1 Introduction

This technical report describes the progress and advances developed for the thesis work titled *Smart Usage of Context Information for the Analysis, Design, and Generation of Power-Aware Policies for Mobile Sensing Apps* during the first year of the doctoral program. In summary, the different efforts during the first year have been focused on producing a solid state of the art revision, as well as on defining the core elements that will power up the methodology for solving the main problem.

In order to ease the comprehension for the reader, this report has been structured as a self-contained document, briefly addressing the scientific context of the research, describing the hypothesis, problem statement and objectives. The structure of the document is as follows. Section 2 presents a brief review of the context of the research, recalling the characteristics of smartphone-based sensing applications and the problematic arised by the energy consumption when continuous access to sensors is needed. Section 3 is prepared as for describing the study performed on state-of-art techniques, covering a taxonomy of existing solutions, as well as a framework for their analysis. Section 4 is aimed at describing the advances produced on our proposed solution, including the insights for achieving learning from user and a general overview of the mechanisms for achieving sensor usage adaptations. Section 5 presents the preliminary experimentation for ensuring the learning of mobility patterns in the mobile device. Section 6 describes the future work for achieving the planned solution. Finally, Section 7 present the conclusions of this report.

2 Research description

This section is aimed at giving an overall description of the context of the research, in order to ease the comprehension of the reader and producing a self-conteined document.

The popularity of smartphones is a remarkable instance of the adoption of technology by society. According to the Ericsson Mobility report [**?**] there were 3,400 million of smartphone subscriptions and 250 million of mobile PCs, tablets and mobile router subscriptions in the 2015. The presence of smartphones is drastically changing our daily lives by providing an increasingly powerful set of tools as a result of significant advances in the field. The capabilities and platform complexity of smartphones are continuously improving, becoming truly system-on-chip (SoC) devices able to adapt their operation over three distinguishing dimensions, namely communication, sensing, and computation, as conceptually shown in Figure 1. The advances on these dimensions have enabled the *context-awareness* [1] paradigm in mobile and pervasive computing, opening the path for a myriad of applications and services. Particularly in the sensing dimension,

[1] For the sake of clarity, we stick to the definition of *context* proposed by Chen and Kotz [**?**], as the set of environmental states and settings that either determines the application's behavior or in which an application event occurs and is interesting to the user.

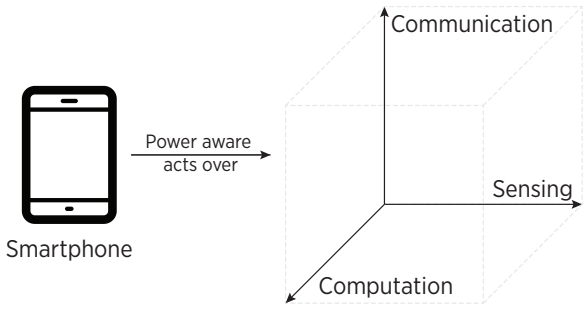

Figure 1: The communication, computation, and sensing features offered by smartphones represent the main dimensions where the functionality of these devices relies on.

modern smartphones include a set of low cost and powerful embedded sensors such as accelerometer, digital compass, gyroscope, GPS receiver, microphone, and camera, among others, providing them with rich capabilities for creating mobile sensing applications at different spatial and temporal scales [**?**, **?**, **?**] and with different levels of accuracy and energy costs. For this class of applications, the core requirements are the background and continuous sensors' data acquisition, as well as the associated on-device computations for extracting useful information for the user [**?**, **?**]. As user interaction can be required for collecting data, two sensing paradigms can be implemented [**?**]. The *opportunistic paradigm* tries to determine the most appropriate moment for automatically collecting data from sensors without human participation at all. On the other hand, the *participatory paradigm* leverages user's abilities requiring his/her participation to describe data and choose the moment for collecting them. Given its autonomy, the opportunistic paradigm represents the preferable choice for continuous mobile sensing.

2.1 Smartphone-based sensing characteristics

Regardless of the sensing paradigm, the internal operation of mobile sensing applications commonly involves a set of stages that describe the closed loop shown in Figure 2, consisting in: (a) *Sensor reading* that involves selecting, configuring, and requesting data to sensors, (b) An *optional preprocessing* for discarding uninteresting data like outliers or reducing noise, (c) *Feature extraction* for obtaining features, i.e., more computationally efficient representations of data, (d) *Classification* of extracted features into classes of special interest, for example human activity aspects that might feed machine learning strategies, and (e) *Post-processing* for offering feedback to user, launching network communication, triggering another sensing-classification chain with the outcome of classification acting as feedback information for a more precise operation of sensors, or for performing further processing. At each stage, the level of context-awareness of the mobile platform is increased, requiring specific algo-

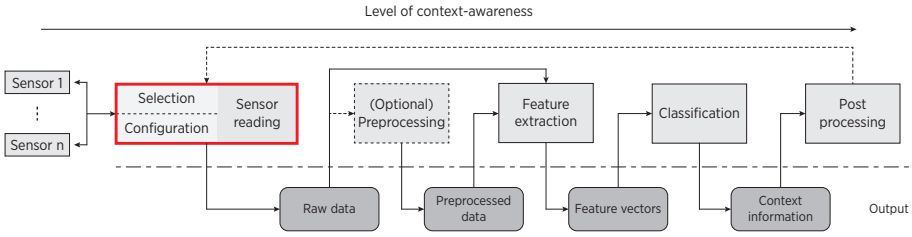

Figure 2: The different stages of mobile sensing applications produce output, shown in the lower gray rounded boxes, whose level of information is increasing at each stage. The post-processing stage might launch a new sensor reading chain as shown by the top dashed line.

rithmic solutions to perform the above described functions. The design of such solutions is not straightforward, as there are a number of factors that impact the operation of each stage. This is the case of the dynamics of user's context which depends on changes produced in the environment, the existing constraints in mobile platforms, and the privacy concerns of personal data manipulation that, as a whole, compromise the sensor readings, classification and post-processing of data. Similarly, the implementation of these applications is hindered by priorities and additional requirements like the latency at which a context change is detected, the accuracy of the inferred context, and the implicit rational usage of energy resources.

Despite the fact there is a common understanding of how to deploy the aforementioned stages in mobile sensing applications, the support for continuous sensing is still an open research problem due to the power demands of long-term sensing and the inherent resource-constrained characteristics of mobile devices. Although modern smartphones feature increasing computing power, memory and longer battery life, the requirements of complex mobile sensing applications represent a great challenge for mobile platforms, especially to the battery consumption, whose typical capacity is on the order of thousands of mAh, growing only 5-10% each year [**?**, **?**]. This growing rate is not at the same pace than the advances in hardware and software components of mobile platforms, which in every new generation of mobile devices impose an ever growing energy demand.

At the same time, it is important to note that since their initial conception, mobile platforms described a layered architecture with a clear focus on managing the heavy interaction with user. because of this, the support for background running processes and for exploiting the power management capabilities existing at hardware level of sensors was discarded, leaving out the power-efficiency of the sensing dimension [**?**]. As a consequence, efficient implementations of power-aware mechanisms that do not jeopardize the accuracy requirements of mobile sensing applications still remain an open challenge, which is augmented by the computational and power constraints of mobile devices, as well as the long-term

and near real-time requirements of these applications.

2.2 Hypothesis

As described in previous section, long-time operation of mobile sensing applications remains as an open challenge, due to the inherent trade-off between the energy consumed when accessing to sensors and the accuracy of the activity being tracked. In simpler terms, the more accuracy obtained when accessing sensors with high frequency, the higher energy consumption. Nonetheless, this research work considers that the dynamics of context information inferred from sensors data is of remarkable value for adapting sensory operations and producing power savings. Thus, the hypothesis pursued in this work states that *intelligent policies produced through context information built from sensors data can be employed to reduce the energy consumption in a mobile device when performing continuous sensor readings.*

In a deeper description, a smart policy is a special rule that defines how sensors should be accessed in order to reduce the energy consumption and achieve mobile app requirements. This research work aims to employ data coming from GPS and inertial sensors in order to obtain context information in terms of mobility patterns, useful to reduce the energy consumption generated by LBS's [2].

2.3 Problem statement

In order to prove the aforementioned hypothesis, this research work identifies the existence of two interrelated problems, which are described as follows. The first one is depicted as a *pattern identification problem* and is focused on detecting changes in the mobility pattern of user from sensors data. Such pattern represents the context information that is helpful to add *smartness* to the policies that adapt sensory operations.

Formally: Given a set $V = \{v_1, v_2, \ldots, v_n\}$ of data values read from sensor S in the time interval $T = [t_1, t_2]$, find the behavior pattern $Pattern_S$ that represents the activity of user.

$$PatternIdentifier(V) \longrightarrow Pattern_S \in Patterns \tag{1}$$

Where $Patterns$ is a set of patterns that represent an interesting state in the user mobility data, namely $\{no\_movement, walking, running, vehicle\_transportation\}$.

On the other hand, the *policy generation problem* is related to the selection and configuration of proper sensors for keeping the user location tracking, while at the same time reducing energy consumption. For this process, it is also important to consider the level of the smartphone's battery and the accuracy requested by the mobile application.

Formally: Given the detected pattern $Pattern_S$ in data from sensor S, parameters for assigning weight to energy eh, and physical constraints status pc of a mobile device, find a policy to adapt the duty cycle of sensors.

[2] LBS, Location Based Service.

$$PolicyGeneration(Pattern_S, eh, pc) \longrightarrow DutyCycle_S \quad (2)$$

2.4 Objectives

Main objective

To reduce energy consumption in the mobile sensing apps, which perform continuous sensor readings, through self-adapting power-aware policies generated from context information obtained from sensors data.

Particular objectives

- To identify mobility patterns from context information obtained from an inertial sensor (accelerometer) and location providers (GPS, WPS).
- To generate policies for a self-adapting sensors' usage from identified mobility patterns, accuracy and energy requirements of mobile application, and status of mobile device's constraints.
- To ease the development of mobile sensing applications that require user location tracking, i.e., LBS, isolating the complexity of sensors' access and the associated efficient energy management.

2.5 Contributions

- A mechanism for detecting mobility patterns from the data read by sensors of mobile devices (especifically GPS and accelerometer).
- A mechanism for generating policies for accessing sensors. Such mechanism employs application requirements (energy and precision hints), level of mobile device constraints, and user's context information (using the pattern detected by previous mechanism). The produced policies will allow to perform a smarter usage of smartphone's sensing infrastructure in continuous sensor readings, reducing the energy consumption.
- A middleware that implements the previously described mechanisms, easing the development of mobile sensing applications.

3 State-of-art works analysis

The content of this section describes the results of the analysis of state-of-art related works. A taxonomy of these solutions is presented, and the pure hardware and hardware-software approaches are briefly described. Finally, the pure software approach is presented at detail, highlighting its advantages over other approaches, and including a framework for decomposing and studying solutions under this category.

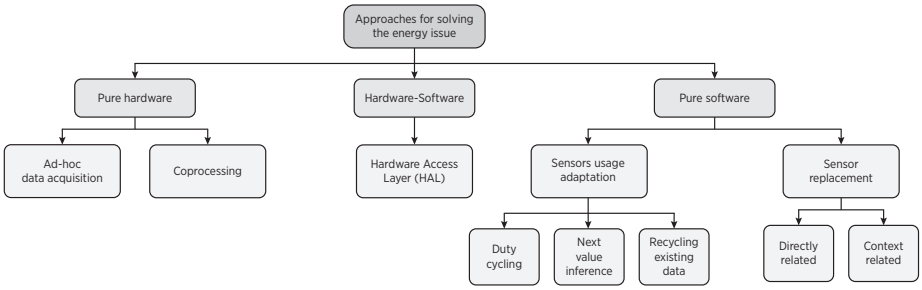

Figure 3: The taxonomy of solutions for power-aware smartphone sensing is divided into three categories: pure hardware, hardware-software and pure software.

Power-aware sensing refers to the set of techniques aimed at continuously monitoring sensor data over long periods of time under the computing, storage, and power constraints of typical mobile devices. Roughly speaking, the power-aware sensing is mostly enclosed into the sensor reading stage of mobile sensing applications, abstracting the complexity of sensors management, and delivering collected data while considering the power constraints of these devices. It is important to remark that such goal is frequently achieved by executing the rest of the stages internally, in order to identify context information useful for feedback purposes and a smarter usage of sensors.

Figure 3 presents a taxonomy of solutions found in literature for power-aware smartphone-based sensing, with the pure hardware, hardware-software and pure software approaches as the main classes. The distribution of each of these approaches across the layers of a mobile platform is depicted in Figure 4, highlighting the increasing support in the higher layers for reacting accordingly to the dynamics of context information through more complex and robust machine learning mechanisms. It is important to recall that although the presented taxonomy allows to classify the power-aware smartphone-based sensing techniques, the relevance of this problem makes that in practice many of the solutions combine different strategies pursuing an integral and better power management. Table 1 summarizes a set of works found in literature aimed at solving the power consumption issue in mobile sensing applications in the context of the presented taxonomy. As it can be noted, the most common strategy followed in the literature is the pure software approach, due to its flexibility and fine-tuning capabilities. Also, because of the heterogeneity of target features, main purpose, mechanisms for measuring power savings, and mobile platforms where the mechanisms proposed in each work were implemented, it is not straightforward to perform a fair comparison among them [**?**, **?**]. The different approaches of this taxonomy are described in the following sections.

Year	Name	Mobile platform	Description	Approach
[?] 2008	*The Mobile Sensing Platform*	Intel iMote with custom hardware	User motion	Pure hardware
[?] 2011	*LittleRock*	Not specified	User motion	Pure hardware
[?] 2013	*Apple iPhone 5s*	Apple iPhone 5s and newer	User motion	Pure hardware
[?] 2010	Not given	Android G1 (Android 1.5)	User location	Hardware-Software
[?] 2012	Not given	Samsung SGH-i917 (Windows Phone 7.5)	Generic sensors access	Hardware-Software
[?] 2009	*EnLoc*	Nokia N95 (Symbian 60)	User location	Pure software
[?] 2009	*SenseLess*	Nokia N95 (Symbian 60)	User location	Pure software
[?] 2009	*EEMSS*	Nokia N95 (Symbian 60)	User location, user activity	Pure software
[?] 2009	*EnTracked*	Nokia N95 (Symbian 60)	User location	Pure software
[?] 2009	*iLoc*	Simulation	User location	Pure software
[?] 2010	*RAPS*	Nokia N95 (Symbian 60)	User location	Pure software
[?] 2010	*SensLoc*	Simulation	User location	Pure software
[?] 2010	*G-Sense*	Simulation	User location, user activity	Pure software
[?] 2010	*A-Loc*	Simulation and Android G1	User location	Pure software
[?] 2010	*Jigsaw*	Nokia N95, Apple iPhone	User location, user activity	Pure software
[?] 2011	*SmartDC*	HTC Desire (Android 2.1)	User location	Pure software
[?] 2011	*CAPS*	Nexus One (Android 1.5-2.2)	User location	Pure software
[?] 2012	Not given	Simulation	User activity	Pure software
[?] 2012	Not given	Samsung Galaxy S (Android 2.3)	User location	Pure software
[?] 2013	*SensTrack*	Google Nexus S (Android 4.1.4)	User location	Pure software
[?] 2013	*Not given*	Custom Samsung Galaxy Nexus, and Samsung Galaxy S4 (Android)	User location	Pure software
[?] 2014	*FreeTrack*	Smartphone not specified (Android 2.2)	User location	Pure software
[?] 2014	Not given	Samsung Galaxy III (Android 4.1.4)	User location	Pure software
[?] 2014	Not given	Blackberry Storm II	User activity	Pure software
[?] 2014	Not given	HTC MyTouch 3G, Google Nexus One, and others (Android)	User location, user activity	Pure software
[?, ?] 2014	Not given	Google Nexus One	User activity	Pure software
[?] 2015	Not given	Simulation (of energy harvesting device)	User activity	Pure software

Table 1: A set of produced works aimed at solving the power consumption issue in smartphone-based mobile sensing. Most of the solutions follow a pure software approach, given its flexibility and adaptive features.

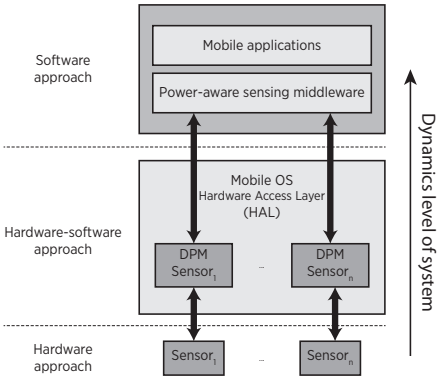

Figure 4: The approaches distributed across the mobile platform stack are more adaptive to system's dynamics on the higher layers.

3.1 Pure hardware approach

The fundamental idea behind the pure hardware approach is the selection of power-aware hardware elements for providing physical data to upper layers of the mobile platform, as well as the definition of mechanisms to adapt the hardware input parameters, like DVFS. These mechanisms allow the creation of control points for manipulating hardware usage, which are known as power modes [**?**, **?**, **?**]. The mobile platform keeps the list of power modes fixed and instruct sensors to work isolated from others. In this way, the hardware components obey a static behavior defined only through power modes, whose control points are exported to upper layers of the mobile platform.

The set of works under this approach typically involve the design of specific hardware to exploit the aforementioned features. This design implies the isolation of the components associated with the measurement of physical signals inside a new unit with a dedicated low power processor (LPP). Since the mobile OS isolates the access to these hardware components, mobile applications remain agnostic about the new hardware unit and can consume its power-aware sensing services without modifying their inner logic. The basic behavior of this unit requires sensors to report their readings to the embedded LPP, which is capable of executing light preprocessing operations like noise filtering. While the hardware unit performs the reading and filtering stages, the rest of the smartphone hardware platform is able to reach its sleep mode, producing power savings. Table 2 shows some of the most representative works under the pure hardware approach. Depending on the ability to process data and discover contextual information autonomously, the pure hardware approach can be implemented in two different ways, namely the *ad-hoc data acquisition* and the *co-processing* variants, as shown in Figure 3.

The ***ad-hoc data acquisition*** variant is restricted to serve the different mobile application requests for raw sensor data, discarding the analysis and

Name	Variant	Sensors involved
LittleRock [?]	Ad-hoc data acquisition	Accelerometer, compass, gyroscope
Apple iPhone 5s [?]	Coprocessing	Accelerometer, compass, gyroscope
The Mobile Sensing Platform [?]	Coprocessing	Accelerometer, barometer, compass, humidity, light, microphone, temperature

Table 2: Representative works under the pure hardware approach. More context-aware features are being deployed directly into hardware components of mobile platforms.

extraction of higher level information.

Besides the raw data delivery, the ***co-processing*** variant adds support to hardware layers for light data processing and detection of higher level information, producing context-awareness at hardware level with low power consumption. In this sense, the sensing facilities of the hardware platform are transformed into smart sensors [?]. The embedded LPP of the sensing unit must be powerful enough to perform such analysis. Smartphone manufacturers have begun to integrate into the device's hardware full suites of sensing capabilities. For instance, Apple has implemented a platform re-design of the *iPhone* smartphone, which starting at *5s* version features an isolated sensing unit able to process sensor data and deliver fitness information like the amount of steps and distance covered by user, as well as the type of activity performed, namely, stationary, walking, running, automotive, cycling, and unknown [?].

Regardless of the variant, a common aspect taken into account in the design of pure hardware approach solutions is the power consumption - usage generality trade-off that emerges when hardware designers are unable to foresee the needs of any future mobile application and implement power-aware mechanisms to support them directly in the circuits. Because of this limitation, the effort of designers is focused on implementing general features that can be employed by many mobile sensing applications, instead of very specific features meaningful only for a few of them.

3.2 Hardware-software approach

The hardware-software approach is aimed at defining system-wide policies for deciding when to turn sensors on and off, or when to switch to a different power mode of a given hardware component. In this level, the hardware-software approach is aware of the existence of different sensors and coordinates the basic operations and interactions between them in a DPM fashion, being able to abstract fine grained parameters into a coarser grained set, isolating the hardware usage for mobile applications and elements in upper platform layers. Unlike the pure hardware approach, the hardware-software approach offers a basic understanding of global activities being performed by the mobile device, and also an implicit observance of dynamic changes in the workload of hardware components for deciding when to adapt hardware operation. Because of the coupled inter-

Reference	Produced HAL	Sensors involved
[?]	API for selecting processor for execution of arbitrary tasks	Accelerometer, microphone
[?]	Automatic substitution, suppression, piggybacking, and adaptation of location providers	GPS, wireless interface

Table 3: Representative works under the hardware-software approach. The produced hardware interfaces improve the overall efficiency of the mobile platform.

action with hardware components of the platform, hardware-software approach solutions are produced as HAL's of the mobile OS or low level middlewares that build an interface that abstracts the access to sensors and isolates their complex management mechanisms. Table 3 presents a few instances of works that follow this approach.

3.3 Pure software approach

Thanks to the context information obtained from sensor data, the pure software approach can bring activity awareness to the mobile platform and make informed decisions for fully dynamic power-aware sensors management as illustrated in Figure 4, where the pure software approach is at the top of the hierarchy. Typically, pure software approach solutions are powered up by a set of classifiers fed with sensors data that individually identify basic aspects or produce abstractions about user's activity. Such user activity aspects are the fundamental elements for defining sensor management policies. In power-aware computing terms, the pure software approach can be understood as *spending power to save power* [**?**]. In this sense, there is an unavoidable amount of energy that is spent while performing computational processes for preprocessing, classifying and discovering context information from sensor data; however, the acquisition of such context information is valuable as for improving the awareness of the smartphone about high level user activities [**?**]. Typically, pure software approach solutions are implemented through a layered middleware with classification and machine learning modules embedded on it, as depicted in Figure 5. This middleware is placed between running mobile applications and the sensing and communication layers of the mobile platform, abstracting its sensing and communication facilities [**?**].

The different variants found in the pure software approach are described in the next Section.

3.3.1 Categories of the pure software approach

Depending on the type of decision made for modifying hardware access, two variants of the software approach can be identified, namely the *Sensors usage adaptation* and *Sensor replacement*.

Name	Variants	Machine learning technique	Sensors involved	Complexity	OL	OO	US
G-Sense [?]	Sensor adaptation (DC)	SDR	GPS	low			
Not given [?]	Sensor adaptation (DC)	SDR	GPS	low			
SenseLess [?]	Sensor adaptation (DC), Sensor replacement (CR, DR)	SDR	WPS, GPS, ACC	low			
SensTrack [?]	Sensor adaptation (DC), Sensor replacement (CR, DR)	SDR	ACC, orientation sensor, GPS, WPS	low			
Not given [?]	Sensor adaptation (DC, VI), Sensor replacement (CR)	SDR	ACC, magnetic field sensor, GPS	low			
EnLoc [?]	Sensor adaptation (DC, VI), Sensor replacement (DR)	SDR, Mobility Tree	WPS, GPS, cellular ID	medium		✓	
EnTracked [?]	Sensor adaptation (DC), Sensor replacement (CR)	SDR	ACC, GPS	medium		✓	
Not given [?, ?]	—	Ameva algorithm	ACC	medium			
Not given [?]	Sensor replacement (CR)	DT	Temperature, humidity, pressure	medium			
Not given [?]	Sensor adaptation (DC)	DT	ACC	medium			
Not given [?]	Sensor replacement (CR)	KNN	Model of ACC-based harvesting device	medium			
SensLoc [?]	Sensor adaptation (DC, RD), Sensor replacement (CR)	SDR	Wi-Fi fingerprinting, GPS, ACC	medium	✓		
CAPS [?]	Sensor adaptation (DC, RD), Sensor replacement (CR)	SDR	GPS, cellular ID	medium	✓		
RAPS [?]	Sensor adaptation (DC, RD), Sensor replacement (CR, DR)	SDR	WPS, GPS, ACC, Bluetooth, cellular ID	medium	✓		
A-Loc [?]	Sensor adaptation (DC, RD), Sensor replacement (CR, DR)	HMM, Bayesian estimation framework	GPS, WPS, Bluetooth, cellular ID	medium	✓	✓	
SmartDC [?]	Sensor adaptation (DC, RD), Sensor replacement (CR, DR)	HMM and LZ predictor	GPS, WPS, Wi-Fi and cellular ID fingerprinting	medium	✓	✓	
Jigsaw [?]	Sensor adaptation (DC), Sensor replacement (CR)	Microphone: NB with Gaussian Mixture Model (GMM). ACC: DT. GPS: MDP.	ACC, Microphone, GPS	high		✓	✓
Not given [?]	Sensor adaptation (DC)	Several. KNN and NN selected as best.	ACC, GPS, WPS, cellular ID, light, device data, mobile app requirements	high			✓
EEMSS [?]	Sensor adaptation (DC), Sensor replacement (CR, DR)	GPS and ACC: SDR. Microphone: SSCH algorithm.	ACC, microphone, GPS	high			✓
iLoc [?]	Sensor adaptation (RD), Sensor replacement (CR)	HMM	Wi-Fi & GSM fingerprinting	high	✓		✓
Not given [?]	Sensor adaptation (DC, RD)	HMM	ACC	high	✓		✓
FreeTrack [?]	Sensor adaptation (DC, RD), Sensor replacement (CR, DR)	HMM	GPS, Wi-Fi, cellular ID, battery status	high	✓		✓

Table 4: Important works proposed following the pure software approach. Many of these works combine different software strategies for boosting their energy performance. (OL: Online Learning from user data, OO: Optimization Oriented solution, US: User State context insight, ACC: Accelerometer).

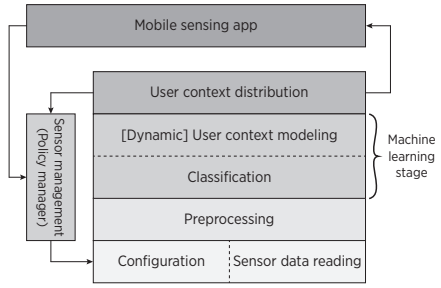

Figure 5: A generic structure of a middleware following a pure software approach defines explicit modules for selecting and configuring sensor access, and classification and machine learning stages.

Sensors usage adaptation It is based on the analysis of historical data in a time window for detecting patterns or events in the user behavior. Once a pattern is detected it can be employed in the next three paths for adapting sensors usage and reduce power consumption:

- ***Duty cycling (DC)***: The frequency for accessing sensors is reduced, introducing sleep periods in hardware, and hence achieving power savings. However, this also produces the accuracy - power consumption trade-off, since the continuous data acquisition is interrupted and mobile applications would work with such data gaps.

- ***Next value inference (VI)***: Obtains an inference of the next value to be delivered by sensors, avoiding an actual access to them, and hence reducing power consumption; however, this introduces uncertainty to the system which may be not acceptable for certain applications. Typically, the inference is produced by probabilistic measures focused on low level dependencies observed in short-time windows of raw sensor data.

- ***Recycling existing data (RD)***: Identifies common circumstances or events in high level context information that happened previously, reusing data from such events. In practice, this is feasible since people describe a consistent behavior, which can be identified by looking at patterns in their activities in long time windows [**?**, **?**]. Since actual access to sensors is avoided, it may also introduce uncertainty to the system if the probability of accurately identifying the event is low.

Sensor replacement It permits a replacement of the sensor currently employed for collecting data with another one that consumes less power and has immediate data availability. Replacing a sensor may involve the definition of a mapping function between the data type of the selected sensor and the replaced one. Such mapping function is needed when the output of sensors is not

compatible; for instance, accelerometer data can used for inferring distance, but only after proper processing (integrating for speed and position). Depending on whether such mapping is needed, two specializations can be found:

- ***Directly related replacement (DR):*** The sensor is replaced with a lower power consumer one that delivers the same data type and guarantees the application's accuracy requirements, without the need of a mapping procedure.
- ***Context related replacement (CR):*** The type of the data returned by the selected sensor is not as expected by the application, implying a mandatory procedure for translating it to the requested data type.

A noteworthy aspect about pure software approach solutions is that typically once their power efficiency is proved in tests and real life implementations, they might be implemented in the hardware or lower layers of the mobile OS. For instance, modern inertial sensor devices incorporate an internal buffer that allows data queuing until it is full or the phone is woken up; also, recent GPS receivers are able to implement directly in hardware *geo-fencing* features [**?**], allowing to wake up the smartphone when it enters in the geo-fenced area. Both functionalities were initially available in the Android platform via software but now are supported at hardware level [**?**, **?**]. Also, lower layers of newer versions of Android allow the *piggy-backing* of close in time GPS requests, reusing the same GPS fix (a location point obtained through the GPS infrastructure) for concurrent mobile applications [**?**]. This transference of platform functionalities from software to hardware is always subject to the aforementioned trade-off between the generality of features and power saving benefits: a very generic mechanism can be suitable for any mobile application but will not obtain the largest power savings; on the other hand, the best power savings are obtained by the most specific and tuned solutions that may result useless for the rest of mobile application scenarios.

Table 4 shows a set of works following the pure software approach. For the sake of brevity, the next section focuses on describing a set of important features found in these works and a framework for studying them, rather than presenting a comprehensive review of each work.

3.3.2 Relevant characteristic of the software approach

The openness and flexibility of the software stack of mobile devices allows to implement a wide variety of strategies and mechanisms that can dynamically adapt to changes detected in context information. In addition to such diversity of strategies, there is also heterogeneity in the scenarios (like LBS, HAR, crowd sensing) and objectives pursued by the different pure software solutions, which hinders their review and makes it hard to produce a definitive framework for their analysis. As an example of this heterogeneity, the reader can consider works that duty cycle sensors usage. In some proposed solutions, the context information employed in the policies for adapting duty cycle is simple, even

raw data with simple heuristics; while on the other hand, some solutions build and process a more complex model of context information before performing a decision with more robust classifiers. In this way, we identify that the granularity (or the level) of context information represents the pivotal feature on which the analysis and description of works can be performed.

The granularity of context information can be mostly characterized by the combination of the next elements: (a) the data type of the input directed to classifiers, (b) the classifier itself, and (c) the length of the time window employed to sense data. Recall that although solutions can follow unconstrained combinations over these elements for inferring context information, at the end the mechanisms for adapting sensors operation will incur into one or more variants of the pure software approach, described in Section 3.3.1. Also, it is worth noting that among the different features described by solutions, the following attributes are helpful to analyze and understand core features of their design and operation. First, solutions can incorporate mechanisms for online learning (OL) from context information, enabling predictive features thanks to observance over long-time windows of sensory data. Also, works can follow and optimization orientation (OO) approach in order to formalize, for instance, a sensor usage adaptation (replacement) mechanism and guarantee a given target in terms of minimum energy consumed and/or the error in activity tracking. Finally, the solutions might employ an enriched version of the context information, known as user state (US) for allowing the device to become fully activity-aware and ease its adaptation over the sensing dimension.

With the previous elements in mind, a framework aimed at studying the different modules of software solutions can be developed. Figure 6 presents an overview of the components of such framework. In the *Machine Learning for Context-Awareness* module, the solutions implement classifiers or mechanisms that allow them to gain understanding of context information. These classifiers can act over short time windows of sensors data, analyzing what the user is doing at the current time interval. Also, the length of such time window can be longer, involving a learning of context information that help to produce more robust inferences about user activity. At the end, regardless of the length of the time window and the complexity of the information learned, this module directs the context information to the *Policy generator* module.

The *Policy generator* module is focused on processing the incoming context information (and other parameters like accuracy and energy requirements) for producing a better selection and configuration of sensors that derive in energy savings. Again, the internal mechanism can be simple or complex, following probabilistic and even classic optimization paradigms. The outcome of this module materialized as one or more of the pure software variants shown in Figure 4.

Finally, the *Hardware adaptation* module receives the configuration specified by the *Policy generator* and applies it to the sensing infrastructure of the smartphone in order to produce power savings.

Then, the different software solutions produced in literature can be decomposed and observe at their differences employing this framework. As a final

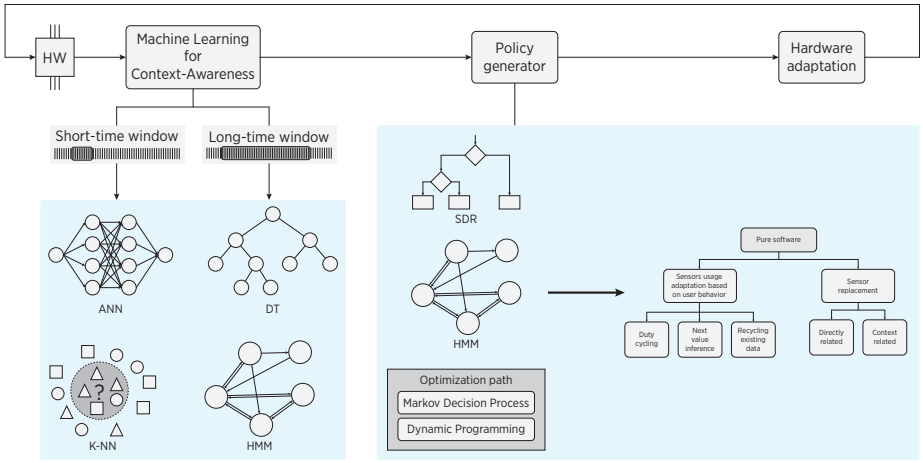

Figure 6: Framework for analyzing and studying the building components of pure software solutions.

comment, recall that the flexibility of the software platform allow solutions to implement any combination of techniques in each module according to their particular objectives and scenarios.

4 Advances in methodology

This section presents a summary of the advances produced in the methodology aimed at solving the problem of interest of this thesis work.

After reviewing the different solutions proposed in literature, it is possible to define a clear perspective for solving the main problem of this research, including the underlying model of the mobility information learned from user, as well as the strategies for adapting the sensing operation. In order to clarify the scope of this research, it is important to state that a pure software approach will be followed, given its flexibility for detecting and adapting to changes detected from context information.

4.1 Overall view of the proposed solution

The first relevant aspect to consider is that the overall problem can be decomposed into two main challenges. In one hand, it is needed to learn and identify the different points of interest in user mobility. On the other hand, it is also critical to perform the tracking of user while moving between these points. In this way, the continuous location tracking leverages on the points of interest as *anchor points* representing the origin and destination of each user trajectory. Every time the user enters into a place of interest, the GPS-based tracking can be stopped and employ additional sensors and strategies for ensuring that user

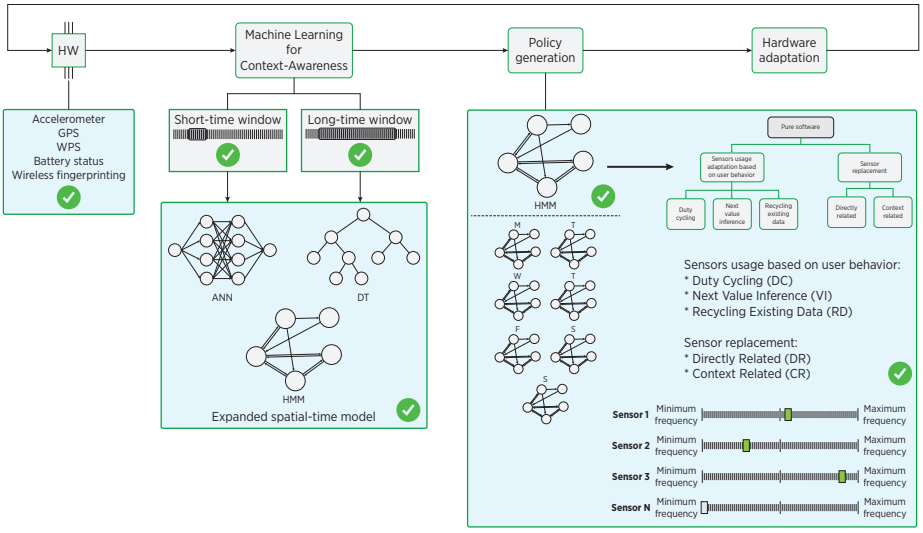

Figure 7: Decomposition of our solution.

remains at such place while reducing power consumption. When user leaves the place, the location tracking can be restarted but also observing at the learned information for predicting the next possible place and employ this information for improving access the access to location providers.

Figure 7 shows the different characteristics of the proposed solution employing the framework described in previous section. As can be seen, the sources of context information to be employed are the accelerometer, GPS and WPS location providers. However, also the battery status and wireless fingerprinting will be used for aiding the arrival to points of interest as follows. Wireless fingerprinting can represent the signature of a place through the stregth of the signal of available access points; the smartphone continuously keep a list of such network devices that when compared with existing signatures can detect whether the smartphone is at a previous known location. In the case of battery status, it is noteworthy that users tend to charge his/her smartphone in particular places like home and work, which allows to avoid unnecessary GPS readings.

In the *Machine Learning for Context-Awareness* module, Artificial Neural Networks (ANN) and Decision Trees (DT) will be employed to analyze short time windows of sensors data for detecting user activity. Hidden Markov Models (HMM) will be used for understanding the transitions between frequent stay points in user mobility. The detection of stay points will be performed employing adaptations of classic GPS trajectory analysis algorithms, like [**?**]. It is important to mention, than the user mobility is not to be modeled through classic HMM, but instead employing a custom expanded spatial-time model, as depicted in Figure 8. This spatial-time model not only keep context information about places in pure location aspects (i.e., latitude and longitude) but also in

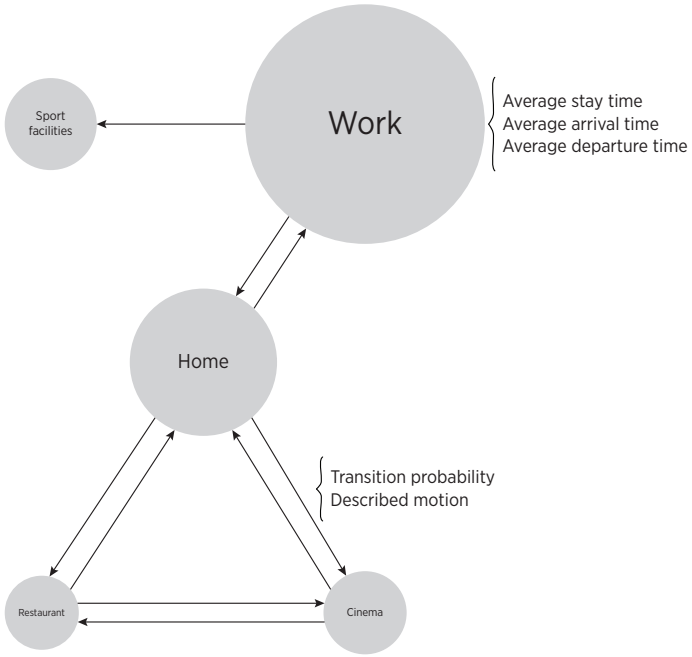

Figure 8: Expanded spatial-time model as the building block of learned information in the proposed solution.

terms of time context, like arrival, departure, and average stay time. Also, this spatial-time model enriches the information available for the transitions between places, including aspects like the described motion (i.e., way of transportation) for a fine grain tunning of sensing operations. Also, note in Figure 7 the existence of particular models for each day of the week, given the differences in the mobility patterns described by people in weekdays and weekends.

Finally, in the *Policy generation* module, the probability of transition between learned places will play a critical role for determining a more adequate set of sensors and sensing configuration that derives in energy savings and the fulfillment of accuracy requirements of the mobile app. Each of the variants existing in the pure software approach will be implemented, recalling the options for replacing the GPS through wireless fingerprinting and the status of the battery. In practice, the proposed solution looks for defining upper and lower thresholds in the sampling frequency of sensors, keeping their operation inside such intervals and making the proper adaptations with the aid of context-aware policies.

4.2 Architecture of the proposed solution

The different elements depicted in previous section are integrated in the architecture shown in Figure 9. Note that there are several classification modules,

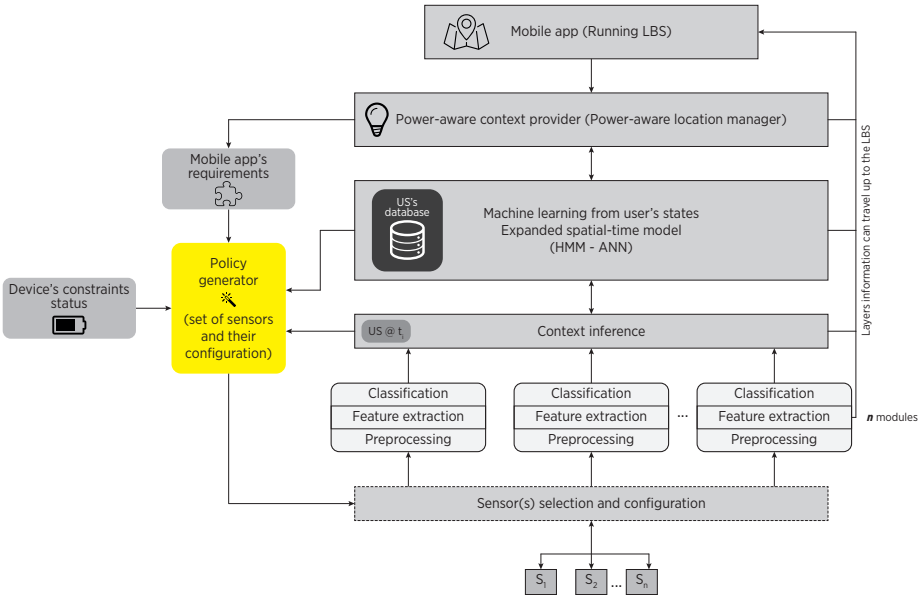

Figure 9: Architecture of the proposed solution.

one per sensor, that will obtain hints or aspects about the status of user. The outcome of these modules is directed to the *Context inference* block, where it is integrated for inferring the current situation of the user (User state, US). Such current state is delivered to the *Policy generator* block and to the *Machine Learning* block.

The *Machine Learning* block uses the information received from the *Context inference* block for building and enhancing the spatial-time model of user mobility, learning his-her mobility patterns. Such task is done, by using HMM augmented with time dimension information as described in previous section. This block also delivers the learned information to the *Policy generator* module.

The *Policy generator* is in charge of defining the next set of sensors and their associated configuration for keeping the tracking of user within the accuracy interval specified by the running mobile app. This block employs current information obtained from sensors (current US) and the historical information maintained in the system (expanded spatial-time model), in order to improve the management of sensors. The battery level is also of special interest for this block in order to ensure the reduction in the energy consumption. Sensing operation is adapted also depending in the battery level, producing a lower sampling frequency when the battery is low.

All this architecture is providing location updates to mobile apps in a power-aware manner. Running mobile apps have to specify the accuracy requirements to the architecture by interacting with the *power-aware location manager* interface, which will inform of this requirement to the *Policy generator* block. At

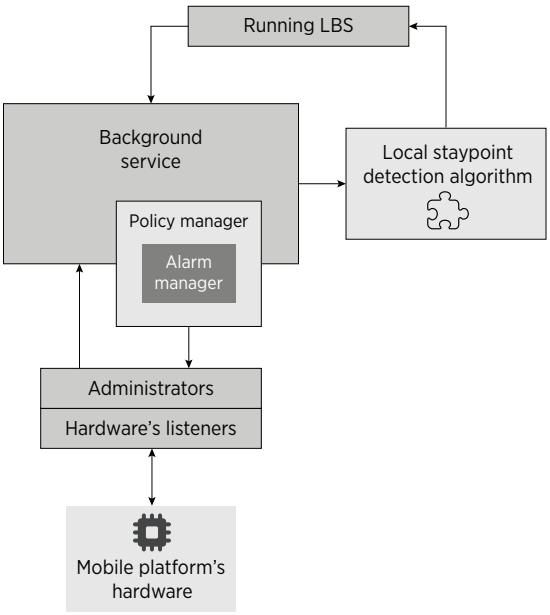

Figure 10: Architecture for on-device detection of points of interest.

the end, the running mobile app is agnostic about all the processes running in the background for collecting and analyzing context and location data, receiving the location updates transparently. Also, the mobile app can receive the outcome produced in each layer of the architecture for specific tasks and furher processing.

5 Preliminary experimentation

This section presents a description of the preliminary experimentation performed for validating the possibility of learning mobility patterns solely in the mobile device.

An important advance developed during the current period is an experimentation focused on ensuring that the learning of the spatial time model is factible in the mobile device. Particularly, an on-device detection of points of interest (also referred as stay points) was pursued through the design of an event-oriented middleware, whose architecture is shown in Figure 10. This middleware has been implemented for the Android platform, which allows the access to smartphone's sensors in an asynchronous manner.

As such, the middleware permits mobile applications to register for stay points notifications by employing a callback mechanism, and autonomously manages the access to GPS receiver, stopping the collection of data and turning it of when needed. The overall operation of our architecture can be summarized

as follows. First, the running LBS communicates to a background running service for requesting the user location tracking and stay points detection. Then, the background service instructs a policy manager for starting the collection of data. After receiving each GPS fix, it is delivered to the running LBS as well as to the stay point detection module for further processing.

More specifically, the components of the architecture perform the following activities. First, the policy manager module is in charge of two main tasks. In one hand, it dispatches the initial request from the background service for collecting GPS data and also manages the scheduling of next GPS readings through sensor access policies. On the other hand, it registers the scheduled readings as alarms (through the `ALARM_MANAGER` service) in the Android software stack in order to obtain an actual adaptative sensing. Such alarm manager strategy is needed since the design of the Android platform implements agressive power-saving restrictions in order to extend the battery life. For our particular task it becomes relevant that an Android-enabled mobile device might reach its sleep power mode and would disable further scheduled readings. This is where the alarms become helpful, since they represent a mechanism for re-activating the mobile device from sleep mode and trigger the execution of needed tasks. However, mobile device still can reach sleep mode while collecting and processing GPS data. The Android platform provides a special token denominated partial `WakeLock` that, when acquired, forces the CPU and low-level hardware components to be activated (except for the screen, which is kept turned off), allowing our middleware to (a) trigger a GPS update, (b) process the data for scheduling a new GPS update with the aid of a reading policy, and (c) release the `WakeLock` and allowing the device to reach its sleep mode again.

The hardware's listeners and administrator layers are developed for controlling the registration - unregistration with hardware facilities of the smartphone, agnostic about the rest of activities performed by the mobile platform.

The preliminary experimentation consisted on developing a sample LBS mobile app hosted by the previously described middleware. Such LBS would solicite GPS updates and the calculation of stay points through the middleware. Several executions of the experiment were conducted, running variations of the algorithm for detection of points of interest and employing the following reading policies.

- Periodic GPS sampling: 1 minute, 3 minutes, and 5 minutes.
- FSM doubling policy. Starting with 1 minute reading period, the policy doubled the schedule time if no movement within a threshold of N meters was detected. In case of movement detection, the policy decreased the reading period to 1 minute. The upper reading period was set at NNN minutes.
- FSM linear policy. Similar to doubling policy but in this case, each of the states were prefixed at 1, 3, 6, 9, and 12 minutes following the same decision based on the detection of movement.

GPS reading period	Algorithm	Stay points detected	Average fixes	Maximum fixes	GPS accesses	Running time
1 minute	Buffered	20	200	456	1744	36 hrs
1 minute	Zigma	21	198	400	1750	37 hrs

Table 5: Results of the first experiment

The parameters for running the stay point detection algorithms in all of our experiments were set at 10 minutes for θ_{tmin}, 1 hour for θ_{tmax} and 150 m for θ_d. Table 5 summarizes the results obtained by these executions.

In all of these results, the smartphone was able to calculate stay points without being affected due to the memory and computing requirements for executing such tasks. Therefore, it was identified the possibility of performing the discovery of points of interest locally at the mobile device without generating a considerable computation overhead.

6 Future work

There are several specific and immediate activities to be performed given the status of the research. First, the presented experimentation is to be augmented for ensuring the possibility of learning the spatial-time model in the mobile device. For instance, further experimentation is needed for implementing more policies and observe the variations in energy savings, as well as for reusing the produced trajectories for accuracy comparisons. Also, a comparison between a running LBS employing the middleware and another LBS sending GPS fixes to an external computer for calculation of stay points will be developed. This will help to depict the magnitude of energy consumption improvements that on-device processing might achieve when compared against computation offloading techniques. It is worth mentioning that after its improvement, this experimentation is to be reported in a scientific article.

Another future task is the detection of context information employing other sources of data besides GPS receiver, namely the accelerometer, battery status, and wireless fingerprinting. For instance, the analysis of accelerometer data will allow to detect user motion with fine granularity, in terms of the activity being performed (walking, running, and moving at vehicle). The battery status will allow to detect user idleness, since people tend to charge their mobile devices at specific places and times. Finally, wireless fingerprinting will permit to identify previous locations for disabling updates coming from location providers and then reduce energy consumption. In this way, by leveraging on such context information, it is possible to create more complex policies that help to reduce power consumption when tracking user location.

7 Conclusions

This technical report presents the advances produced during the last year of the thesis work titled *Smart Usage of Context Information for the Analysis, Design, and Generation of Power-Aware Policies for Mobile Sensing Apps*. In particular, the most of advances are related to the production of a solid state-of-art analysis, which was prepared as a review journal article. Also, a brief summary of the current status of the methodology was presented, highlighting its components and interaction. One of the most important aspects of this methodology is that it is focused in two main challenges, the learning of places of interest and the tracking of user location when moving between such places. Additionally, a preliminary experimentation showing the possibility of learning mobility patterns solely in the mobile device was described. Finally, a description of the immediate future work to be developed in this research was also presented.