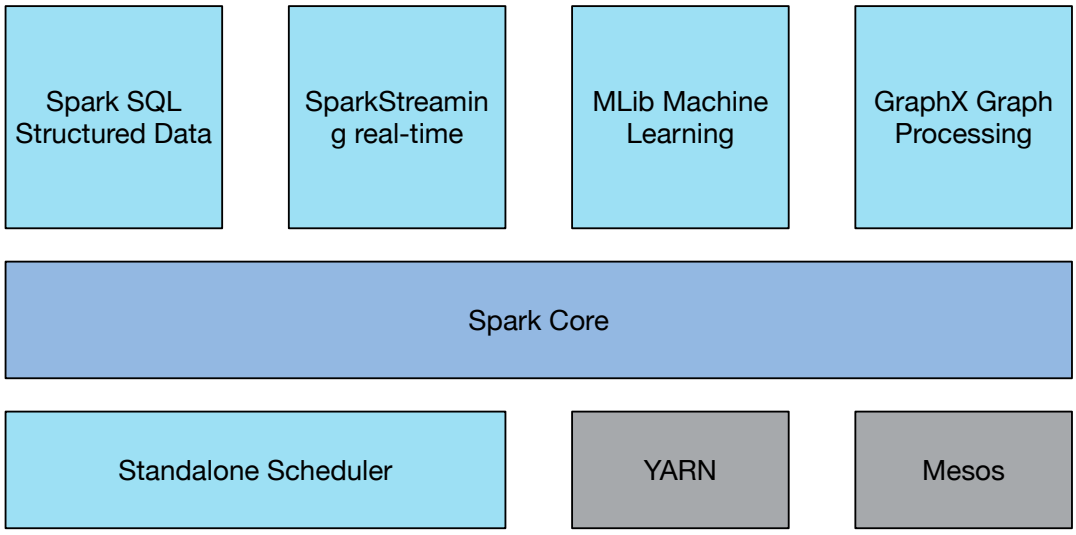


FIGURE 11.5: Schematic showing the components of the Apache Spark Stack.

What follows is a brief introduction to the Spark components, as shown in Figure 11.5:

- **Spark Core.** It contains the basic functionalities of Spark, scheduling, memory management, fault recovery, interacting with the storage systems, and more.
- **Spark SQL.** This package allows to work with structured data. It enables queries via SQL but allows to include also other sources, such as Apache Parquet[11] or JSON[12] files. A part from providing the interface, this component allows also to intermix SQL queries with the programmatic data manipulations supported by the different languages for complex analytics.
- **Spark Streaming.** It enables processing of live streams of data. API are provided that closely match the Spark Core's RDD API.
- **MLib.** It is a library that contains common machine learning (ML) algorithms such as classification, regression, clustering, as well as model evaluation and data import.
- **GraphX.** Is a library for manipulating graphs and performing graph-parallel computations.
- **Cluster Managers.** Similarly to Apache Cassandra, Spark is designed to efficiently scale up from one to many thousands of workers. To achieve this and improve the flexibility, it allows to use different cluster managers, including Hadoop YARN[13], Apache Mesos[14], and a simpler ad-hoc solution.

[11] https://parquet.apache.org/
[12] http://www.json.org/
[13] https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html
[14] http://mesos.apache.org/

11.3.2 Resilient Distributed Datasets (RDD)

The Spark's core abstraction for working with data is the Resilient Distributed Dataset (RDD) which can be simply defined as an immutable distributed collection of elements. They are the core elements of the Spark framework, in fact, all the work can either create new RDDs, transform existing ones or produce results from them. The data contained in RDDs is automatically split into partitions and distributed across the cluster to parallelize the operations performed on them.

RDDs can be created in two ways: by *loading* an external dataset or by *distributing* a collection of objects. Once created, two possible operations are possible on them: *transformations* and *actions*. The former creates a new RDD from an existing one for example by filtering the data that match a predicate. The latter, compute a result based on an RDD and either return it to the driver program or save it to an external storage system, e.g., HDFS. It is important to notice that Spark computes RDD in a lazy fashion, meaning that is happens the first time they are used in an action. Consider the example in which an application reads a 1TB text file and stores it in a RDD. Immediately after this, it asks to perform a task only for the first line in the file. If Spark waits to read all the triggered commands, it can optimize its operations (this is easier to understand when multiple commands are run in series). In the example just mentioned, Spark waits for the command to read the first line to execute the load operation. Actually, it never loads the whole file but on the other hand loads only the first line. Another characteristics is that RDDs are recomputed each time an application runs an action on them but the RDD is stored in memory (distributing it in the nodes) after the first action.

11.4 Instantiating the StreamBase (SB) System

The technological decisions we took for instantiating the reference architecture in the SB are made to make it modular and easy to develop, maintain and deploy. In fact, our idea is to have one instance of the system for every project we will carry on, both for research but also industrial applications.

Docker[15] containers orchestrated by Kubernetes[16] are at the core of the system and define its architecture. We have a computer cluster managed by Kubernetes that will deploy containers on demand, depending on the configurations and on the system load.

Every element of the system will be developed as a Docker container: this will provide the modularity and scalability required by the SB. A component, as presented in the architecture, can be a unique container or a set of multiple running containers. For example, the Anonymization component is one container that will manege all the requests by the other components, while the Knowledge Update component will be logically composed by multiple container, one per Procedure. In the description of the system, when we referred to "instantiation" of an element, we meant that the corresponding Docker container will be created and executed, managed by Kubernetes.

Also the Cassandra cluster and the Apache Spark cluster will be created using docker containers so that to automatically scale up and down when required. In the case of Cassandra the situation is a bit more complicated because we must take into account the persistance of the data: as we explained in Section 11.1.3, the Docker

[15] https://www.docker.com/
[16] https://kubernetes.io/

containers orchestrated by Kubernetes are dynamic, meaning that they can be deployed and deleted without notice. This cannot be tolerated in the case of a database system such as Cassandra, where the data must be persisted. For this reason, we created dedicated volumes that are detached from the containers as such. A **Cassandra Node (CN)** is composed by a **Power Resource (PR)** and the **Data (D)**, defined as follows:

- **Data (D).** This is the disk used to store data for the experiments. The usable disk space should be considered to be $1/4$ of the total hard drive space on the physical machine. $1/2$ of the space is used by the backup configuration (best option will be RAID 10) and the additional $1/2$ is used by Cassandra to compact the data (See Cassandra Compaction Strategies for more information). This means that if we want to be able to store 1TB of data we need a physical disk of 4TB;
- **Power Resources (PR).** RAM and CPU used by the Cassandra process(es) to consume the data, where consuming means writing to disk while collecting or reading from disk to do analytics and reply to queries
- **Cassandra Node (CN).** It is a Docker container composed by 1 unit of D and 1 unit of PR attached to it as a mounted Volume, where D and PR are logically decoupled. The minimum requirements for one CN instance are:

  PR:

  - at least 16GB of RAM (32GB best option);
  - 4 CPU cores (8 best option).

  D:

  - One node can easily handle 1TB of data, we cannot overcome this limit. Considering the requirements on D as above, each CN needs up to 4TB of data

At this point, the Cassandra Database for a use case will be composed by multiple Cassandra Nodes (CN).

Moving to the deployment of the architecture for the different use case, the technical solutions we adopted are used to facilitate the procedure: with Kubernetes we can define a set of containers, i.e., Pod that can be deployed with a single command an this pod will contain all the elements needed for the experiment i.e., a Cassandra Cluster, an Apache Cluster, and all the components of the SB.

11.5 Summary

In this section we described how we implemented the reference architecture into a working prototype of the SB. More in details, we described which are the main software solutions and technologies we used. We described own the implementation is based on Docker containers, orchestrated by Kubernetes. This solution brings huge advantages in terms of modularity, and scalability which is automatically managed by Kubernetes. The containers can be replicated when the load requires it so that to be able to satisfy all the requests.

We then presented the technology we adopted for the database system, Apache Cassandra. It is a distributed database system able to scale to terabytes of data by

simply adding machines to the cluster. This feature is strictly related with Docker containers and Kubernetes. In order to let Cassandra store streams of sensor data, some consideration have been made and a data model has been presented for the scope. We also tested Cassandra on our test clusters to see what are its performances under heavy loads.

Finally, we presented the computational framework we leverage on, Apache Spark. It is a distributed application that allows to schedule works that are automatically distributed on a cluster of machines, with a mechanism similar to Mpa Reduce. Using this solution allows to keep the system components simple and easy to manage, while delegating all the huge workloads to this cluster.

Part IV

Use cases

Chapter 12

Knowdive Experiments

The Knowdive experiments are a set of internal use cases done on the members of the Knowdive Group[1] at the Department of Information Engineering and Computer Science of the University of Trento. Their main aim is to test the solutions developed in this thesis before the real deployment in the wild. These experiments are easier to carry out since they usually involve less people, which already know how to use the mobile application given their computer scientist background and have an additional motivational factor of helping the group that we can leverage on.

The experiments done so far were two, one dedicated to debugging the general functionalities of the i-Log application and SteamBase system in general to prepare the real life deployment on the university students (SmartUnitn One). The second, on the other hand, was focused on the test of more specific features, such as the audio collection and the marketplace deployment.

12.1 Knowdive One

Knowdive One is the first real-life test of the methodologies and solutions described in this thesis to see if the implemented SB could scale well with multiple users. We decided to instantiate the system and use the i-Log mobile application to collect data using as users the members of the Knowdive Group at the Department of Information Engineering and Computer Science at the University of Trento.

The most important outcome of this first experiment was to test the i-Log mobile application and the backend infrastructure of the SB in terms of data collection and storage and in particular to quantify the impact the data collection has on the smartphone battery life.

12.1.1 Objectives

The objective of this first trial was to debug the whole architecture, both the mobile client and the backend infrastructure and understand if the distributed solutions we adopted were capable of managing an increasing number of users and scale. In particular, we mainly focused on the data collection and management aspects since they were never tested before outside a laboratory setting. Additionally, this internal use case was designed to quantify the impact on the battery life of the whole data collection process for the upcoming experiment SmartUnitn One.

12.1.2 Smartphone Battery Consumption

As it is well known in the state of the art, using the smartphone as a sensing device has some advantages but also many disadvantages. The first and most important

[1] http://disi.unitn.it/~knowdive

one concerns the battery life of the device [Ferreira, Dey, and Kostakos, 2011]. A smartphone is designed to be used for an entire day, but this means that when the screen is off, the applications are generally either not supposed to run or do only little work, thus allowing the smartphone to preserve energy. On the other hand, when the smartphone is used to collect data about the user, an application is always running in the background and this increases the energy demand, making it difficult to cover en entire day. For i-Log the situation is the same, and with this first use case we had the chance to quantify the phenomena and try to reduce its effects.

The device chosen for the battery measurements is a Samsung Galaxy S4 (GT-I9500) since it has an above average number of sensors among the currently available phones on the market. We can divide these sensors into two classes: the *hardware sensors* that are real sensors used to collect data at a certain *sampling frequency*. There are 9 sensors belonging to this category on this device: Magnetic Field (MF), Temperature (T), Humidity (H), Pressure (P), Accelerometer (A), Gyroscope (GY), Orientation (O), Gravity (GR) and Light (L). The sensors belonging to the second class are called *radio sensors* because they all use a radio signal, and they are: Location Sensors (GPS or NET), Wi-Fi, Bluetooth (BT or BTLE) and the Microphone (MIC). They cannot be considered sensors *per se* because they are not designed with the purpose of collecting data. They are smartphone components with different purposes in the system that we decided to use in a way that allowed us to collect user's personal data. Since they are not real sensor, there is no real sampling frequency but we had to define an equivalent parameter to perform the analysis. We then defined the *sampling interval* as the interval between two consecutive data collections.

First, we present the consumption we collected with the phone in *idle state*, namely the phone turned on with no application running, no sensor collection in process and the screen turned off. Then we illustrate *each sensor* discharging behavior and we identify different *sensor groups*, putting together those sensors that have a similar energy consumption. Then we present how *the sampling frequency* affects each group and try to compare the values. Finally, we measured *multiple sensors* collecting data together to understand if using them in parallel affects the battery life more than using them singularly.

12.1.2.1 Idle state

When considering the phone in idle, we evaluated the idle energy consumption over multiple states given by a combination of different options available in the Android OS[2]. The options we considered are those that can be reached by swiping down from the top of the screen with two fingers:

- **Wi-Fi enabled**: the WIFI chip is enabled.
- **Wi-Fi connected**: same as above with the addition that the phone is connected to one of the available networks.
- **Bluetooth**: the bluetooth chip is enabled.
- **Location**: the locations settings are enabled.
- **Power Saving**: this is an option available on many recent Android devices that allows to smartphones consume less energy by diminishing the clock speed of the main CPU in addition to other tweaks.

[2]Not all the OS versions and smartphone have them

TABLE 12.1: Table showing all the possible states of the phone when idle and the relative current consumption values in mA. The "-" symbol means disabled while the "+" means enabled.

Current $[mA]$	Power Saving	Wifi Enabled	Bluetooth Enabled	Location	Wifi Connected
3.9	-	-	-	-	-
4.5	+	-	-	-	-
3.8	-	+	-	-	-
3.8	+	+	-	-	-
4.8	-	-	+	-	-
4.6	+	-	+	-	-
3.9	-	-	-	+	-
3.9	+	-	-	+	-
4.9	-	+	+	+	-
4.9	+	+	+	+	-
7.0	-	+	-	-	+
7.1	+	+	+	-	+
7.6	-	+	-	+	+
8.5	+	+	+	+	+

The corresponding 14 states given by a combination of the above mentioned options enabled (+) or disabled (-) are showed in Table 12.1 with the corresponding measured current consumption values in mA.

From this analysis, we are able to draw the following conclusions:

1. By simply enabling the different options (the first 10 states in the table) there is little change in the measured energy consumption that can be considered as a standard deviation due to the absolute resolution of the multimeter.

2. In the idle state, the Power Saving option does not help much in preserving the energy stored in the battery. This is due to the little computation required to keep the smartphone on, so there is really no way to save battery by reducing the CPU clock. On the other hand, when the phone is connected to a Wi-Fi network we can see a noticeable decrease in the battery life of the smartphone even if there is no transmission over the communication channel, since it is not connected to any network.

The energy consumption of the smartphone in idle state is very important in this work. In fact, we have to subtract the idle current consumption and then normalize over it to have an objective measure. From now on, the presented consumption values refer only to the normalized sensor component of the measured energy consumption. A consumption value of 200, for example, means that the specific sensor consumes 200 times the energy of the phone in idle state.

TABLE 12.2: Table showing the normalized consumption values in % with respect to the current consumption in idle state of all the hardware sensors in the Samsung Galaxy S4 smartphone, grouped according to similar consumption patterns. We show the values at different sampling frequencies and the average value of each group at each frequency.

Group	Sensor	Sampling Frequencies					
		low (5Hz)		medium (20Hz)		high (100Hz)	
		single	avg.	single	avg.	single	avg.
HW_1	Magnetic Field (MF)	6.85	6.22	7.05	6.74	7.22	7.04
	Temperature (T)	5.27		6.68		6.94	
	Humidity (H)	5.92		6.54		6.79	
	Pressure (P)	6.82		6.71		7.20	
HW_2	Accelerometer (A)	214.68	213.13	216.66	217.24	7.22	227.65
	Gyroscope (GY)	208.63		209.92		208.77	
	Orientation (O)	215.00		217.01		226.67	
	Gravity (GR)	214.20		225.38		229.54	
	Light (L)	1697.47		1749.88		2705.76	

TABLE 12.3: Table showing the normalized consumption values in % with respect to the current consumption in idle state of all the other sensors in the Samsung Galaxy S4 smartphone.

Sensor	Sampling Intervals		
	low (5min)	medium (1min)	high (5sec)
Microphone (MIC)	1665.80		
Bluetooth (BT)	46.90	81.49	126.65
Bluetooth LE (BTLE)	333.16	353.01	364.57
Wi-Fi Networks (WIFI)	44.77	46.72	50.77
Network Loc (NET)	36.31	36.34	37.06
GPS Loc (GPS)	1819.93	2032.38	3138.99

12.1.2.2 Single Sensors

The results of the analysis are summarized in Table 12.2 for the hardware sensors and in Table 12.3 for the radio sensors. The results are presented at the different sampling frequencies (5Hz (low), 20Hz (med) and 100Hz (high)) and sampling intervals (5min (low), 1min (med) and 5sec (high)) respectively. The hardware sensors show a certain homogeneity in the results; moreover, we were able to identify 2 groups that have a similar discharging pattern plus the Light (L) sensor, that is excluded from both. On the other hand, the radio sensors show little homogeneity and only between WIFI and NET, while all the others have very different behaviors in terms of both absolute consumption and frequency dependency.

TABLE 12.4: Table showing the frequency dependency of the energy consumption of the different sensor groups in percentage respect to the lowest sampling frequency.

Group \ ΔFreq	low-med	low-high	Impact
HW_1	8.46%	13.18%	Negligible
HW_2	1.93%	6.81%	Negligible
Light (L)	3.08%	59.39%	Substantial
Bluetooth (BT)	73.76%	170.03%	Negligible
Bluetooth LE (BTLE)	5.95%	9.42%	Negligible
Wi-Fi Networks (WIFI)	4.35%	13.38%	Negligible
Network Location (NET)	0.07%	2.05%	Negligible
GPS Location (GPS)	11.67%	72.47%	Substantial

12.1.2.3 Sensor Groups

In Table 12.2 we show the results for the hardware sensors by grouping together those sensors with similar battery discharging behaviors. We were able to identify two main groups: HW_1, i.e., Magnetic Field (MF), Temperature (T), Humidity (H) and Pressure (P) and HW_2, i.e., Accelerometer (A), Gyroscope (GY), Orientation (O) and Gravity (GR). Additionally, the results for the Light (L) sensor are presented but are not grouped with any other sensor since it behaves very differently. In fact, it consumes a significant amount of energy, 10 to 400 times more than any other hardware sensor in the device. The results are presented per sensor per frequency and averaged per frequency per group. The Light (L) sensor can be compared to the GPS, which is the most energy demanding sensor. Other conclusions that we can draw from these results are that sensors from HW_2 consume 30 to 40 times more energy than those belonging to group HW_1. By taking into consideration this division while defining which sensors are used to resolve a service, we can balance the energy consumption of the overall system.

12.1.2.4 Frequency-dependent Consumption

We know that the usability of the sensor data is strictly dependent on the sampling frequency, especially in context aware applications where outdated data can generate false clues and where a continuous stream of data is needed. An example of this phenomenon is the GPS signal: if I receive GPS updates every 5 minutes when I am driving and I need real time navigation information, the data is unusable since I will not receive the service I want, while at the same time if I am at work in my office it is likely I do not need frequent GPS location information. This frequency-dependent usability of the data is true also for other sensors like the hardware ones. The authors in [Ryu et al., 2013] claim that they can detect the correct average step count from Accelerometer data with an accuracy of up to 98.9% at 10Hz sampling rate and 99.6% at 20Hz sampling rate. These two examples explain the importance of the sampling frequency and show that the problem can be two-fold: we can have oversampling and undersampling at the same time. With the former we generate more data than needed with a resulting waste of energy while the latter implies having not enough data to provide the service to the user.

By looking at the results of Table 12.2 for the hardware sensors and Table 12.3 for all the others, we can compute the energy consumption variations based on the frequency and produce the new results showed in Table 12.4. It shows the percentage of variation of the consumption in using medium and high frequencies with respect to the low ones. We identified two possible behaviors for the frequency variation: *Negligible* or *Substantial*. The former refers to the fact that if the sensor has to be used, there is no difference in using it at the lowest or the highest sampling frequency. On the other hand, the latter refers to the fact that if the sensor has to be used, it should be used at the appropriate frequency; otherwise, an energy waste will occur.

As shown in Table 12.4, considering the low-med variations we have different values for the two hardware groups. For HW_1 we have a value of 8.46% for low-medium variation and 13.18% for low-high. Generally speaking, such variations values should not be discarded. However, in this case, since the starting min value is incredibly low (6.22%) if compared to the idle energy consumption value, we can conclude that the variation of the energy consumption depending on the sampling frequency for the sensors belonging to HW_1 is *negligible*. A similar result can be seen for group HW_2 but with an opposite relation between the variation and the initial min consumption. In this case we have a more significant initial minimum value with respect to the idle sensor consumption (213.13%) but the low-medium and low-high variations are lower, 1.93% and 6.81% respectively. We can then conclude that for both HW_1 and HW_2 there is no measurable change in the energy consumption at the increase of the sampling frequency. It is then suggested to use the highest sampling frequency for the hardware sensors belonging to groups HW_1 and HW_2 in order to have more data and produce potentially better results.

Similar considerations can be made for Wi-Fi Networks (Wi-Fi), Bluetooth (BT) and Bluetooth LE (BTLE) where the frequency dependency is *negligible*. As an opposite result, for GPS Location (GPS) and Light (L) sensors we have a very high dependency between the frequency variation and an increase in the energy consumption, especially for the low-high case with values of 72.47% and 59.39%. Additionally, these sensors have a very high low energy consumption value. We can conclude that for these two sensors, where the frequency dependency is *Substantial*, it is better to optimize as much as possible by using them at the lowest frequency.

12.1.2.5 Parallel Sensing Consumption

The last analysis we performed concerns the usage of multiple sensors simultaneously for the two hardware sensor groups HW_1 and HW_2. This analysis is very important for context aware applications where multiple services should be activated at the same time from multiple sensor streams. We wanted to understand whether using multiple sensors simultaneously could generate different energy consumption patterns with respect to singular sensors and possibly an unwanted overhead. We evaluated this at different sampling frequencies like in the previous analysis, i.e., low, medium and high; the results are showed in Table 12.5. Considering the sensors in groups HW_1 or HW_2 we can see that the energy consumed when all the sensors in one group are collecting data at the same time is basically the same of using only one sensor in the group. Also, considering all the sensors of both groups $HW_1 + HW_2$ collecting simultaneously, we see that their consumption can be approximated by the sum of the consumption values of one of the sensors in HW_1 and one of the sensors HW_2. More formally, the total consumption of the sensing process for the hardware sensors can be expressed as the sum of the consumption of any sensor in all the groups. Suppose there is a function $c()$ that calculates the

TABLE 12.5: Table showing the normalized consumption values in % with respect to the current consumption in idle state while using multiple sensors in parallel per Group.

Sensor Combination \ Freq	low (5Hz)	medium (20Hz)	high (100Hz)
HW_1			
T	5.27	6.68	6.94
T+P	6.16	6.59	6.85
T+P+H	6.61	6.65	6.48
T+P+H+MF	6.85	6.91	6.68
HW_2			
A	214.68	216.66	245.61
A+GR	217.26	216.58	248.90
A+GR+GY	216.29	216.58	247.59
A+GR+GY+O	212.95	216.46	236.31
$HW_1 + HW_2$			
T+P+H+MF+A+GR+GY+O	244.60	250.71	252.50

energy consumption of its argument that can be a single sensor j in a group i (Equation 12.1), multiple sensors from a single group i (Equation 12.2) or multiple sensors from multiple groups (Equation 12.3). In this scenario a sensor j in group i working at sampling frequency f is represented as S_{ij}^f where $1 < i < 2$, $1 < j < 4$ and $f \in [low, medium, high]$. The final computed overall consumption of $HW_1 + HW_2$ for $i = 1$ and $i = 2$ is presented in Equation 12.4.

$$CP_{ij}^f = c(S_{ij}^f) \tag{12.1}$$

$$CP_i^f = c(S_i^f) \cong c(S_{ij}^f) \quad \forall j, f \tag{12.2}$$

$$CP^f = \sum_{i=1}^{2} c(S_i^f) \cong \sum_{i=1}^{2} c(S_{ij}^f) \quad \forall j, f \tag{12.3}$$

$$CP^f \cong c(S_{1j}^f) + c(S_{2j}^f) \quad \forall j, f \tag{12.4}$$

The reason for this behavior can be attributed to the fact that most of the energy consumption in collecting the data goes into enabling the hardware itself rather than in processing the generated information by the CPU. For this reason, it is then more efficient to collect data from all the sensors at the same time because the amount of generated data per energy unit is higher.

12.1.3 Outcome

What emerged from the analysis of the battery consumption is that the GPS sensor must be optimized due to its high energy demand; the Light (L) sensor should not be used at all, since it consumed too much energy if compared with the value of

the data it generates; the Bluetooth generates data for free when the component is already in use by the system or it does consume an almost negligible amount of energy and should be optimized; finally all the other sensors have a negligible energy consumption and can be used without restrictions.

Additionally to the energy consumption analysis, this experiment was important because allowed us also to discover and fix numerous bugs in the i-Log application. Debugging a mobile application is usually hard because different smartphones and configurations can affect the application and generate different errors which are difficult to reproduce.

12.2 Knowdive Two

Knowdive Two is a second internal use case carried out in collaboration with the members of the Knowdive Group to test additional functionalities added to the i-Log application with respect to the first version. In particular, we focused on testing a new sensor, i.e., the audio sensor, that collects audio chunks that will be used to infer the user context from sound. Additionally, we improved the application deployment process to make it ready for production by publishing it in the Play Store[3] in order to facilitate deployments on large number of users.

12.2.1 Audio Sensor

Inferring the context dimensions from the sound collected in the environment around the user has been done before in the research community [Heittola et al., 2013; Heittola et al., 2010; Scott and Dragovic, 2005; Betsworth et al., 2013; Lu et al., 2009; Zeng et al., 2008; Eronen et al., 2006] especially for locations or activities that require sound. For this reason, we decided to add this functionality in i-Log to test if it can generate meaningful results. We decided to collect audio in chunks of ten seconds every minute, specifically to preserve the privacy of the user. We collect 10 seconds every minute.

To develop the algorithms that have to analyze the audio in the backend, we needed a lot of labeled audio data. We decided to use this Knowdive Two use case to collect this type of data. Approximately 6000 hours of audio recordings were collected during the whole experiment.

12.2.2 Fast and Easy Deployment

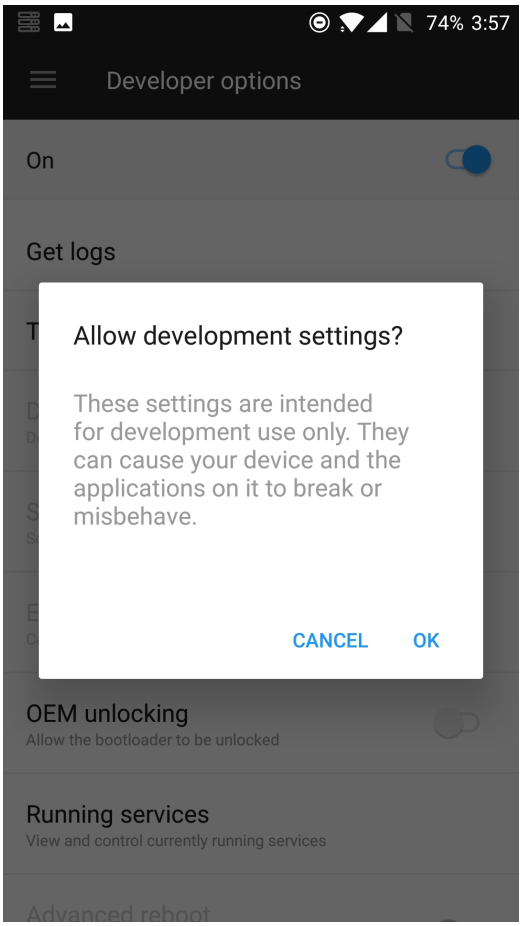
Before this test, for both Knowdive One and SmartUnitn One (see Chapters 12.1 and 13.1 respectively) the i-Log application needed to be installed manually on the smartphone of the participants. The reason was that the application was not stable enough and did not follow the policies to be published on the Play Store so that it could be downloaded directly from the participants.

A published application goes through an approval process done by Google Inc[4] that is based on different rules the developers have to follow. Among them there are rules about the content, the intellectual property, the privacy and security. For security reasons, in order to install an application that is not published, on Android, the procedure is quite long and complex. It can be done only[5] from the developer computer that has to compile the source code directly into the destination smartphone. The whole manual installation procedure can be divided into sub-procedures as follows:

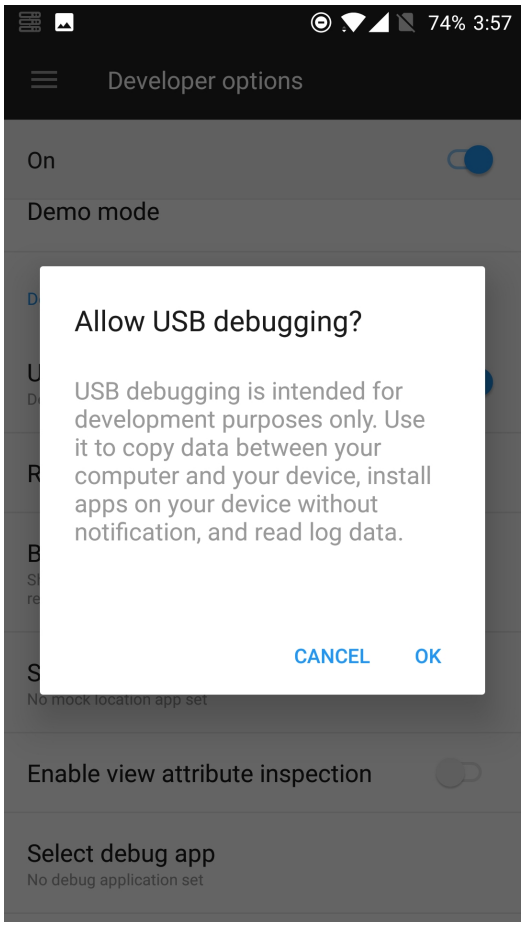
Enabling Developer Options. In order to install an application on an Android device from the computer of the developer, the so called Developer Mode must be enabled on the destination smartphone. Some applications do not even allow to be executed on smartphones while the Developer mode is enabled, for security reasons[6]. In other words, enabling this modality prevents the user from using her

[3] https://play.google.com/
[4] https://play.google.com/about/developer-content-policy/
[5] Depends on the application and on the permissions the user granted.
[6] These applications are mainly Bank or Credit Cards applications

phone normally, and this should be absolutely avoided. Moreover, it allows to perform advanced operations that done by an can damage the smarpthone itself.



(A) Message showing information about the risks of breaking or misbehaving the smartphone and the applications running on it.



(B) Message showing details about the USB debugging mode.

FIGURE 12.1: The two screenshots show the procedure to enable the Developer Mode on an Android device.

The modality by default is disabled on any Android smartphone and to be enabled, the following steps must be taken:

- Go into the **Settings**, **About Phone** menu and click 5 times the **Build Number** item. A message pops-up saying that the "Developer Options" mode is enabled.
- Go back to **Settings**, click on **Developer Options** and enable them. A message pops-up showing the danger of such as operation, as shown in Figure 12.1a.
- Scroll down till the item **USB Debugging** and enable it. Also in this case, a message pops-up showing potential risks, as shown in Figure 12.1b.

Once these operations have been made, the computer can correctly see the smartphone attached via an usb cable and the application can be installed.

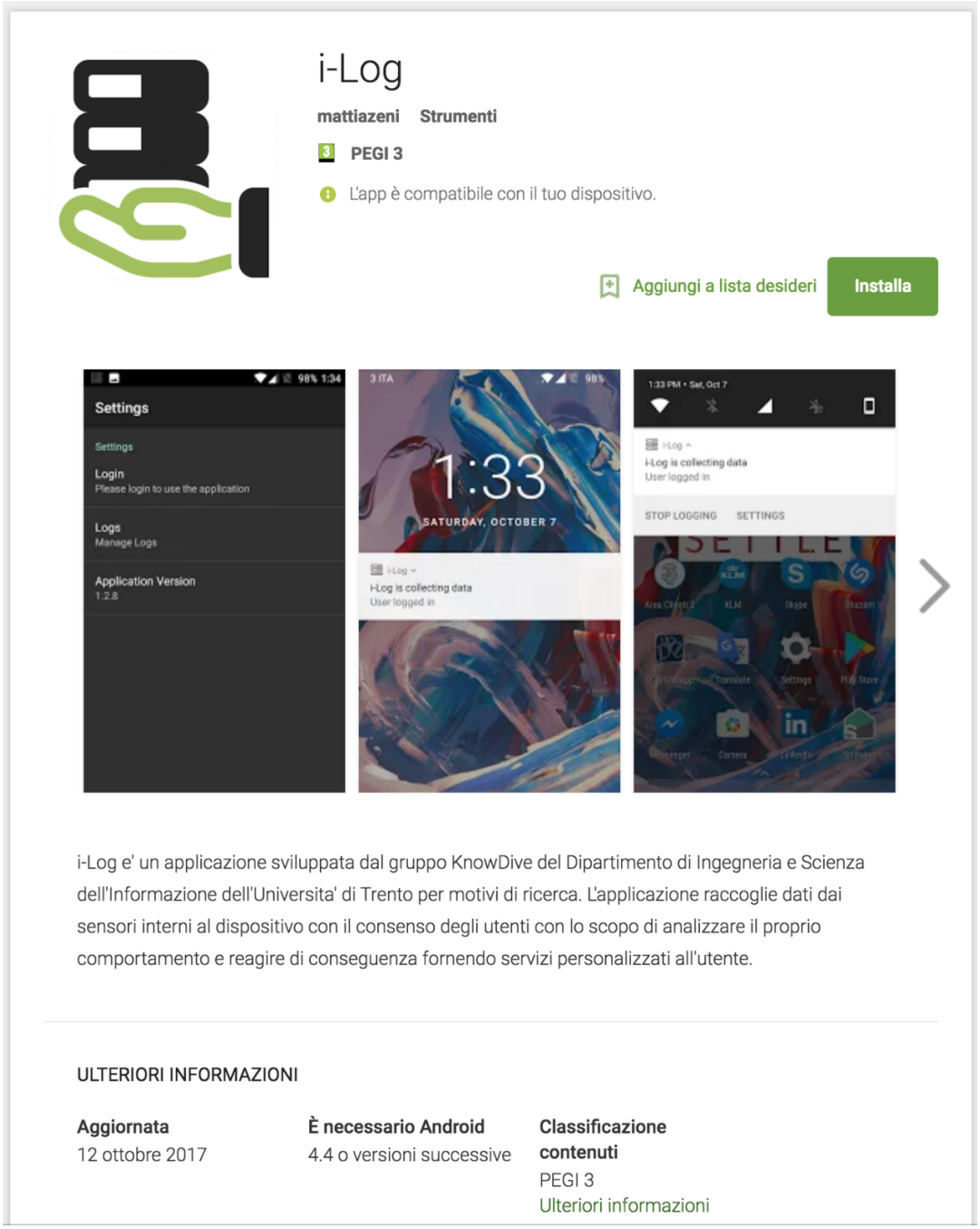


FIGURE 12.2: The i-Log mobile application published on the Google Play Store.

Compiling the Application. Compiling an application made by thousands or millions of lines of code is not an immediate task; it may usually take several minutes. In the case of a mobile application, and i-Log in particular, the code is quite compact and this operation can be done in a variable amount of time that goes from 2 up to 5 minutes.

Updating the Application. Updating an application on Android is easy and automatic for those applications downloaded from the Play Store. It usually happens

at night, when the smartphone is in charge and connected to WiFi. For an unpublished application instead there is no such process and the developer has to design the whole procedure. In fact, we created an endpoint on the server where each phone could query to see if a new update was available. This requires the phone to continuously check for updates and this corresponds to an energy waste. Additionally, the update could not be automatic but it always required the user confirmation.

This is what is required to manually install an application on an Android device. On iOS, there are even more restrictions because the device must be registered by the developer and she cannot use any smartphone she wants. If we consider the SmartUnitn One use case where we had 72 participants, doing this procedure took 2 full days of work because we had to organize sessions to meet the students and install i-Log on their smartphones.

The Play Store allows to publish an application according to different policies: public, internal to the organization, beta tester. Since we needed to control who uses i-Log, we could not yet make it public, since at the moment its usage is linked with a specific use case. We could not even make it internal to the organization (University of Trento) since some of the use cases in the near future will be on people outside it. We then decided to go for the beta tester program. It allows to send an invitation to a specific email address that will be redirected to the Google Play Store as shown in Figure 12.2.

Publishing an application on the Play Store requires additional work in order to make it compliant to the requirements. Furthermore, creating the app page itself in the store requires some time but we believe this will facilitate the deployment process and let us save an enormous amount of time in the next use cases.

Chapter 13

SmartUnitn Experiments

The SmartUnitn experiments are a set of use cases in which we applied the technologies developed in this thesis on the students of the University of Trento. The main objective is to find the correlation between how students allocate their time and their academic performances. The final goal of these experiments is to provide useful services to improve the students academic experience and even academic performances. Our aim is to perform such experiments on a regular basis, every six months, each time adding elements to the analysis and improving the overall results.

13.1 SmartUnitn One

In this section we describe the main elements characterizing the SmartUnitn One use case, in particular the objectives, the requirements, how we designed it by adapting the SB and finally some of the results we obtained from the data generated from the participants.

SmartUnitn One is the first large scale data collection trial performed with the SB on non-expert users, external to the Knowdive Group (see Chapter 12.1). The experiment was conducted by the Department of Information Engineering and Computer Science in collaboration with the Department of Sociology and Social Research of the University of Trento on students from the same institution with the final objective of filling the empirical gap concerning students time allocation and academic performance by providing a detailed description of how their time management affects their academic achievements both in terms of grades but also number of credits (i.e., number of exams).

13.1.1 Objectives

The main goal of the SmartUnitn One experiment is to understand the correlation between students time allocation and their academic performances. To do so, we instantiated the SB and install the i-Log mobile application on the students' smartphones so that we were able to collect data about them. The analysis consisted in extracting useful information from such data related to how students manage their time, and then understand if there was a correlation with their academic performances. Transitioning from high school to university is usually hard for young students because the organization in completely different. Students are no longer asked to follow schedules or instructions from the teacher but are completely free to organize their study time as they wish. It is common knowledge that a person who is able to manage her time wisely is more likely that she successfully completes the transition from high school to university. Currently, there is a lack of data about students time allocation in Italy, which are only available as aggregate data, e.g., [Mucciardi,

2013]. The focus is generally on how much time students spent for specific activities, thus ignoring the exact time these activities are occurring or their sequence. With our approach we are able also to understand precisely when students do what, e.g., we can see the differences in studying during the night or during the day, among others.

Since this is the first time we used the SB outside a controlled laboratory setting (see Chapter 12.1) this was a good opportunity also to see if the system could scale well. We collected data from all the smartphones internal sensors of the participants, and additionally we administered a questionnaire to the students. This allows us to have a dataset which is annotated and the annotations refer to how the user interpreted the situation.

The data collection lasted two weeks: during the first one, students were asked to answer a time diary on their smartphone about their time use, while the application was collecting sensor data in the background. During the second week they were only required to have the application running for collecting sensor data. To promote compliance and data quality, students received a fixed money compensation, as an incentive to participate, with additional three final prizes assigned to random users that used the application in the correct way. This quality measure was based on three parameters:

1. How much data their smartphones recorded in via GPS, Bluetooth, and Wi-Fi. We chose these three sensors since they are the only sensors that students could decide to turn off;
2. How many answers students answered;
3. How long they kept the application running, knowing that they could turn it off at any moment;

The experiment was conducted during the Winter semester, in November-December 2017, for two weeks.

13.1.2 Requirements

The requirement analysis for the SmartUnitn One use case starts from the general problems this thesis is tackling as defined in Chapter 2. In details, we adapted them on the specific target of users and the goal the use case has.

Data Collection. The different dimensions of the data collection we had to take into consideration can be summarized as follows:

- When we started designing the experiment we did not know exactly on which data we will be going to perform the analysis in order to understand the correlation between students time allocation and academic performances. For this reason, we should collect as much data as possible;
- We had to keep in mind that the way the student uses her phone could not be affected by the data collection process. For this reason, we should reduce the collection frequency so that to not affect the battery life of the phone;
- For privacy reasons, not all the data could be collected. We should avoid those data that are sensitive, like the content of the sms, among others;
- Some sensors like the GPS, WiFi and Bluetooth are the only one that can be turned off by the user. Then, we should incentivize the usage of these sensors.

- Since the size of generated data can be important, we should allow the synchronization only over a "verified" WiFi network.

User Participation. The students involvement was fundamental for the success of the experiment. We should design a mechanism to incentivize the users in using i-Log that takes into account all the dimensions of the problem:

- The users should use i-Log for as many hours as possible, even on a 24h basis if possible;
- The users should answer as many questions (questionnaire) as possible during the first week of the experiment;
- The user should keep on as long as possible the three sensors (GPS, WiFi and Bluetooth) she can manually disable

User Privacy. The privacy must guaranteed to all the participants of the experiment.

- All the data should be collected in an anonymized way and the results as well should be computed on the aggregated sample instead of on the single student;
- The user must always be informed of the collection of her data. For this reason, a specific solution should be designed so that to alert her about the data collection on the smartphone;
- The user should be in control of the collection process. We then need to allow her to stop the collection any time she wants to.
- Before starting the data collection, we obtained the approval from the ethical committee of the University of Trento.

13.1.3 Design

This section presents the design process we followed to prepare the SmartUnitn One use case. It consists of different steps that ultimately allow us to select the people for the experiment, the data we will collect from them and finally what questions they need to answer during the data collection. All these elements are highly dependent on the use case.

13.1.3.1 Sample Selection

In SmartUnitn One, the focus was on student from our university. We sent out a call for collaboration to all the students enrolled at University of Trento in the Academic year 2015-2016 and in particular only those who fulfill these criteria:

- to have filled three university surveys in order to obtain their socio-demographic data, shown in Table 13.1, and other characteristics, e.g., psychological and time use related, that could be later matched with our application's data. The demographics reported in the table are only some of the one that are available. We decided to show these ones because are the one we believe are most related with the academic performances of the students: many works correlate the difference in academic performances between men and women, the same for the faculty they belong to i.e., Engineering vs Philosophy. The scholarship

factor is more related with the number of exams a student does since at the University of Trento, in order to get the scholarship, the students have to fulfill some minimum requirements such as obtain more that 18 credits before March of the first academic year, among others;

- to attend lessons during the period of our experiment in order to describe their daily behavior during the university experience;
- to have an Android smartphone with an Android version 5.0.2 or higher. This is motivated by the fact that some functionalities in i-Log cannot run on devices with older versions of the operating system.

TABLE 13.1: Socio-demographics of SmartUnitn One students

Gender		**Departments**		**Scholarship**	
Male	Female	Scientific	Humanities	True	False
61.1%	39.9%	56.9%	43.1%	37.5%	62.5%

From the students reached by the call (almost 300), 72 replied and finally gave their availability for the experiment. One important thing to underline is that this recruitment procedure was done only by the team involved in the experiment, without leveraging on the University. We believe that having done an official call from the general office of the university would allow to have much more participants and this is what we want to do for the next experiments.

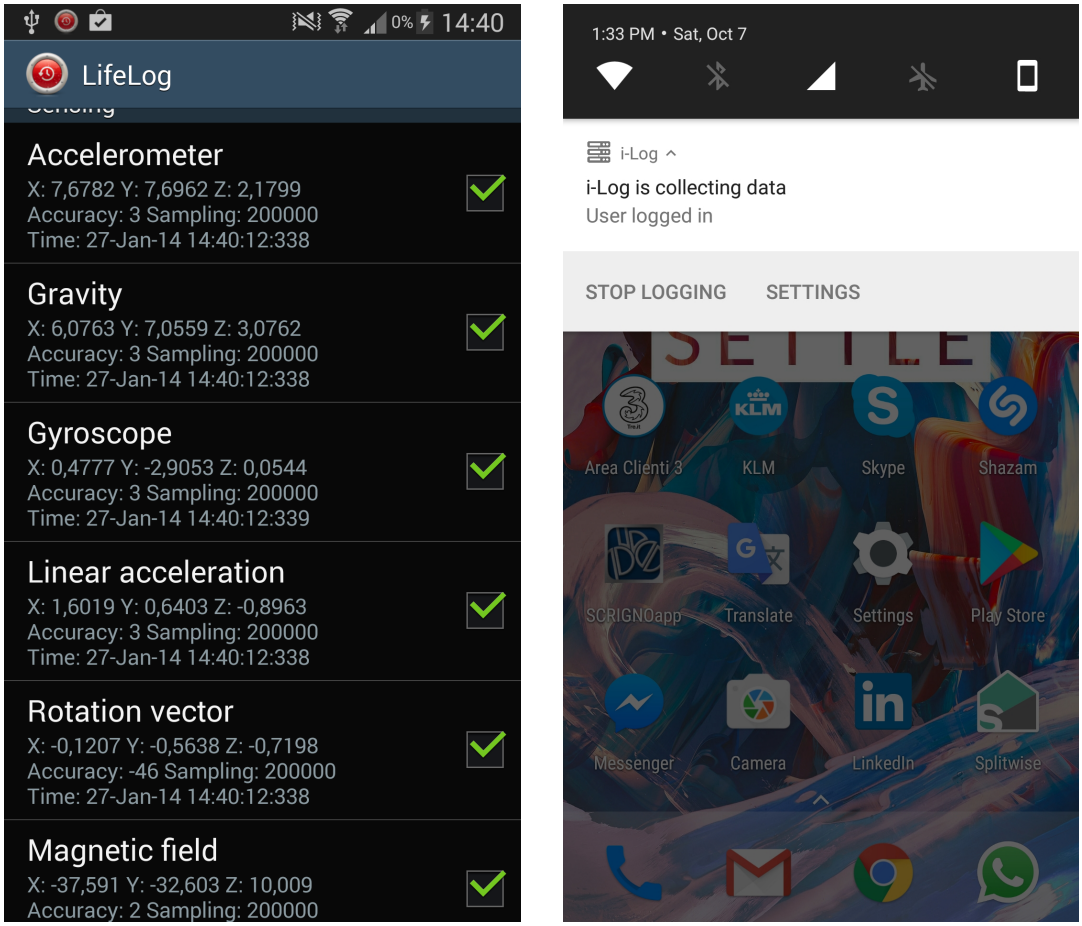
The students were asked to attend an introductory presentation where they are presented with the aims of the project and how to use the i-Log application. If they wished to participate, after the presentation they signed a consent form, and then installed i-Log on their own smartphones. Users were informed about all aspects of the management of their personal information concerning privacy, from data collection to storage to processing.

13.1.3.2 i-Log Application User Interface

The i-Log application was originally designed to be the main tool for the data scientist to collect data from her own smartphone to be used to test the system and algorithms. For this reason, it was not user-friendly and complicated to use for a normal user. Figure 13.1a shows what was the main window of the original application called LifeLog back in 2014. As we can see, the data scientist had complete control on the data collection: it could see the collected data in real time, it could enable/disable single sensors and had many more settings he could set.

With a non-expert user it was unfeasible to keep the same user interface. There are several reasons for this:

- From a technical point of view, the UI was requesting too many CPU resources in order to update the views in real time since the sensor values were changing multiple times per second.
- The user was scared by seeing all those data flowing. She had the idea of being monitored more than necessary.
- The possibility of disabling single sensors was a problem since the user could erroneously toggle them.



(A) Original design of the LifeLog application.

(B) Final design of the i-Log application.

FIGURE 13.1: The two screenshots show the progresses made on the i-Log user interface to make it more user-friendly.

For this use case we redesigned completely the user interface, removing all the unnecessary elements. We ended up removing the application main window, and showing only the notification in the system control panel where the user could stop the data collection or open a setting menu where she could login with her credentials, as shown in Figure 13.1b. We also removed the application from the list of recent apps because our tests suggested that many users were annoyed by it. In all the versions of the Android OS, there is a physical button that allows to visualize the recent apps used and still open. It is usually presented as a list and the user can swipe every app to completely close it and save memory and battery. Having i-Log in that list was perceived in the wrong way by the user that most of the time, intentionally or unintentionally, was closing it.

13.1.3.3 Sensor Selection

Since this was the first trial with real participants that were external to our group (and then people that can be considered non-experts) we did not have a clear idea yet of what to collect since we did not design how to analyze the data. For this reason, we collected data from all the available sensors in every smartphone of the participants, taking into account the battery requirement that relates to the sampling

frequency of each sensor. In order to affect the battery life as little as possible, we decided to reduce the frequency at which we were collecting data, as showed in Table A.1 in Appendix A.

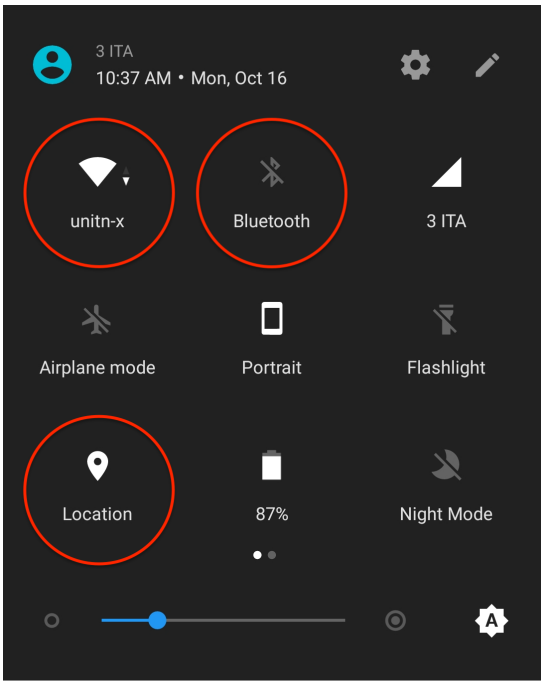


FIGURE 13.2: The user can manually disable the GPS, WiFi and Bluetooth sensor through the top menu in the Android operating system.

As described in Section 4.1.2, the data collection on the smartphone is remotely configurable in order to adapt to all the use cases. However, we have little or no control on three of the available sensors i.e., the GPS, the WiFi and the Bluetooth. In fact, we can decide to collect data from them or not, we can define the collection frequency, but the user is always in control. As shown in Figure 13.2, using the top menu, the user can manually disable them. By doing so, the data collection from those sensors stops and i-Log has no other mean of obtaining that information. Unfortunately those are also very important sensors because they allow to localize the users.

13.1.3.4 Time Diaries Design

Students were asked to reply some question during the day for the first week of the experiment. These questions were administered as time diaries.

Time use surveys are particularly relevant approaches, since they are widely used to investigate a specific aspect of people's time management, e.g., working, academic performance, and so on [Claessens et al., 2007]. In fact, we based our modelling for activities on several time use surveys, especially the American Time Use Survey (ATUS) [Shelley, 2005].

To test and apply our methodology, we interacted with sociology experts in the SmartUnitn One project for linking student behaviour and academic performance. Students are recruited via surveys and participate by signing a consent form allowing an application, is installed on their smartphones. The project lasted two weeks: during the first one, students needed to answer a questionnaire on their day and must carry their phone with them for the collection of sensor data. The interaction

with sociology experts for this project led to the following methodological considerations:

1. **From ontology to annotation lists:** Following the sociology experts inputs, we made our general ontology model into a list of annotations, without any sort of hierarchy. In fact, a simpler, leaner presentation is more likely to elicit and engage the students' answers, coupled with a controlled vocabulary for reducing possible ambiguities. In order to capture the most salient triple of location, activity and social relations [Hellgren, 2014], the annotations act as a list of possible answer for the corresponding questions, i.e. "Where are you?" (locations), "What are you doing?" (activities) and "Who is with you?" (social relations).

2. **No *WI*:** In the case of this experiment, out of the four context dimensions, the sociology experts do not deem the *WI* relevant. Thus, no mapping with the object context is required.

3. **Ordering of the questions:** According to the sociology experts, and in general for time use surveys [Hellgren, 2014], activities are more relevant than locations and social relation in the experiment. Thus, the ordering of the three question mirrors this hierarchy: activities first, locations second and then social relations.

4. **No locations and activities constraints:** In activity recognition, locations can often act as constraints for the activities performed there [Riboni and Bettini, 2011]; for instance, when in bathrooms, people take a shower instead of cooking. However, from a sociological point of view, constraints may lead to a loss of valuable sociological data, e.g., students studying in places not explicitly designed for it, such as workplaces, bars or gyms. As a result, no constraints are imposed between the locations and activities annotation lists.

5. **Adding "Other":** In time use surveys, the answer "Other" is a standard option with possible variations, e.g., the "n.e.c." field (i.e., Not Elsewhere Classified) in the ATUS Shelley, 2005. Methodologically speaking, this means that the possible activity, location or social relation is outside the research scope of the sociologist, so it does not matter; "Other" covers such cases [Claessens et al., 2007]. Ontologically speaking, "Other" acts as an element of openness, i.e., as a placeholder node in the ontology to accommodate and expand new pieces of information to be added in time to an ontology.

The result of the mapping between our initial ontology and the sociological inputs for the experiment is three different lists of annotations. Notice that there is a decreasing level of granularity among activities, locations, and social relations in the mapping.

- **Activities:** Figure 13.3 shows the mapping of activities, i.e., the *WA* context, and the question about activities. Here the annotations are adapted by the first tier of activities, especially for "Relax", which maps to 4 annotations, i.e., "Hobbies", "Cultural Activity", "Other Free Time", and "Social Life". This coarseness in the mapping is due to the fact that, in order to capture high level patterns, activities are required to be very general. Furthermore, more detailed activities, as underlined by the sociology experts, would cause more cognitive load in terms of memory for students and force them to answer more questions to reach an unnecessary fine grained level of detail.

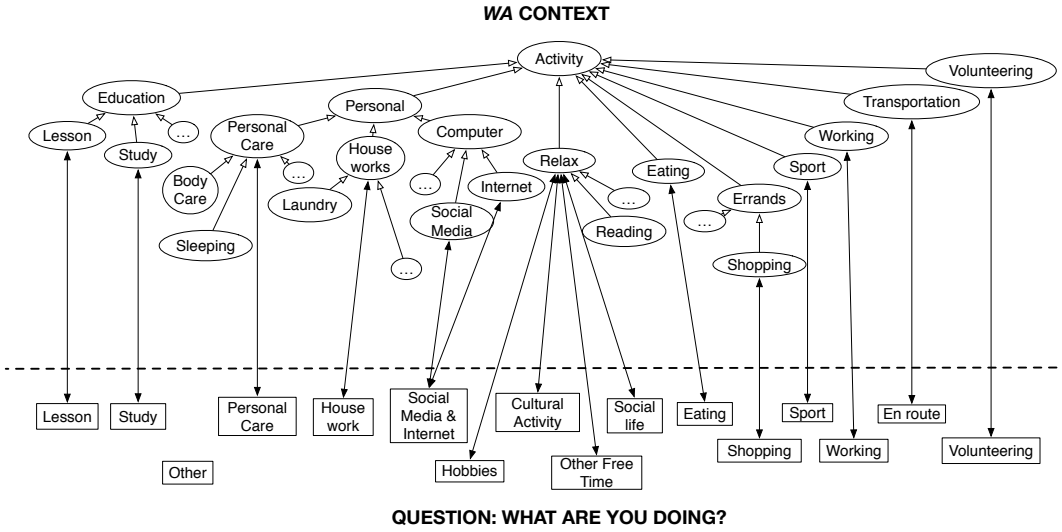


FIGURE 13.3: The mapping from *WA* to the activities annotation list.

- **Locations:** Figure 13.4 shows the mapping from the locations, i.e., the *WE* context, to the question about locations. Here the mapping is almost one to one with the lowest tier, except for “Other University place" and “Other Home", since they group more specific types of buildings.

  Notice that, even though “En route" is an activity, it refers to actual locations. So, if a student chooses it, then, instead of the options in Figure 13.4, a list of means of transportation is provided and the question is “How are you travelling?”. The possible means of transportation are listed exactly as suggested by the sociology experts, i.e., “By Foot", “By Bus", “By Train", “By Car", “By Motorbike", and “By Bike".

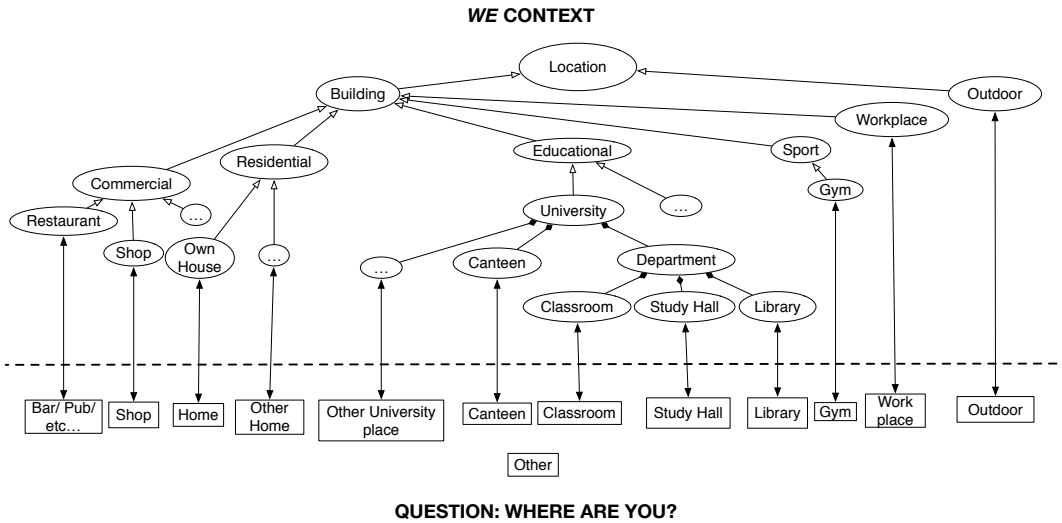


FIGURE 13.4: The mapping from *WE* to the locations annotation list.

- **Social relations:** In the case of social relations, unlike locations and activities, the mapping is one to one, since they are a simple list in the *WO* context, as shown in Figure 13.5.

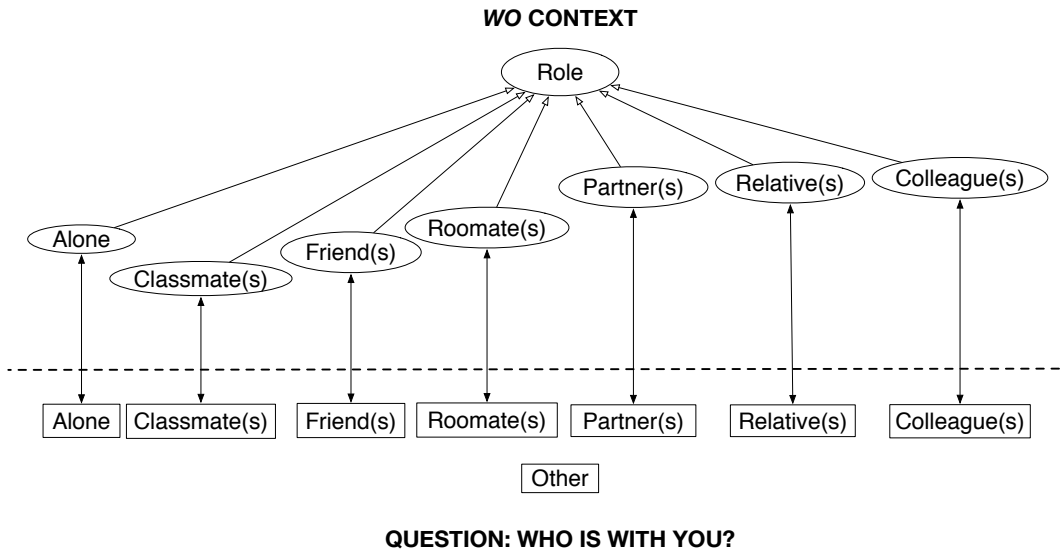


FIGURE 13.5: The mapping from *WO* to the social relations annotation list.

The three lists of annotations compose the questionnaire to be administered to students, shown in Table 13.2.

TABLE 13.2: The questionnaire administered to the users.

What are you doing?	**Where are you?**	**Who is with you?**
Lesson	Class	Alone
Study	Study Hall	Classmate(s)
Eating	Library	Friend(s)
Personal Care	Other University place	Roomate(s)
En Route(*)	Canteen	Partner(s)
Social Life	Bar/Pub/etc...	Relative(s)
Social media/internet	Home	Colleague(s)
Cultural Activity	Other Home	Other
Sport	Workplace	
Shopping	Outdoors	
Hobbies	Gym	
Other Free Time	Shop	
Work	Other Place	
Housework	**(*) How are you travelling?**	
Volunteering	Be Foot	
Other	By Bus	
	By Train	
	By Car	
	By Motorbike	
	By Bike	
	Other	

Each list of answers is the mapped set of annotations from Figure 13.3, i.e., activities answering the question “What are you doing?", Figure 13.4, i.e., locations

answering the question “Where are you?" and Figure 13.5, i.e., social relations answering the question “Who is with you?". The link between the fourth question “How are you travelling?” and the “En route" activity is shown via an asterisk at the end of the latter.

13.1.4 Results

There are three main three main categories of results from SmartUnitn One: the dataset, the biases and the quality of the answers.

13.1.4.1 Behavioural Dataset

The immediate result of the SmartUnitn One experiment is a dataset of annotated sensor data about university students. We collected a total of 110 Gb of data from the 72 students for the whole duration of the experiment. The resulting dataset is a behavioural dataset that is fully semantic, and exploiting sociological insights from the very beginning. It is also merged both with socio-demographic characteristics of students obtained through surveys and academic performance data from the administrative office of the University of Trento. The privacy is guaranteed by the i-Log infrastructure through data anonymization in all the steps, from data acquisition, to storage to processing.

This dataset allows us to study what affects the academic performances of the students under many possible points of view: the locations they visit, how they use their time, how they use the smartphone or social networks, among many others. In the next sections we present some of the results we published starting from this dataset.

13.1.4.2 Quantifying Students Biases

The main tool that sociologists employ for investigating human behaviour with respect to time use are time use surveys (also defined as time diaries), where respondents provide answers about their management of time. However, there are several issues related to respondents' behaviour that affect time use surveys. For instance, unwillingness to respond or insufficient cooperation [Juster and Stafford, 1991] are when the sample of respondents avoid answering questions entirely or become gradually unresponsive during the survey. These behaviours may be due to, e.g., the respondent not understanding instructions or not paying attention to introductory explanations on the survey process [West and Sinibaldi, 2013]. Conditioning is a change in respondents' behaviour as a result of keeping the diary, i.e., that their reported behaviour changes to underline or minimise aspects that they believe to socially desirable with respect to the survey aims [Corti, 1993]. For instance, people may report that they spend more time on exercising than they actually do because they are self conscious about their health.

The two most important issues are memory bias [Freedman et al., 2013] and carelessness [Corti, 1993]:

1. **Memory bias** is the inadequate recall of respondents when reporting their time use. The main reason is that the answers are often given with a delay after being carried out [Tourangeau, Rips, and Rasinski, 2000]. We formalize this behaviour as a parameter called $\mathbf{\Delta_{QA}}$, i.e., the time interval (in minutes) from when the question is presented to respondents to when the answer is given.

2. **Carelessness** is the set of behaviours that lead to hurriedness when reporting time use. This term accounts for misleading answers, be they intentional or due to, e.g., distraction or similar factors. We formalize it as a parameter called $\mathbf{\Delta_{A(X,Y)}}$, defined as the time interval (in seconds) elapsed from when the user starts answering one question of the time diary entry X and answers another question Y, where $X >= 1$, $Y <= Z$, and Z is the total number of questions and $X < Y$. The higher the value, the better in terms of reliability.

These two issues are a source of inconsistencies in survey data. We define inconsistencies as the mismatch between the obtained survey data and their adherence to reality. In order to identify them, a further source of information is needed; smartphones can act as such thanks to their sensors. In fact, their sensors can used as proxy for recognizing locations, activities and also social relations with different degrees of accuracy and coverage.

Table 13.3 presents general statistics about the answers provided by the students during the first week of the experiment. Out of a total number of 27111 collected answers, 9905 were empty because expired, with a final value of valid answers of 17207. It is important to notice that the reasons for these answers to be expired are various and one of them is that the student was sleeping. We also report data about the "Home" answer because will be used in Section 13.1.4.3. Moreover, out of 8729 answers indicating "Home", only 6474 of them are associated with sensor data about the actual position, either via GPS or NETWORK. This is due to the fact that students could disable the data collection about their location at any time.

TABLE 13.3: Statistics about the collected answers using i-Log within the SmartUnitn One project.

Total	27111	100%
Empty	9905	36.53%
Valid	17207	63.46%
Home	8729 (6474)	32.19% (23.87%)

In the SmartUnitn One project the students had up to 2.5 hours to reply to a question, hence the Δ_{QA} ranges from 0 to 150min. Figure 13.6 shows the distribution of the values of the parameter across all the 17207 answers generated by the students during the 7 days of the experiment. The mean value was identified as 30.44 minutes with a standard deviation of 37.5, which is much higher than the standard 10 minutes in traditional time use surveys [Claessens et al., 2007]. The general trend is that many answers were given for a low value of Δ_{QA} and decreased upon nearing to the maximum limit of 150 minutes to answer. This result can be attributed to the incentive mechanism, since students were incentivized to reply to as many questions as possible.

The same result is presented in Table 13.4, giving more emphasis on the percentages of the distribution; notice that only 41.5% of the questions were replied in less than the standard 10 minutes interval [Romano, 2008]. This result suggests that perhaps this threshold is too low. Considering instead the average value of 30.4 minutes obtained from our data, the coverage reaches an higher value of 66.4%. At the same time, we can also see that 82.3% of the answers were given within an hour.

In order to better characterize these results, we evaluated them according to other variables because students' interaction with the survey could depend on elements related to the outside world or their socio-demographic characteristics. More in detail, we looked at the following elements:

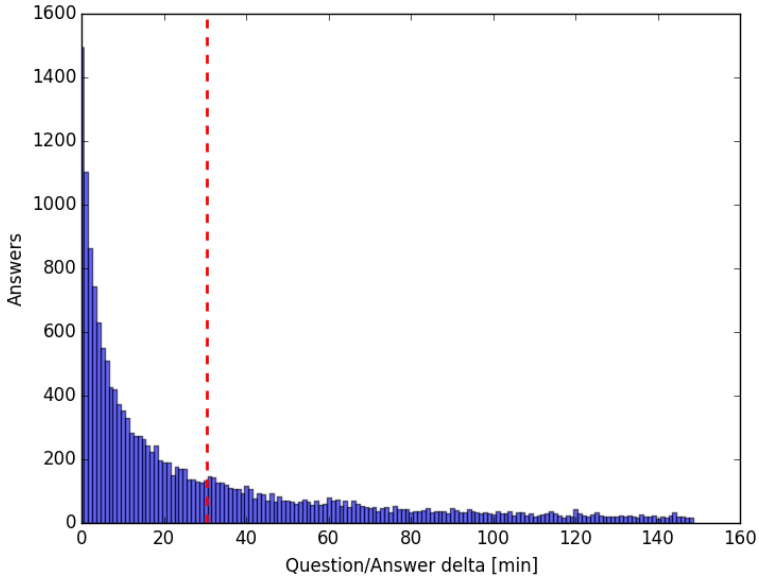


FIGURE 13.6: Distribution of the Δ_{QA} parameter. The red and blue dashed vertical lines represents the mean value of 30.4.

TABLE 13.4: Distribution of the Δ_{QA} parameter at different time slots.

	10min	**30min**	**60min**	**90min**	**120min**	**150min**
Δ_{QA}	41.5%	24.9%	15.9%	8.4%	5.4%	3.8%

- Time variations by considering differences for both working-days (from Monday to Friday) and the weekend (Saturday and Sunday) and for different time intervals within the day (2hours slots, starting from 7AM to 11PM);
- Individual characteristics such as:
  - Gender;
  - Faculty in which students are enrolled, distinguishing between scientific faculties and humanities;
  - Whether student are exempted from university fees.

The results are presented in Tables 13.5 and 13.6 where the former shows the average values of Δ_{QA} while the latter shows how the distribution changes.

An interesting pattern emerged. Firstly, students took 5 more minutes on average when completing the questions during the weekend with respect to the week days, moving from 34.7 minutes down to 29.7. This result can be better observed in Figure 13.6, where answers appear to distribute towards the higher values. Secondly, not all time intervals within the day show the same behaviour. We found a significant increase of 9.2 minutes in Δ_{QA} in the first time slot (7AM-9AM), which covers the time slot where many students were probably still sleeping. Considering the other time slots, there is little difference among them except for a slight increase in Δ_{QA} considering the last two time slots of the day, i.e., 7PM-9PM and 9PM-11PM. The socio-demographics variables reveal a delay of time answer both for males (mean value of Δ_{QA} of 31.6 minutes) with respect to females (mean value of Δ_{QA} of 29.9 minutes) and for students enrolled in a scientific faculty (mean value of Δ_{QA} of 31.8 minutes) with respect to those from humanities (mean value of Δ_{QA} of 29.7 minutes). Finally, students with scholarship (mean value of Δ_{QA} of 29.6 minutes) have

TABLE 13.5: Mean values and standard deviation for the Δ_{QA} parameter calculated based on real world and socio-demographical variables.

	Mean	**St. Dev.**
Working-days	29.7	36.9
Week-end	34.7	38.9
7AM - 9AM	39.6	42.2
9AM - 11AM	25.5	31.8
11AM - 11PM	23.4	28.9
1PM - 3PM	24.8	30.4
3PM - 5PM	24.3	27.8
5PM - 7PM	23.2	27.9
7PM - 9PM	27.3	31.9
9PM - 11PM	27.0	30.7
Male	31.6	36.5
Female	29.9	39.0
Scientific	31.8	36.7
Humanities	29.7	38.6
Scholarship	29.6	36.3
No Scholarship	31.7	38.2

TABLE 13.6: Distribution of the Δ_{QA} parameter at different time slots based on real world and socio-demographical variables.

	10min	**30min**	**60min**	**90min**	**120min**	**150min**
Working-days	43.1%	24.7%	15.4%	8.0%	5.2%	3.6%
Week-end	35.9%	25.9%	18.0%	9.7%	6.1%	4.3%
7AM - 9AM	32.6%	23.3%	18.2%	12.4%	8.0%	5.5%
9AM-11AM	48.5%	22.5%	14.8%	7.2%	4.5%	2.4%
11AM-1PM	47.7%	26.3%	14.6%	6.1%	3.4%	1.9%
1PM-3PM	46.8%	26.1%	14.1%	6.9%	4.2%	1.9%
3PM-5PM	44.1%	27.6%	17.0%	7.5%	2.6%	1.3%
5PM-7PM	46.2%	27.4%	15.7%	6.4%	3.0%	1.3%
7PM-9PM	42.3%	26.5%	17.0%	7.4%	4.3%	2.5%
9PM-11PM	42.8%	26.0%	17.2%	8.5%	3.8%	1.7%
Male	40.0%	25.1%	16.5%	8.9%	5.5%	3.8%
Female	43.6%	24.7%	15.1%	7.6%	5.2%	3.7%
Scientific	40.3%	24.5%	16.6%	8.8%	5.8%	4.0%
Humanities	43.0%	25.7%	15.1%	7.8%	4.9%	3.5%
Scholarship	43.2%	25.1%	15.2%	7.9%	5.0%	3.6%
No scholarship	40.3%	24.9%	16.5%	8.7%	5.7%	3.9%

a lower Δ_{QA} than students without it (mean value of Δ_{QA} of 31.7 minutes).

Moving to $\Delta_{A(X,Y)}$, we calculated an average value of 8.8 on the 17207 answers, with a standard deviation of 33.2; however, this results cannot be compared with other previous methodologies since they are not based on smartphones. Table 13.7 shows how the answers are distributed across all the possible values of $\Delta_{A(X,Y)}$. We defined six time intervals to represent all the samples; moreover, the upper bound was set to 60 seconds since only 0.04% of the answers were given with a duration above this threshold.

TABLE 13.7: Distribution of the $\Delta_{A(X,Y)}$ parameter at different time slots.

	4sec	**8sec**	**12sec**	**20sec**	**40sec**	**60sec**
$\Delta_{A(X,Y)}$	36.96%	40.04%	11.14%	7.05%	4.01%	0.76%

Similarly to Δ_{QA}, we tried to evaluate also $\Delta_{A(X,Y)}$ according to other variables related to the outside world or the students' socio-demographic characteristics. In this case, the results do not show any relevant variation. As an example, Figure 13.7 refers to the distribution of $\Delta_{A(X,Y)}$ calculated depending on the whole week, weekdays and the weekend. We can only report a small variation and in particular a delay of 1.02% during the weekend.

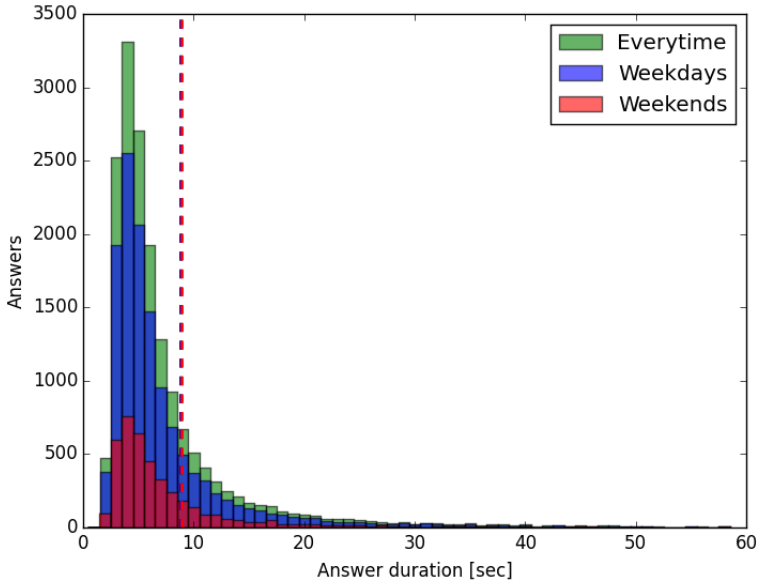


FIGURE 13.7: Distribution of the $\Delta_{A(X,Y)}$ parameter accounting for weekdays and weekends. The red and blue dashed vertical lines represents the mean values of respectively 8.87 and 8.78.

13.1.4.3 Using the Biases to Find Inconsistencies in Students Home

To the best of our knowledge, our approach is the first one that allows to quantify these biases. The result of such a quantification is that we can account for them when using the answers generated by the students, thus finding **inconsistencies**.

Among different sensing strategies that could be employed for evaluating the inconsistencies in answers out of sensor data, we chose to focus on the location of

the students since it is easier to detect. Among the available location options from the questionnaire, we chose students' homes, since it is reasonable to think that a student could have at most 2 homes. We validated this claim by manually comparing students' collected locations with their demographic data provided by the University of Trento; in fact, the average number of locations per student was 2.17. Students were instructed during the presentation meeting to consider "Home" as their main residential locations. They were supposed to do so regardless of the fact that they were either residents in Trento or commuters or that they usually went back to their hometown during the weekends; this last case would in fact imply an additional residence. Nonetheless, in case they where staying in other private building, e.g., houses belonging to members of their social circles, then they were supposed to mark them as "Other Private Home". From a sensor point of view, the answer "Home" was associated with a unique location point collected by the smartphone when the answer was generated, either from the GPS sensor or calculated through the network Wi-Fi connection. By clustering together the points belonging to the "Home" answers for each student, we were able to identify a certain number of proxy locations. Hence, each student should have at maximum two clusters, assuming some of those returning home in the weekend may have mistakenly chosen their hometown.

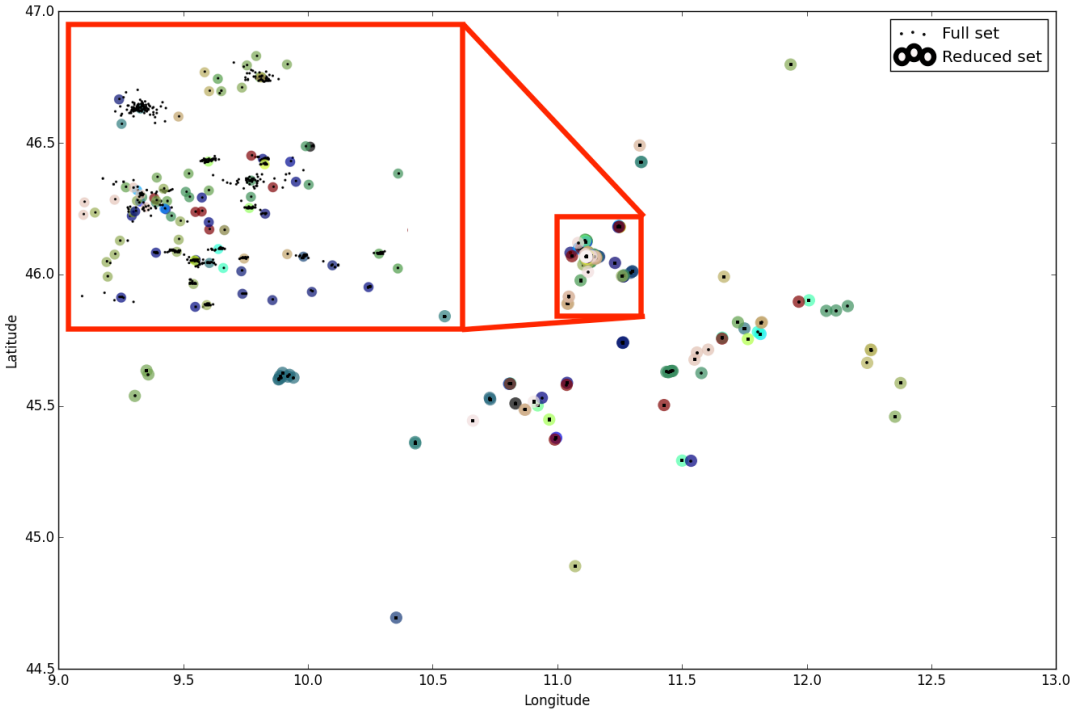


FIGURE 13.8: Distribution of the 'Home' clusters across a latitude, longitude representation. The clusters are represented by the colored circles labelled as reduced set where each color represents a student. The black dots of the full set represent the original points we computed the clusters from. The red area is a zoom over the municipality of Trento that contains most of the dots.

We used DBSCAN [Ester et al., 1996] as clustering algorithm, which is based on the spatial density of the points to cluster. Among the parameters of the algorithm there is ε, the maximum distance between two points so that they are considered as in the same neighborhood. To characterize this value, we averaged the accuracy of all the points collected by the location sensor at the time of the "Home" answer

input. In fact, the location collected with i-Log is composed by a tuple of information, namely *latitude*, *longitude*, *altitude*, *speed*, *bearing*, *accuracy* and *provider*. The *accuracy* variable contains information about the accuracy of the current measurement, where the lower the better, represented in meters from the point centered in $latitude, longitude, altitude$, and usually varies from few meters when using the GPS up to tens of meters when using the Wi-Fi network provider, or even up to hundreds of meters when using the cellular network provider. We initially observed an average value of 395.81 meter with a standard deviation of 5705.27 meter on a total of 6548 considered points. However, we noticed that 74 of those 6548 positions where characterized by a very high value of the accuracy variable, ranging from 20 Kms up to 180 Kms. This was certainly due to a cold start problem of the GPS sensor, since, during the first few seconds after a period of inactivity, it usually provides data with a very low accuracy. Since we could reconstruct the source of these data, which affected only 1.1% of the total points, we decided to treat them as outliers and exclude them from the analysis. After this exclusion, we found an average value for the accuracy of 108.27m with a standard deviation of 314.68m that can be correctly used as ε parameter for the DBSCAN algorithm.

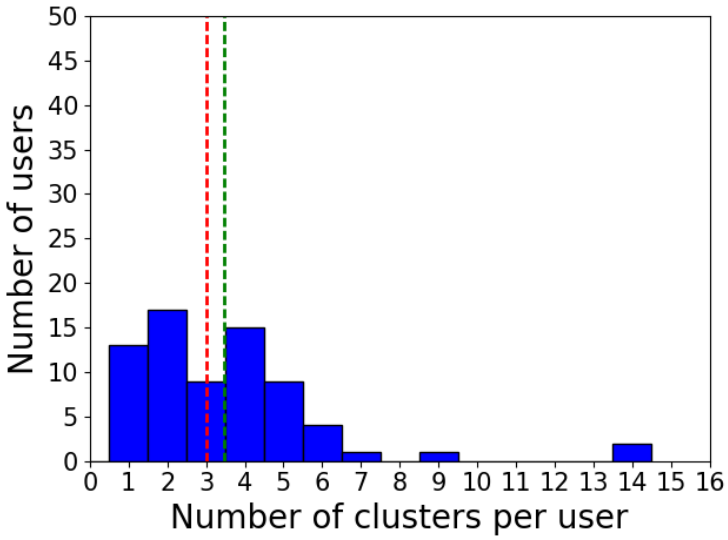


FIGURE 13.9: Distribution of the number of clusters for the “Home" location.

Figure 13.8 shows the clusters (reduced set) generated from all the points (full set) extracted from the “Home" answers, reported in a latitude/longitude reference system. The area covered between latitude 44.5-47.0 and longitude 9.0-13.0 represents the northern region of Italy, where the students of our University usually reside. The highlighted red area in the Figure is a zoom over the municipality of Trento that contains the majority of the points and consequently of the clusters.

Moving to the analysis of the results, Figure 13.9 represents the distribution of the number of clusters for the “Home" location across the students participating in the experiment.

While 2 homes was a reasonable and expected upper bound, the results showed that the clusters of size 1 to 6 have higher occurrence values with two peaks for 2 and 4 clusters. Moreover, a few students claimed that their home corresponds to 9 or even 14 different places. This result shows a mean value of 3.47 home locations

per student with a standard deviation of 2.49 and a median value of 3.0. This result does not follow our expectations of at most 2 homes per student and shows that our approach correctly identified inconsistency in the location data for the "Home" answer. The clusters are calculated out of 6409 points belonging to all the users in the experiment.

13.1.4.3.1 Memory bias vs. inconsistencies in "Home" Concerning Δ_{QA}, we considered an additional value other than the 30.44 minutes from Section 13.1.4.2, since we wanted to compare with the 10 mintues time interval usually used in time surveys, also adopted in [Sonck and Fernee, 2013].
Figures 13.10 and 13.11 show how the distribution of the number of clusters per students changes depending on whether the Δ_{QA} parameter is greater and lower than 10 minutes, respectively.

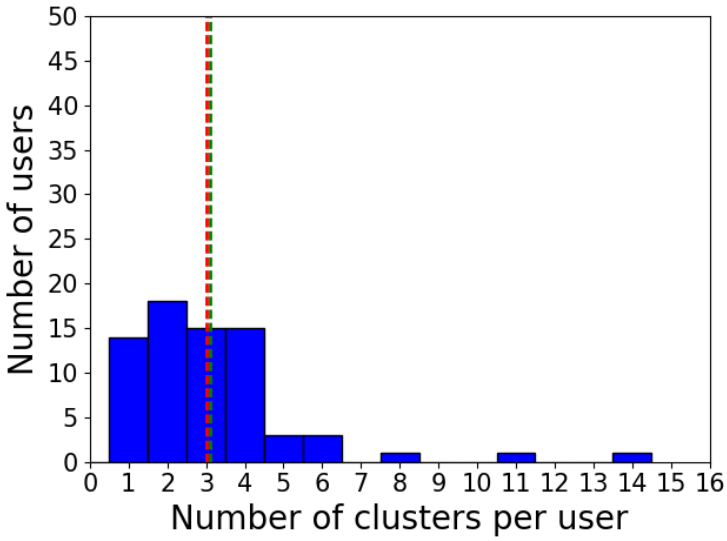


FIGURE 13.10: Distribution of the number of clusters for the "Home" location with Δ_{QA} higher than 10min.

For Δ_{QA} greater than 10 minutes, we can underline that the result is almost the same as the one without considering this parameter. In fact, the distribution did not change, showing a mean value of 3.11 with a standard deviation of 2.19 and a median value of 3.0. On the other hand, with a value of Δ_{QA} lower than 10 minutes very different clusters distribution with an average value of 2.28 with a standard deviation of 1.54 and a median value of 2.0. These examples respectively account for 3887 and 2563 location points belonging to all the participants in both cases.

The second value of Δ_{QA} we decided to test was the mean delta value out of students answers from Section 13.1.4.2. Figure 13.12 and Figure 13.13 show the cluster distribution with Δ_{QA} higher and lower than 30.44 minutes, respectively.

The results for the clusters distribution for Δ_{QA} are almost the same for value greater or lower than 30.44. For values greater than the threshold we have a mean of 2.52 and standard deviation 1.88 and a median value of 2.0. In the other case, the results have a mean value of 2.81 and standard deviation 1.92 and a median value of 2.0. The former results are produced out of 2315 location points while the latter out of 4135 both belonging to all the participants.

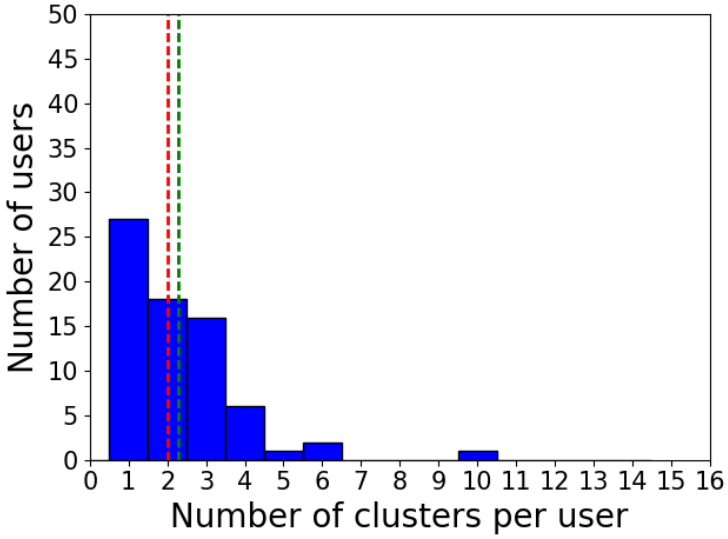


FIGURE 13.11: Distribution of the number of clusters for the “Home" location with Δ_{QA} lower than 10min.

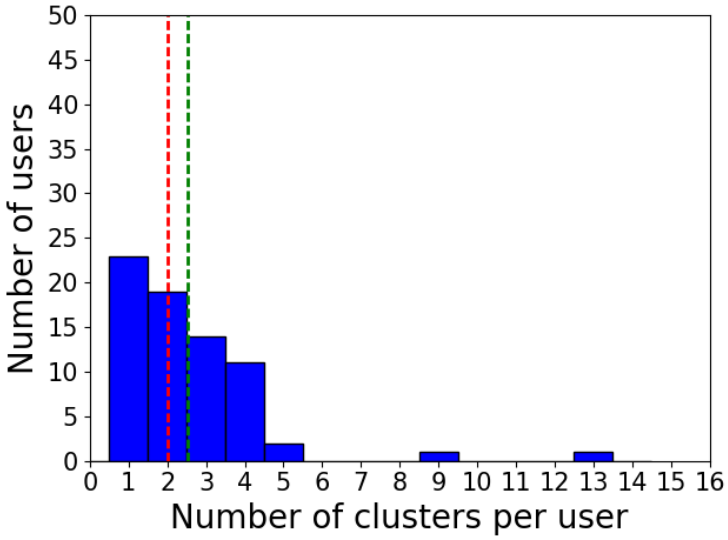


FIGURE 13.12: Distribution of the number of clusters for the “Home" location with Δ_{QA} higher than 30,44min.

Summarizing the considerations for the Δ_{QA} parameter as illustrated in Table 13.8, we can say that staying below a certain threshold value can improve the consistency of the results. The fact that this phenomenon is more evident for the standard time use survey threshold value of 10 minutes, which in [Sonck and Fernee, 2013] is a source of criticism by respondents, suggest that non-mobile solutions are effectively filled at a later time, and that this same value may not be directly applicable to smartphone-based survey. The same happens with the case of answers provided after the Δ_{QA} of 30.44 minutes. Overall, these results provide a quantitative estimation, in addition to a corroboration, of the first issue of time use surveys identified in Section 13.1.4.2, i.e. memory biases in respondents.

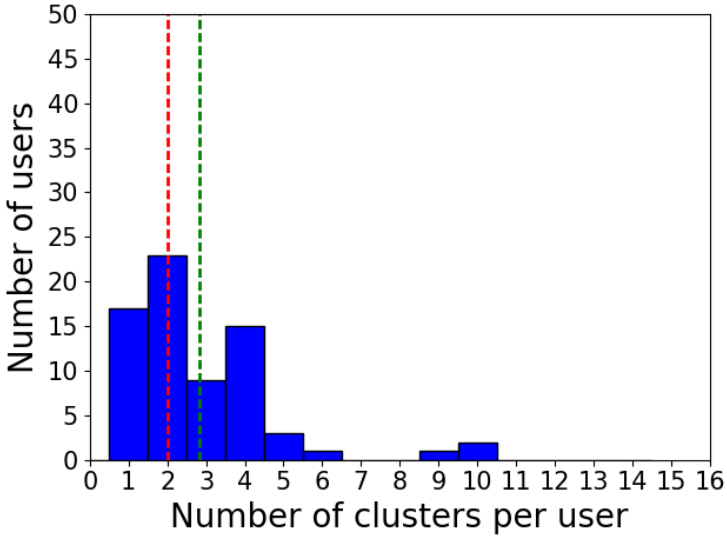


FIGURE 13.13: Distribution of the number of clusters for the “Home" location with Δ_{QA} lower than 30,44min.

TABLE 13.8: Summary of results of the clustering for the Home location depending on the Δ_{QA} parameter.

	Mean	**St. Dev.**	**Median**	**Points**	**Students**
$>$ **10min**	3.11	2.19	3.0	3887	71
$\leqslant$ **10min**	2.28	1.54	2.0	2563	71
$>$ **30.4min**	2.52	1.88	2.0	2315	71
$\leqslant$ **30.4min**	2.81	1.92	2.0	4135	71

13.1.4.3.2 Carelessness vs inconsistencies in “Home" We now present how the $\Delta_{A(X,Y)}$ metric affects the same answers about students' home; in this case the threshold value is on average 8.8 secondsfrom Section 13.1.4.2, which is due to the lack of other metrics in the literature. Figure 13.15 represents the cluster distribution for the values of $\Delta_{A(X,Y)}$ lower than 8.8 sec, and underlines how the distribution of clusters tends to peak at the lower values, with a mean value of 3.15 and a standard deviation of 2.44 and a median value of 3.0. These values are computed out of a number of 5247 initial points, representing all the students in the dataset. Figure 13.14, showing the cluster distribution for the values of $\Delta_{A(X,Y)}$ higher than 8.8 sec, indicates that the number of clusters moves towards the expected result of a correlation between data quality and questions completion time.

TABLE 13.9: Summary of results of the clustering for the Home location depending on the $\Delta_{A(X,Y)}$ parameter.

	Mean	**St. Dev.**	**Median**	**Points**	**Students**
$\leqslant$ **8.8sec**	3.15	2.44	3.0	5247	72
$>$ **8.8sec**	2.21	1.30	2.0	1186	70

The average value is 2.21 with a standard deviation of 1.30 and a median value of 2.0 calculated out of 1186 points, which represent 70 out of 71 students. Table 13.9 presents a summarization of the results for the $\Delta_{A(X,Y)}$ parameter, which, although it does not have an additional value in the literature to be compared with, can still

provide insights of the carelessness issues, since a low value of $\Delta_{A(X,Y)}$ may indicate hurriedness when answering the questions.

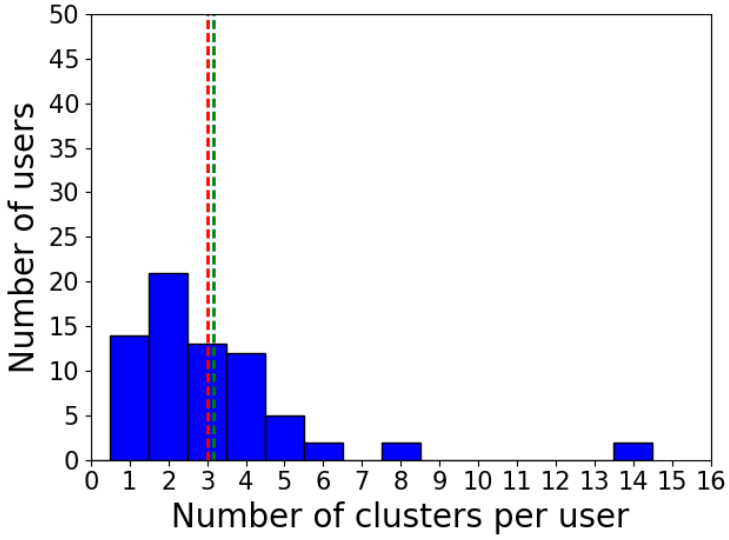


FIGURE 13.14: Distribution of the number of clusters for the “Home" location with $\Delta_{A(X,Y)}$ greater than 8.8sec.

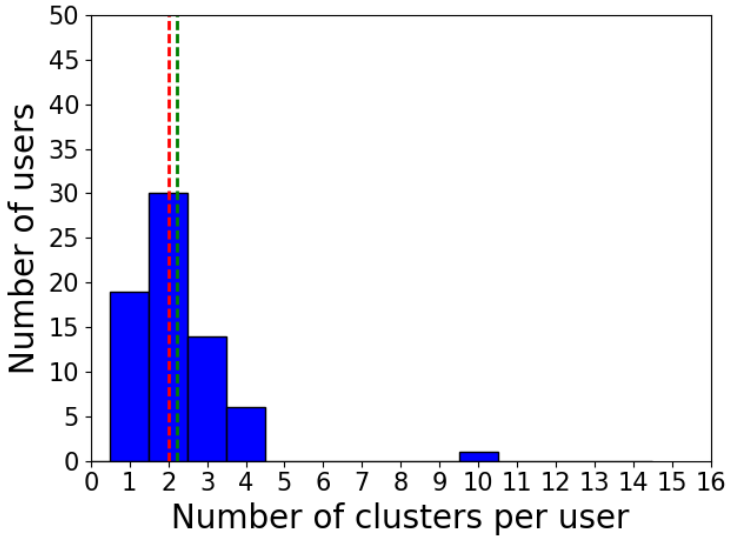


FIGURE 13.15: Distribution of the number of clusters for the “Home" location $\Delta_{A(X,Y)}$ lower than 8.8sec.

13.2 SmartUnitn Two

SmartUnitn Two is the second experiment that will take place in the Winter Semester of the academic year 2017/2018, more or less in the same period of SmartUnitn One. The hope in this case is to reach more students, up to 300. We will also try to reach the participants of the first experiment in order to have data belonging to the same users after a year.

The main difference with respect to SmartUnitn One will consist in using the new version of i-Log, as presented in Section 12.2, so that the user will be able to autonomously download and configure it, without the need to involve the developer in this process. Additionally, we added an additional question with respect to the three available in the previous version which regards the *mood* of the student. There are multiple researches in Sociology that deal with the mood of the participants during their activities, but as for the other data, the mood is collected with a granularity that is not good enough to produce good results. By asking the user this parameter very 30 minutes, we hope to produce a dataset about the mood of the students that will be used in other researches. We will ask the student to indicate their mood based on the question: "How do you judge your mood?". The answer will be a number, from 1 to 10, with 1 corresponding to negative and 10 to positive.

At the moment of writing this thesis the experiment has not been carried out yet, and for this reason we have no results to present.

Part V

Conclusions

Chapter 14

Related Work

The work presented in this Ph.D. thesis has a strong interdisciplinary component. In fact, we deal with approaches from different areas like life logging, ubiquitous computing, context-aware systems, big data management, knowledge management, human computer interaction, psychology, sociology and finally data mining and machine learning. This Chapter presents the state of the art in the different research communities related to the most relevant among these fields.

14.1 Context Awareness

As specified by Pentland in [Pentland, 2000], machines have to be aware of the context in which the user is involved in order to work autonomously. [Knappmeyer et al., 2013] provides a survey on the current work in the research community that deals with context-awareness. The authors claim that the context is an element that only humans can see, interpret and use and is constituted by "implicit situational information" that are used to "increase the conversational bandwidth". The main element that allows to the context to exist is the fact that humans have a global vision of how the world and society work. Moreover, the goal of context-aware systems is to simplify the interactions between the user and the machine: the user no longer becomes responsible for choosing which information is relevant and which is not. If computers were to access context information, we could improve the quality of their output and their services. [Schilit and Theimer, 1994] first addressed the notion of context-awareness almost 20 years ago by claiming that the context is provided as "location, identities of nearby people and objects, and changes to those objects". [Abowd et al., 1999] presents context as series of examples or synonyms. A closer proposal to our understanding of context is provided by [Dey, Abowd, and Wood, 1998], defining context as "the user's physical, social, emotional of informational state"; nonetheless, [Abowd et al., 1999] definition of context is widely cited in the literature. It claims that context "is any information that can be used to characterize the situation of an entity. An entity is a person, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". In [Liu, Li, and Huang, 2011] a similar survey is presented with analog results to [Abowd et al., 1999], which means that defining what context is still represents an open issue.

14.2 Context Modelling

In order to exploit contextual information in the activity recognition field there is the need to represent and model the context. The main area of related work is context

modelling, starting with CoBrA [Chen, Finin, and Joshi, 2003], an agent-based infrastructure, designed for campus spaces, capable of performing several context operations such as modelling, reasoning, and knowledge sharing. CONON [Wang et al., 2004] focuses on modelling locations by providing an upper ontology and lower domain-specific ontologies organized into a hierarchy. PiVOn [Hervás, Bravo, and Fontecha, 2010] consists of four independent ontologies (users, environment, devices, and services), used to describe smart environments. The users perform tasks that have a goal and use some services, while the device ontology defines specifications of devices. Lastly, the environment ontology represents the position of objects and their type of location. CaCONT [Xu et al., 2013] defines several types of entities, focusing on locations. It provides different levels of abstraction for specifying information about the location of entities, e.g., GPS and location hierarchies. Finally, the Mining Minds Context Ontology [Villalonga et al., 2015] represents contexts defined as a triple of locations, activities and emotions, that in turn are grouped according to an aggregating element, e.g., amusement, housework, commuting and so on. Our main novelty with respect to these works is that our methodology for modelling context is consistent with and accounts for both sociological approaches and sensor data for activity recognition.

14.3 Life Logging in Ubiquitous Computing

As the definition of context shows, the crucial aspect of context-aware applications is information. Therefore, it is necessary to have updated information about the users and the environment surrounding them, especially in those fast-varying situations as in the ubiquitous computing field. In the research community, many are working on the collection of data finalized to the inference of the context. This practice is similar to the ones of Lifelogging, which is the systematic collection of information with the final objective of allowing review of future logs; however, there are some differences. Some of them are pointed out in [Rawassizadeh et al., 2013], where an innovative Lifelogging system called UbiqLog is presented. The authors claim that unlike context-aware applications, lifelogging needs to store the collected information for a much longer period, e.g., at least the life of a person, with a need to focus on privacy and annotation. This work proposes an interesting approach: to configure the sensors and provide high flexibility to the data structure, in order to allow the addition of other sources of information later. Since Lifelogging and logging for context-awareness share some similarity, for our purposes we can partially consider solutions from the lifelogging field. Currently, there are two main approaches to the collection of data from users: (1) by using external dedicated devices, and (2) by using less complex and general purpose devices such as smartphones and wearables. In this second approach, the collection and interpretation of the information is more challenging since we do not have a sensor for each variable to be monitored and those available sensors are less accurate, since they are grouped in only one place and have more constraints such as the battery life of the devices. At the same time, these integrated solutions are completely unobtrusive and therefore are potentially applicable outside of the constrained experimental setups of laboratories. Moreover, they can continuously generate more truthful data; this is why we chose them as the main collection approach for this work. [Lu et al., 2010] focuses on internal smartphone's sensors such as GPS, accelerometer and microphone and presents a sensing engine that is used to log user information. [Froehlich et al., 2007] presents a collection framework for mobile devices which allows to automatically log smartphone

sensors data such as environmental data, device usage, among others. In [Eagle and Pentland, 2006] the authors present a logging approach based on smartphones registering users' location through Bluetooth beacons, the proximity among different users and the smartphone usage with the final purpose of studying social dynamics of groups. [Sellen and Whittaker, 2010] criticizes those life logging solutions that capture everything while they claim that one solution should focus on specific elements. [Kikhia, Hallberg, and Synnes, 2009] presents a context-aware life-logging system which can improve the quality of life of people with mild dementia. [Rekimoto, Miyaki, and Ishizawa, 2007] deals with continuous WiFi-based location logging of a person to detect life patterns. The motivation for focusing only on WiFi is that the GPS does not work indoor and then it does not allow for monitoring all the movements during the day. On the other hand, [Rekimoto, Miyaki, and Ishizawa, 2007] claims to have obtained an accuracy which is similar to the one using the GPS. [Belimpasakis, Roimela, and You, 2009] presents Experience Explorer, a client-server architecture that enables life logging, via mobile context collection, and processes the data so that meaningful higher-level context can be derived. [Kiukkonen et al., 2010] describes a data collection campaign done on 170 participants in Lusanne for a year. The authors in [Jalal and Kamal, 2014] present a real-time life logging system via depth imaging-based human activity recognition leveraging on cameras. [Petroulakis, Askoxylakis, and Tryfonas, 2012] illustrates a life-logging solution in smart environments, presenting the challenges and the security threats such a system can pose. Their conclusion is that the Internet of Things is characterized by the lack of suitable security mechanisms and protocols because of the limited resources of the smart objects. The authors in [Blum, Pentland, and Troster, 2006] present InSense, an interest-based lifelogging solution. They evaluate the user context in real time and try to predict moments of interest using sensor data collected from a PDA. The following works [Kiukkonen et al., 2010; Farrahi and Gatica-Perez, 2011; Chittaranjan, Blom, and Gatica-Perez, 2013] present interesting analysis of the data collected from mobile devices, in particular about the location of the users.

In addition to academic work, we must cite also the different commercial solutions aimed at logging what the user does, proving that it is a growing industry. Such devices, which usually focus on the healthcare and fitness dimension, are FitBit,[1] Nike+ FuelBand,[2] Jawbone,[3] among others.

14.4 Hybrid Approaches to Activity Recognition

Activity recognition is a mature field of studies; in fact the first works in the computer science communities date back to 1980s [Kautz, 1987; Kautz and Allen, 1986; Lesh and Etzioni, 1995; Charniak and Goldman, 1993]. It aims at recognizing the actions and goals of one or more agents from a series of observations on the agents' actions and the environmental conditions. A general definition of activity recognition is provided by [Cheng, 2013]: "to output a label of human activity Y given a set of input observations X".

There are two main methodologies to infer activities performed by the user starting from raw data: (1) *supervised* and (2) *unsupervised*. The former is the most adopted one, and produces more accurate results. A series of samples are collected in order to "train" the algorithms so that they can later recognize the activity. On the other

[1] http://www.fitbit.com
[2] http://www.nike.com/us/en_us/c/nikeplus-fuelband
[3] https://jawbone.com

hand, unsupervised techniques do not need a training set. In this case we are not dealing anymore with a classification problem, instead we face a recognition problem or pattern discovery. The key element of both these data-driven approaches is that they can handle noisy and incomplete data [Chen and Nugent, 2009] with excellent results in some cases. However, they also present important limitations, as [Rodríguez et al., 2014] notes:

- **Scalability:** solely data-driven supervised methods require a huge amount of training data for each activity to be recognizes. If the system has to recognize multiple activities, this approach cannot scale.
- **Modularity:** if a new set of activities is added to the initial set to be recognized, data-driven methods require a new training phase
- **Consistency:** if an activity can be performed in different ways, we must train a mathematical methods for each possible way in which the activity can be performed.

In order to solve these limitations, hybrid techniques have been proposed in the research community. They use ontologies and their semantic inference capabilities in combination with sensor data in order to increase the final recognition accuracy. [Rodríguez et al., 2014] illustrate also a possible solution that these methods can provide to solve the problems above:

- **Scalability:** the semantics inference given by the ontology partially tackles this issue, to the point of not requiring training data in some cases.
- **Modularity:** hybrid techniques allow for the addition of new activities on-the-fly by updating the ontology.
- **Consistency:** with hybrid techniques we can define an approach that can be reused.

Work in this area focuses on smart homes environments (SH) and activities of daily livings (ADL) [Liu et al., 2016]. This means dealing with small environments with limited variability, allowing for an *a priori* defined description of the environment itself [Riboni and Bettini, 2011; Chen, Nugent, and Wang, 2012; Ye, Stevenson, and Dobson, 2015]. The main issue of ontologies in this area is their scalability, due to the small environments they represent, and the fact that they are designed in a “bottom-up" fashion, i.e., sensors are the only source of information for modelling; most of them are not open domain and too dependant on their application domain. [Cheng, 2013] presents a new approach that does not need training samples and therefore can recognize unseen complex activities. The solution consists of a framework that uses human knowledge to identify the hierarchies of human activities. These activities are decomposed into atomic units that are then individually recognized and used with their sequential order to recognize the original complex activity. In [Riboni et al., 2011] the authors point out the utility of such hybrid techniques. They claim that there are not exhaustive evaluations about the effectiveness of these methods yet and that as a preliminary analysis, they behave worse respect to standard data-driven approaches. The main reason is that in the actual solutions the temporal reasoning between activities is not considered, e.g., a certain activity follows another one after a certain amount of time and that another activity can be performed in parallel to the first two. The authors suggest that by adding this

element to hybrid methods it is possible to obtain results as good as those of data-driven approaches with HMM techniques. In [Riboni and Bettini, 2011] a framework called COSAR for the recognition of activities using a combination of data-driven and ontology-driven approaches is presented. In particular, mobile sensor data in combination with the structured knowledge provided by ontologies allows for the recognition of the activity performed by the user, increasing the overall accuracy with respect to only data-driven methods. Moreover, the ontology presented assists the system in recognizing complex activities that otherwise will not be recognized. Other researches in this field do not refer to personal user data collected by general purpose mobile devices. For instance, [Chen and Nugent, 2009] present an innovative hybrid approach in the smart homes field. Their system facilitates the domain knowledge reuse and exploits semantic reasoning for activity recognition with an interesting result in the final recognition accuracy of 94.44%.

14.5 Database Technologies

At the core of this thesis there is the user and her data. To store and manage effectively this amount of data (also called personal big data) a good database system must be used.

In the last decade a lot of databases have been developed to accommodate the need for storing information in the different areas of computer science. There are databases dedicated to highly structured data (SQL databases), while others perform better with unstructured data (NoSQL databases). This section presents what are the most frequently used databases with their respective characteristics.

The authors in [Han et al., 2011] present a survey on NoSQL databases. They claim that for large scale and high-concurrency applications, using classic relational databases to store and query dynamic user data has appeared to be inadequate. Similarly, [Tudorica and Bucur, 2011] compares three main database technologies. The comparison is based on five main features they identified to be the most important ones. The results of their analysis is presented in Table 14.1.

TABLE 14.1: Databases comparison with respect to five key features from [Tudorica and Bucur, 2011].

Feature	**Cassandra**[4]	**HBase**[5]	**MySQL**[6]
Persistence	YES	YES	YES
Replication	YES	YES	YES
High availability	Distributed	Distributed	Distributed, available with MySQL Cluster
Transactions	Eventually Consistent	Locally	Consistent
Rack-locality awareness	YES	YES	YES

[Moniruzzaman and Hossain, 2013] presents an analysis of the current technologies among the NoSQL databases showing the different dimensions of the problem and comparing the nine most popular technologies. The authors in [Abramova and Bernardino, 2013] go more in depth with the technologies and present a comparison of the two most popular databases: MongoDB and Cassandra. Their results show that MongoDB becomes slower at the increasing of the data size and it starts

to perform poorly. On the other hand, Cassandra gets faster while working with an increase of data. Moreover, even for the update operations Cassandra is faster. Their conclusion is that MongoDB falls short with increase of the data while Cassandra still has a lot to offer.

14.6 Time diaries

In this section we discuss which are the main tools sociologists use for their researches in analyzing the human behaviour. Among them, one of the most important and most used ones are those that allow researchers to study time allocation, i.e., how people use their time. Time-use research is defined as an interdisciplinary field of study dedicated to learning how people allocate their time during an average day. The comprehensive approach to time-use research addresses multiple issues, i.e., political, economic, social, and cultural ones.

The main tool for time use research are time diaries [Sorokin and Berger, 1939], where respondents are asked to indicate three main dimensions of their everyday life: *i)* the activities they perform (sometimes indicating also secondary activities, i.e., activities that the respondent reports being done at the same time as the diary (primary) activities [Juster and Stafford, 1991]), *ii)* the locations they visit and *iii)* the people around them. Usually diaries consist of tables divided by time intervals of 10 minutes [Romano, 2008], covering the whole day, where each interval is an entry divided in the corresponding dimensions. In addition, time diaries may be either open or structured. Open time diaries allow respondents to record activities and events in their own words, which requires manual decoding in accordance with a uniform classification criteria, where activities are ordered in mutually exclusive groups [Robinson, 1985]. In structured time diaries, all activities are based on precoded categories, so it is the user who decides which activities to report [Hellgren, 2014]. Additionally, time diaries can be administered either as “leave behind diaries", where the respondents fill the data in real time as the day progresses [Juster and Stafford, 1991], or as “recall diaries", where respondents have to recall their activities for the previous day [Pentland et al., 1999]. A major drawback for time diaries is that they are expensive and time consuming, especially for the amount of work required to process the data collected, e.g., the correct coding of open answers by dedicated coders [Hellgren, 2014].

Sociologists have only recently begun to explore the use of smartphones with time diaries. The first (and only) pilot study using smartphones as a survey tool [Sonck and Fernee, 2013] developed a diary app where a selected sample of about 150 people was asked to record their activities for two days, i.e., a Wednesday and a Saturday, by selecting them from a list of 41 activities from the Harmonized European Time Use Survey (HETUS) [EUROSTAT, 2009]. Respondents could also retrospectively record their activities the following day. Smartphones were used to collect the respondents' positions via GPS every 10 minutes in addition to log-data of their calls and SMSs. This work allowed to establish that smartphone-based diaries do not differ substantially from other time diaries in terms of number of answers provided.

In this thesis we adopted the concept of paper-based time diaries and adapted them to be administered on smartphones. This aspect, used in combination with the sensor data collection i-Log allows, in an innovative approach in the sense that the user can contribute in annotating her own data. Moreover, our solution allows to take into consideration also the human factor; since the user may not be an expert, she can sometimes generate unreliable data.

14.7 Participatory Sensing

In ubiquitous computing, there is an active area of research that focuses on collecting annotations from users in real life scenarios, i.e., participatory sensing [Kanhere, 2011]. The main idea is to have everyday users collect and share sensed data from their surrounding environments using their mobile phones. In this area, there has been recent interest in understanding the best approaches to elicit not only data, but their annotations as well. [Chang, Paruthi, and Newman, 2015; Chang et al., 2017] analyze three approaches for the data collection, i.e., *Participatory* (PART), *Context-Triggered In Situ* (SITU), and *Context-Triggered Post Hoc* (POST). Participatory refers to users actively collecting data, i.e., they actively use a specific instrument in order to collect data. Context triggered in situ and post hoc refer to obtaining annotations from user data during the experiment when a certain event happened, or to prompt users afterwards to obtain retrospective annotations. These approaches were experimented on 37 users that had to record their travelling habits. The users had to record and annotate at least two trips per day, stating the type of transportation and other optional details, on a dedicated app on their phone called Minuku that could collect data and send questionnaires. Every four days, this app would switch from one data collection approach to the other to test its effectiveness. As a ground truth to compare users annotations, participants had to carry a special camera that took a photo every 30 seconds, so that researchers could validate users' annotations retrospectively. The results suggest the PART approach is the most effective approach since it produces a larger amount of activity data and with less noise, although SITU and POST leads to more activity recordings.

Chapter 15

Conclusions and Future Work

15.1 The Context

This Ph. D. Thesis deals with the semantic gap problem in the context of personal big data. The problem consists in the lack of coincidence between low-level sensor streaming data collected by sensors in a machine-readable format and high-level semantic knowledge that can be generated from these data and that only humans can understand thanks to their intelligence, habits and routines. To allow the machine to automatically generate meaningful knowledge for the user out of the huge amounts of noise sensor data collected from the smartphone, it needs to be aware of the context and in general about any information that the user is aware of. The reason of this problem lies in the fact that the same sensor data can be analyzed and produce very different results from the human perspective. Being inside a building, or one meter away outside the window, is very different for a human being while for the machine this can make little difference. At the same time is also true that the user can interpret the situation and produce different output depending on it, while the machine that has no contextual information cannot do that. Within this, the user is the core of the problem but also the solution. We need to represent the contextual information the user is aware of and let the machine use them to analyze the data in a context-aware way. We do this by formalizing the user context and representing it using the entity-centric approach. The smartphone is the key device used for both collecting the personal big data from the users but also to let them exploit the services produced by the generated knowledge.

15.2 The Contributions

Within this context, the contributions developed in this thesis can be summarized as follows:

A methodology that addresses the semantic gap problem. The methodology is based on an interdisciplinary approach able to generate human level knowledge from streaming sensor data in open domains. It leverages on two different research fields: one regarding the collection, management and analysis of big data using machine learning techniques and the other that deals with ontologies, where each one respectively maps to one of the two dimensions of the semantic gap problem. The methodology focuses on the idea that the user and the world surrounding her can be modeled using an incremental method, defining the elements the user believes are relevant. These elements are the people, the locations the events and the artifacts that compose her context. Modelling them according to the entity-centric approach,

allows to have entities with attributes and relations among them. With such a structure used to represent the user knowledge, we defined an approach that is able to bridge the semantic gap and allows to create and update such a knowledge starting from the sensor streaming data collected from the user's smartphone. The approach allows to create small, simple, modular and compositional micro-tasks that allow to deal with the single attributes of each entities. By continuously updating these attributes so that to react to context changes, leveraging both on the sensor data but also other contextual information, it allows to bridge the semantic gap in the open domain. With this we are able to create a structure out of the unstructured, noisy and highly variable sensor data. Finally, the generated knowledge that is meaningful for the user, can be used to provide services back to her that will ultimately improve her quality of life.

A reference architecture. The problem we are trying to solve presents multiple sub problems that must be taken into consideration. By creating the reference architecture that implements the above methodology with solved this problems. They mainly refer to the aspects related to personal big data collection and analysis, with a particular focus on the performances and on privacy of the user. The result is an architecture that is general enough and not constrained on a single use case or on a specfiic technology.

An instantiation of the architecture in the SB. We started from the methodology and the reference architecture and we implemented the system creating a real working prototype we are currently using for our researches. For doing so, we used state of the art software solutions and technologies, based on distributed systems for both the databases but also the computation. This allows to scale and be able to serve always an higher number of users. The adoption of Apache Cassandra and Apache Spark allow to horizontally scale when the load of the system increases by adding machines to the respective clusters. This is very important because in every use case we increase the number of users. Moreover, we spent a lot of time on making the system modular and compositional, following the idea expressed in the methodology. To do so we used the Container Architecture (CA) paradigm that is based on microservices highly interconnected one with respect to the other.

The evaluation of the SB in four use cases. We tested the implemented system in four use cases, every one with an increasing number of users and features.

15.3 The Use Cases

We evaluated all the dimensions of the methodology and system developed in this thesis in four different use cases. For each of them, we instantiated the system and collected data from the users using the i-Log mobile application.

Knowdive One. It was the first real-life test of the methodologies and solutions implemented in the SB. This test was run on the members of the Knowdive Group at the Department of Information Engineering and Computer Science of the University of Trento. The objective was to debug the whole architecture, both the backend infrastructure but also the mobile client to understand if it could work well and scale with an increasing number of users. We mainly focused on the data collection and

management aspects they were never tested before outside a laboratory settings. In particular, in addition to the discovery and fix of numerous bugs, we were able to study how the logging process affects the smartphone battery life, which is the main drawback in using such devices for collecting personal big data. Our findings show that the internal hardware sensors such as the accelerometer or the gyroscope do not affect much the battery life. On the other hand, the GPS or the radio sensor in general consume a tangible amount of energy and, if not used wisely (reducing the sampling rate), the battery life can be affected.

Knowdive Two. This was the third use case chronological and the second done on the members of the Knowdive Group. Like the first one, in this case we wanted to test the system and in particular some new functionalities that were introduced after the first two use cases. These functionalities were related to the addition of a new sensor, that allows to collection the audio from the devices. Many works in the research community in the field of ubiquitous computing use the audio to infer the user location. We decided to do the same and for this we collected the audio from the users and we applied machine learning techniques to map the noises to the locations as labelled by the users themselves. To preserve the user privacy, we collected only 10 seconds of audio for every minute. Additionally we used a new deployment methodology to install the application on the participants' smartphones that leverages on the Play Store. Before this use case, the installation of the application was done manually by the developer's computer. With this, we adapted and improved the application so that to be able to publish it in the Google Play Store. We constrained the download by publishing the application as a Beta Version. In this way, we could send invitations to download the app only to selected users since we didn't want external users to use it. This process allowed us to same a huge amount of time to start a new use case.

SmartUnitn One. This was the very first large scale use case executed on people in the wild, outside a controlled environment. It was managed by the Department of Information Engineering and Computer Science and the Department of Sociology and Social Research of the University of Trento on the students of the same institution. The final goal was to study how the students' allocation of time affects their academic achievements both in terms of grades but also on the number of credits, i.e., number of exams. Additionally, we tested if the technical solutions were able to manage an increased number of user, that reached a total of 72 people that generated data for two weeks. With this use case we were able to identify that people are not always reliable while they annotate their data (by providing answers to the questionnaires) but we were able also to detect these inconsistencies by merging the answers with the data collected by the smartphones' sensors. The two main sources of inconsistencies were memory bias and carelessness. The former being related to the time difference between when the question was asked and when it was replied, while the latter due to the hurriedness in replying to the questions. Once found, these inconsistencies can be discarded or even compared with other correct answers and fixed.

SmartUnitn Two. The last use case we designed, but that still has to take place is again performed on university students. It will be carried out in November 2017 on many more student from the University of Trento. The goal is the same as for SmartUnitn One: study how the students time allocation affects their academic performances. This time, we modified the questionnaire to ask also for the mood of

the students. In fact, multiple works in sociology try to correlate this aspect to the users' life. Additionally, we made additional improvements to the i-Log application collecting more data and reducing the battery consumption.

15.4 Future Work

A number of opportunities to extend the scope of this thesis were left for future work[1], either for lack of time or limitation of resources (after all we are in Italy, aren't we?). In what follows we describe some of these possibilities.

An interesting feature we would have time to evaluate and eventually integrate in the SB is related with the sharing of the generated knowledge among different users. As we said, the knowledge generation task is highly personalized, but there are some entities that are seen in the same way by multiple people and then can be shared. For example, if one user identifies a location as a bar, all the others in the platform could exploit this information. This will also add interesting dimensions to the problem, since for one person a place can be the "Home", while for the other it can be a "Shop" (where the home is an apartment above the shop, but the smartphone sensors cannot identify this difference).

One technical aspect that is currently missing in the platform is the mobile application dedicated to iOS devices. On the smartphone market the two most diffused operating system are Android and iOS. At the moment, i-Log works only on Android and we are aware we are excluding a considerable amount of people. Before going in production with the SB we need to produce one version of the application that runs on iOS devices. We are also aware that iOS is more restrictive in terms of functionalities the applications can implement, and then it will be a challenge to collect the same information.

Apply the methodology and system in other use cases. For example, within the QROWD project, we have already scheduled a use case that will leverage on 1000 citizens from the Municipality of Trento, with the final objective of studying how they move around the city. Often this analysis is related with the analysis of the modal split, in other words, how the citizens split their journey to reach their final destinations. We will install i-Log on the citizens' devices and we will collect data for an extended period of time. This will improve how the citizens live their city but will also help the municipality in defining new laws and strategies to reduce traffic and pollution. In fact, a law imposes the Municipality of Trento to reduce by 1% on a yearly bases the amount of cars used by citizens. This is not an easy task, and ad-hoc solutions should be studied.

[1] Future work is highly framed in the context of the QROWD Project (EU H2020 Grant n.732194, http://www.qrowd-project.eu/)

Appendix A

i-Log Sensors List

What follows it the list of sensor streams that i-Log can collect in the latest version available (1.3.0 at the time of writing).

TABLE A.1: i-Log sensor list with details about the logging process

Type		Sensor	Frequency	Data	Battery
HW		Acceleration	20 Hz	H	L
		Linear Acceleration	20 Hz	H	L
		Gyroscope	20 Hz	H	L
		Gravity	20 Hz	H	L
		Rotation Vector	20 Hz	H	L
		Magnetic Field	20 Hz	H	L
		Orientation	20 Hz	H	L
		Temperature	20 Hz	L	L
		Atmospheric Pressure	20 Hz	L	L
		Humidity	20 Hz	L	L
		Proximity	On change	L	L
		Position	1/60 Hz	L	H
		Audio	1/60 Hz	H	M
SW	BC	Running Application	1/5 Hz	H	N
		WIFI Networks Available	1/60 Hz	L	M
		Bluetooth Device Available	1/60 Hz	L	M
		Bluetooth LE Device Available	1/60 Hz	L	M
	P	Screen Status [ON/OFF]	On change	L	N
		Flight Mode [ON/OFF]	On change	L	N
		Battery Charge [ON/OFF]	On change	L	N
		Battery Level	On change	L	N
		Doze Modality [ON/OFF]	On change	L	N
		Headset plugged in [ON/OFF]	On change	L	N
		Audio mode [Silent/Normal]	On change	L	N
		Music Playback	On change	L	N
		WIFI Network Connected to	On change	L	N
		Incoming Calls (No audio)	On change	L	N
		Outgoing Calls (No audio)	On change	L	N
		Incoming Sms (No text)	On change	L	N
		Outgoing Sms (No text)	On change	L	N
		Notification	On change	L	N
		Touch event	On change	L	N

It follows the division made in Section 4.1.2 in Hardware (HW) and Software

(SW) sensors and about the latter, an additional division is made according to Broadcast Events (BC) and Poll Events (P). We then illustrate at which frequency we collect the data for every sensor in order to balance the need for data with the need to preserve the battery life. Finally, the last two columns 'Data' and 'Battery' show the impact each sensor has in terms of generated data and energy consumption, respectively. Where (H) stands for High, (L) for Low, (M) for medium and (N) for null.

Bibliography

Abowd, Gregory et al. (1999). "Towards a better understanding of context and context-awareness". In: *Handheld and ubiquitous computing*. Springer, pp. 304–307.

Abramova, Veronika and Jorge Bernardino (2013). "NoSQL databases: MongoDB vs cassandra". In: *Proceedings of the international C* conference on computer science and software engineering*. ACM, pp. 14–22.

Belimpasakis, Petros, Kimmo Roimela, and Yu You (2009). "Experience explorer: a life-logging platform based on mobile context collection". In: *Next Generation Mobile Applications, Services and Technologies, 2009. NGMAST'09. Third International Conference on*. IEEE, pp. 77–82.

Betsworth, Liam et al. (2013). "Audvert: Using spatial audio to gain a sense of place". In: *IFIP Conference on Human-Computer Interaction*. Springer, pp. 455–462.

Blum, Mark, Alex Pentland, and Gerhard Troster (2006). "Insense: Interest-based life logging". In: *IEEE MultiMedia* 13.4, pp. 40–48.

Camp, JL (2004). "Digital identity". In: *IEEE Technology and society Magazine* 23.3, pp. 34–41.

Carpenter, Jeff and Eben Hewitt (2016). *Cassandra: The Definitive Guide: Distributed Data at Web Scale*. " O'Reilly Media, Inc."

Carroll, Aaron and Gernot Heiser (2010). "An Analysis of Power Consumption in a Smartphone." In: *USENIX annual technical conference*. Vol. 14. Boston, MA.

Chang, Yung-Ju, Gaurav Paruthi, and Mark W Newman (2015). "A field study comparing approaches to collecting annotated activity data in real-world settings". In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, pp. 671–682.

Chang, Yung-Ju et al. (2017). "An investigation of using mobile and situated crowdsourcing to collect annotated travel activity data in real-word settings". In: *International Journal of Human-Computer Studies* 102, pp. 81–102.

Charniak, Eugene and Robert P Goldman (1993). "A Bayesian model of plan recognition". In: *Artificial Intelligence* 64.1, pp. 53–79.

Chen, Harry, T Finin, and A Joshi (2003). "An intelligent broker architecture for context-aware systems". In: *PhD proposal in computer science, University of Maryland, Baltimore, USA*.

Chen, Liming and Chris Nugent (2009). "Ontology-based activity recognition in intelligent pervasive environments". In: *International Journal of Web Information Systems* 5.4, pp. 410–430.

Chen, Liming, Chris D Nugent, and Hui Wang (2012). "A knowledge-driven approach to activity recognition in smart homes". In: *IEEE Transactions on Knowledge and Data Engineering* 24.6, pp. 961–974.

Cheng, Heng-Tze (2013). "Learning and Recognizing The Hierarchical and Sequential Structure of Human Activities". In:

Chittaranjan, Gokul, Jan Blom, and Daniel Gatica-Perez (2013). "Mining large-scale smartphone data for personality studies". In: *Personal and Ubiquitous Computing* 17.3, pp. 433–450.

Claessens, Brigitte JC et al. (2007). "A review of the time management literature". In: *Personnel review* 36.2, pp. 255–276.

Corti, Louise (1993). "Using diaries in social research". In: *Social research update* 2.2.

Dean, Jeffrey and Sanjay Ghemawat (2008). "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1, pp. 107–113.

Dey, Anind K, Gregory D Abowd, and Andrew Wood (1998). "CyberDesk: A framework for providing self-integrating context-aware services". In: *Knowledge-Based Systems* 11.1, pp. 3–13.

Do Van Thanh, IVAR JØRSTAD (2007). "The Ambiguity of Identity". In: *Identity Management*, p. 3.

Eagle, Nathan and Alex Sandy Pentland (2006). "Reality mining: sensing complex social systems". In: *Personal and ubiquitous computing* 10.4, pp. 255–268.

Eronen, Antti J et al. (2006). "Audio-based context recognition". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1, pp. 321–329.

Ester, Martin et al. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34, pp. 226–231.

EUROSTAT (2009). *Harmonised European Time Use Surveys (2008 Guidelines)*. https://www.h2.scb.se/tus/tus/AreaGraphCID.html.

Farrahi, Katayoun and Daniel Gatica-Perez (2011). "Discovering routines from large-scale human locations using probabilistic topic models". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.1, p. 3.

Fernandez, Pane and Juan Ignacio (2012). "Distributed Identity Management". PhD thesis. University of Trento.

Ferreira, Denzil, Anind K Dey, and Vassilis Kostakos (2011). "Understanding human-smartphone concerns: a study of battery life". In: *Pervasive Computing*. Springer, pp. 19–33.

Fox, Brent I and Bill G Felkey (2016). "The Quantified Self". In: *Hospital Pharmacy* 51.2, pp. 189–190.

Freedman, Vicki A et al. (2013). "Interviewer and respondent interactions and quality assessments in a time diary study". In: *Electronic international journal of time use research* 10.1, p. 55.

Froehlich, Jon et al. (2007). "MyExperience: A System for in Situ Tracing and Capturing of User Feedback on Mobile Phones". In: *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*. MobiSys '07. San Juan, Puerto Rico: ACM, pp. 57–70. ISBN: 978-1-59593-614-1. DOI: 10.1145/1247660.1247670. URL: http://doi.acm.org/10.1145/1247660.1247670.

Gilbert, Seth and Nancy Lynch (2002). "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services". In: *Acm Sigact News* 33.2, pp. 51–59.

Giunchiglia, Fausto (1993). "Contextual reasoning". In: *Epistemologia, special issue on I Linguaggi e le Macchine* 16, pp. 345–364.

Giunchiglia, Fausto et al. (2017). "Mobile Social Media and Academic Performance". In: *International Conference on Social Informatics*. Springer, pp. 3–13.

Giunchiglia Fausto, Bignotti Enrico and Zeni Mattia (2017). "Personal context modelling and annotation". In: *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*. IEEE, pp. 117–122.

Gurrin, Cathal, Alan F Smeaton, Aiden R Doherty, et al. (2014). "Lifelogging: Personal big data". In: *Foundations and Trends® in Information Retrieval* 8.1, pp. 1–125.

Han, Jing et al. (2011). "Survey on NoSQL database". In: *Pervasive computing and applications (ICPCA), 2011 6th international conference on*. IEEE, pp. 363–366.

Heittola, Toni et al. (2010). "Audio context recognition using audio event histograms". In: *Signal Processing Conference, 2010 18th European*. IEEE, pp. 1272–1276.

— (2013). "Context-dependent sound event detection". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2013.1, p. 1. ISSN: 1687-4722. DOI: 10.1186/1687-4722-2013-1. URL: https://doi.org/10.1186/1687-4722-2013-1.

Hellgren, Mattias (2014). "Extracting More Knowledge from Time Diaries?" In: *Social Indicators Research* 119.3, pp. 1517–1534.

Hervás, Ramón, José Bravo, and Jesús Fontecha (2010). "A Context Model based on Ontological Languages: a Proposal for Information Visualization." In: *J. UCS* 16.12, pp. 1539–1555.

Huang, Kuan-Tse, Yang W Lee, and Richard Y Wang (1998). *Quality information and knowledge*. Prentice Hall PTR.

Hume Llamosas, Alethia Graciela (2014). "Distributed Contact and Identity Management". PhD thesis. University of Trento.

Jalal, Ahmad and Shaharyar Kamal (2014). "Real-time life logging via a depth silhouette-based human activity recognition system for smart home services". In: *Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on*. IEEE, pp. 74–80.

Johnson, Derick A and Mohan M Trivedi (2011). "Driving style recognition using a smartphone as a sensor platform". In: *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*. IEEE, pp. 1609–1615.

Jones, Keith (2008). "Building a context-aware service architecture". In: *IBM developerWorks*.

Juster, F Thomas and Frank P Stafford (1991). "The allocation of time: Empirical findings, behavioral models, and problems of measurement". In: *Journal of Economic literature* 29.2, pp. 471–522.

Kanhere, Salil S (2011). "Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces". In: *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*. Vol. 2. IEEE, pp. 3–6.

Karau, Holden et al. (2015). *Learning spark: lightning-fast big data analysis*. " O'Reilly Media, Inc."

Kautz, Henry A and James F Allen (1986). "Generalized Plan Recognition." In: *AAAI*. Vol. 86. 3237, p. 5.

Kautz, Henry Alexander (1987). "A formal theory of plan recognition". PhD thesis. University of Rochester. Department of Computer Science.

Khan, Adil Mehmood et al. (2014). "Activity recognition on smartphones via sensor-fusion and kda-based svms". In: *International Journal of Distributed Sensor Networks* 10.5, p. 503291.

Kikhia, Basel, Josef Hallberg, Kare Synnes, et al. (2009). "Context-aware life-logging for persons with mild dementia". In: *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. IEEE, pp. 6183–6186.

Kiukkonen, Niko et al. (2010). "Towards rich mobile phone datasets: Lausanne data collection campaign". In: *Proc. ICPS, Berlin*.

Knappmeyer, Michael et al. (2013). "Survey of context provisioning middleware". In: *IEEE Communications Surveys & Tutorials* 15.3, pp. 1492–1519.

Koenig, I., A. Q. Memon, and K. David (2013). "Energy consumption of the sensors of Smartphones". In: *Wireless Communication Systems (ISWCS 2013), Proceedings of the Tenth International Symposium on*, pp. 1–5.

Ladd, A.M. et al. (2004). "On the feasibility of using wireless ethernet for indoor localization". In: *Robotics and Automation, IEEE Transactions on* 20.3, pp. 555–559.

Laney, Doug (2001). "3D data management: Controlling data volume, velocity and variety". In: *META Group Research Note* 6, p. 70.

Lau, Sian Lun and Klaus David (2010). "Movement recognition using the accelerometer in smartphones". In: *Future Network and Mobile Summit, 2010*. IEEE, pp. 1–9.

Laudon, Kenneth C (1986). "Data quality and due process in large interorganizational record systems". In: *Communications of the ACM* 29.1, pp. 4–11.

Lee, Woojoo et al. (2012). "Power Conversion Efficiency Characterization and Optimization for Smartphones". In: *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*. ISLPED '12. Redondo Beach, California, USA: ACM, pp. 103–108. ISBN: 978-1-4503-1249-3. DOI: 10.1145/2333660.2333687. URL: http://doi.acm.org/10.1145/2333660.2333687.

Lesh, Neal and Oren Etzioni (1995). "A sound and fast goal recognizer". In: *IJCAI*. Vol. 95, pp. 1704–1710.

Liu, Wei, Xue Li, and Daoli Huang (2011). "A survey on context awareness". In: *Computer Science and Service System (CSSS), 2011 International Conference on*. IEEE, pp. 144–147.

Liu, Ye et al. (2016). "From action to activity: Sensor-based activity recognition". In: *Neurocomputing* 181, pp. 108–115.

Lu, Hong et al. (2009). "SoundSense: scalable sound sensing for people-centric applications on mobile phones". In: *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, pp. 165–178.

Lu, Hong et al. (2010). "The Jigsaw continuous sensing engine for mobile phone applications". In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, pp. 71–84.

Lupton, Deborah (2016). *The quantified self*. John Wiley & Sons.

Merkel, Dirk (2014). "Docker: lightweight linux containers for consistent development and deployment". In: *Linux Journal* 2014.239, p. 2.

Micallef, Nicolas et al. (2015). "Sensor use and usefulness: Trade-offs for data-driven authentication on mobile devices". In: *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*. IEEE, pp. 189–197.

Moniruzzaman, ABM and Syed Akhter Hossain (2013). "Nosql database: New era of databases for big data analytics-classification, characteristics and comparison". In: *arXiv preprint arXiv:1307.0191*.

Mucciardi, Massimo (2013). "Student time allocation and self-rated performance–Evidence from a sample survey in Sicily (Italy)". In: *Electronic International Journal of Time Use Research*.

Pentland, A. (2000). "Looking at people: sensing for ubiquitous and wearable computing". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.1, pp. 107–119. ISSN: 0162-8828. DOI: 10.1109/34.824823.

Pentland, Wendy E et al. (1999). *Time use research in the social sciences*. Springer.

Perrucci, Gian Paolo, Frank HP Fitzek, and Jörg Widmer (2011). "Survey on energy consumption entities on the smartphone platform". In: *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*. IEEE, pp. 1–6.

Petroulakis, Nikolaos E, Ioannis G Askoxylakis, and Theo Tryfonas (2012). "Life-logging in smart environments: Challenges and security threats". In: *Communications (ICC), 2012 IEEE International Conference on*. IEEE, pp. 5680–5684.

Posta, Christian (2016). *Microservices for Java Developers. A Hands-On Introduction to Frameworks and Containers*. " O'Reilly Media, Inc."

Rawassizadeh, Reza et al. (2013). "UbiqLog: a generic mobile phone-based life-log framework". In: *Personal and ubiquitous computing* 17.4, pp. 621–637.

Redman, Thomas C (2008). *Data driven: profiting from your most important business asset*. Harvard Business Press.

Rekimoto, Jun, Takashi Miyaki, and Takaaki Ishizawa (2007). "LifeTag: WiFi-based continuous location logging for life pattern analysis". In: *LoCA*. Vol. 2007, pp. 35–49.

Riboni, Daniele and Claudio Bettini (2011). "COSAR: hybrid reasoning for context-aware activity recognition". In: *Personal and Ubiquitous Computing* 15.3, pp. 271–289.

Riboni, Daniele et al. (2011). "Is ontology-based activity recognition really effective?" In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. IEEE, pp. 427–431.

Robinson, John P (1985). "The validity and reliability of diaries versus alternative time use measures". In: *Time, goods, and well-being* 3.

Rodríguez, Natalia Díaz et al. (2014). "A survey on ontologies for human behavior recognition". In: *ACM Computing Surveys (CSUR)* 46.4, p. 43.

Romano, MC (2008). "Time use in daily life. A multidisciplinary approach to the Time use's analysis". In: *Tech. Rep. ISTAT No 35*.

Ryu, UkJae et al. (2013). "Adaptive step detection algorithm for wireless smart step counter". In: *2013 International Conference on Information Science and Applications (ICISA)*. IEEE, pp. 1–4.

Schilit, Bill N and Marvin M Theimer (1994). "Disseminating active map information to mobile hosts". In: *Network, IEEE* 8.5, pp. 22–32.

Scott, James and Boris Dragovic (2005). "Audio location: Accurate low-cost location sensing". In: *Pervasive Computing*, pp. 307–311.

Sellen, Abigail J and Steve Whittaker (2010). "Beyond total capture: a constructive critique of lifelogging". In: *Communications of the ACM* 53.5, pp. 70–77.

Seward, Julian (1998). *bzip2*.

Shah, Mohit et al. (2012). "Lifelogging: Archival and retrieval of continuously recorded audio using wearable devices". In: *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on*. IEEE, pp. 99–102.

Shelley, Kristina J (2005). "Developing the American time use survey activity classification system". In: *Monthly Lab. Rev.* 128, p. 3.

Smeulders, Arnold WM et al. (2000). "Content-based image retrieval at the end of the early years". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.12, pp. 1349–1380.

Sonck, Nathalie and Henk Fernee (2013). "Using smartphones in survey research: a multifunctional tool". In: *Sociaal en Cultureel Planbureau*.

Sorokin, Pitirim Aleksandrovich and Clarence Quinn Berger (1939). *Time-budgets of human behavior*. Vol. 2. Harvard University Press.

Standard, NIST-FIPS (2001). "Announcing the advanced encryption standard (AES)". In: *Federal Information Processing Standards Publication* 197, pp. 1–51.

Swan, Melanie (2012a). "Health 2050: the realization of personalized medicine through crowdsourcing, the quantified self, and the participatory biocitizen". In: *Journal of personalized medicine* 2.3, pp. 93–118.

— (2012b). "Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0". In: *Journal of Sensor and Actuator Networks* 1.3, pp. 217–253.

— (2013). "The quantified self: Fundamental disruption in big data science and biological discovery". In: *Big Data* 1.2, pp. 85–99.

Tourangeau, Roger, Lance J Rips, and Kenneth Rasinski (2000). *The psychology of survey response*. Cambridge University Press.

Tudorica, Bogdan George and Cristian Bucur (2011). "A comparison between several NoSQL databases with comments and notes". In: *Roedunet International Conference (RoEduNet), 2011 10th*. IEEE, pp. 1–5.

Turnbull, James (2014). *The Docker Book: Containerization is the new virtualization*. James Turnbull.

Villalonga, Claudia et al. (2015). "High-Level Context Inference for Human Behavior Identification". In: *International Workshop on Ambient Assisted Living*. Springer, pp. 164–175.

Viredaz, Marc A, Lawrence S Brakmo, and William R Hamburgen (2003). "Energy management on handheld devices". In: *Queue* 1.7, p. 44.

Wang, Xiao Hang et al. (2004). "Ontology based context modeling and reasoning using OWL". In: *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*. Ieee, pp. 18–22.

West, Brady T and Jennifer Sinibaldi (2013). "The quality of paradata: A literature review". In: *Improving Surveys with Paradata*, pp. 339–359.

Windley, Phillip J (2005). *Digital Identity: Unmasking identity management architecture (IMA)*. " O'Reilly Media, Inc."

Wlodarczyk, Tomasz Wiktor (2012). "Overview of time series storage and processing in a cloud environment". In: *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, pp. 625–628.

Xu, Nan et al. (2013). "CACOnt: a ontology-based model for context modeling and reasoning". In: *Applied Mechanics and Materials*. Vol. 347. Trans Tech Publ, pp. 2304–2310.

Yamansavaşçılar, Barış and M Amaç Güvensan (2016). "Activity Recognition on Smartphones: Efficient Sampling Rates and Window Sizes". In: *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*. IEEE, pp. 1–6.

Ye, Juan, Graeme Stevenson, and Simon Dobson (2015). "KCAR: A knowledge-driven approach for concurrent activity recognition". In: *Pervasive and Mobile Computing* 19, pp. 47–70.

Zeng, Zhi et al. (2008). "Adaptive context recognition based on audio signal". In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. IEEE, pp. 1–4.

Zeni, Mattia, Ilya Zaihrayeu, and Fausto Giunchiglia (2014). "Multi-device activity logging". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. ACM, pp. 299–302.

Zhang, Lide et al. (2010). "Accurate online power estimation and automatic battery behavior based power model generation for smartphones". In: *Proceedings of the eighth IEEE/ACM/IFIP*. ACM, pp. 105–114.

Zhou, Lijuan Marissa and Cathal Gurrin (2012). "A survey on life logging data capturing". In: *SenseCam 2012*.