proceedings

MDPI

Proceedings

Self-Adaptation of mHealth Devices: The Case of the Smart Cane Platform †

Inmaculada Ayala [1,*], Joaquín Ballesteros [2], Juan Rafael Caro-Romero [3], Mercedes Amor [1] and Lidia Fuentes [1]

1 Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, 29071 Málaga, Spain; ayala@lcc.uma.es (I.A.); pinilla@lcc.uma.es (M.A.); lff@lcc.uma.es (L.F.)
2 Mälardalens Högskola, 722 20 Västerås, Sweden; joaquin.ballesteros@mdh.se
3 Departamento de Tecnología Electrónica, Universidad de Málaga, 29071 Málaga, Spain; jrcaro@uma.es
* Correspondence: ayala@lcc.uma.es; Tel.: +34-952-132-846
† Presented at the 13th International Conference on Ubiquitous Computing and Ambient Intelligence UCAmI 2019, Toledo, Spain, 2–5 December 2019.

Published: 20 November 2019

Abstract: Nowadays, more than one billion people are in need of one or more assistive technologies, and this number is expected to increase beyond two billion by 2050. The majority of assistive technologies are supported by battery-operated devices like smartphones and wearables. This means that battery weight is an important concern in such assistive devices because it may affect negatively its ergonomics. Saving power in these assistive devices is of utmost importance for its potential twofold benefits: extend the device life and reduce the global warming aggravated by billion of these devices. Dynamic Software Product Lines (DSPLs) are a suitable technology that supports system adaptation, in this case, to reduce energy consumption at runtime, considering contextual information and the current state of the device. However, a reduction in battery consumption could negatively affect other quality of service parameters, like response time. Therefore, it is important to trade-off battery saving and these other concerns. This work illustrates how to approach the self-adaptation of smart assistive devices by means of a DSPL-based strategy that optimizes battery consumption taking into account other QoS parameters at the same time. We illustrate our proposal with a real case study: a Smart Cane that is integrated with a DSPL platform, Tanit. Experimentation shows that it is possible to make a trade-off between different quality concerns (energy consumption and relative error). The results of the experiments allow us to conclude that the Tanit approach elongates battery duration of the Smart Cane in one day (an increase of a 6% with a relative error of 1%), so we improve the user quality of experience and reduce the energy footprint with a reasonable relative error.

Keywords: m-health; self-adaptation; e-health; Smart Cane; gait analysis; phase detection

1. Introduction

The World Health Organization estimates that more than one billion people (mostly elders and people with disabilities) are in need of one or more assistive products, such as canes or wheelchairs. However, this number is expected to increase beyond two billion by 2050 [1]. With the aim of delivering personalized, high-quality, and efficient assistive health technologies [2,3], different organizations encourage innovative digitally-enabled solutions. The combination of Internet of Things (IoT), Wireless Body Area Networks (WBANs), personal mobile devices, and daily wearables has made assistive health technologies a relevant application domain. Mobile smartphones and wearables provide a great opportunity to monitor and evaluate any health condition during daily activities, such as walking, in a non-intrusive way. Assistive wearables are battery-operated, and the device's power requirements

determine the battery type and size. Of course, users demand more ergonomic, smaller, and thinner devices with a long-lasting battery life. A good strategy is to introduce a self-adaptation mechanism in the device that reconfigures the system during its daily use, to consume less power. However, optimizing battery consumption while maintaining ergonomics and providing an adequate QoS is a challenging task.

In these kinds of smart assistive wearables, with limited computation capabilities, common strategies for optimizing battery life are reactive, static (i.e., power saving mode is always activated when battery capacity is under certain level), and very simple (the power saving mode consists of modifying the system settings, e.g., increase the sampling rate, or reduce precision). Despite of its simplicity, an always on power saving mode could affect negatively the quality of the user experience, such as the precision to calculate system parameters considering the user health conditions. Therefore, a good alternative is to consider also QoS goals as part of the battery optimization strategy involving other properties of the system (e.g., the relative error, or the sampling precision), i.e., the reconfiguration is triggered to maintain a certain QoS. Dynamic Software Product Lines (DSPLs) are a well accepted approach to manage system adaptation of mobile devices [4,5] by producing software capable of adapting to changes by means of late binding the variation points that can change during system execution in response to a certain event (e.g., low battery level).

This work describes how a DSPL approach combined with a goal-oriented strategy can be effectively applied for the self-adaptation of real e-health IoT devices. A previous contribution [6] introduced a goal-oriented self-adaptation approach of agent architectures using DSPL. In this work, the approach is generalized to illustrate how it can be effectively applied for the self-adaptation of real IoT devices for healthcare and assistance, such as the Smart Cane platform presented in [7]. The Smart Cane is a walking cane endowed with sensors and a limited microcontroller and connected to a smartphone. This Smart Cane monitors user walking activity to perform gait analysis. Gait analysis is widely acknowledged as a clinically useful tool for identifying problems with human mobility. In this work, we improve the simple built-in power saving capabilities of Smart Canes, by optimizing not only the battery consumption but also maintaining the relative error of the gait analysis in adequate values. The soundness of our approach is measured though a set of experiments, which check battery elongation. The experiments are driven by a set of research questions, which allow to concrete the benefits of our approach from the data collected by the experimentation. The results of the experiments allow us to conclude that the DSPL approach elongates battery duration of the Smart Cane in one workday (an increase of a 6% with a relative error of 1%). Note that this saving power strategy provides potential twofold benefits: extend the device's life, maintaining a good quality of user experience, and reduce global warming, aggravated by billion of these devices (beyond two billion by 2050 [1]).

This paper is structured as follows: Section 2 presents some related work; Section 3 provides the necessary background to understand the contributions of this paper; Section 4 explains the integration between our DSPL and the Smart Cane Platform; Section 5 shows the validation of the system using the goal, question, metric approach; and Section 6 presents our conclusions and plans for future work.

2. Related Work

Energy saving in mobile devices outdoor is an important concern, which has more relevance in m-Health systems. In the IoT and more specifically in healthcare, there is a big issue regarding the power supply (both the life of the battery and the time phase for battery substitution). There are concerns not only regarding battery cost and recycling treatment but also the massive scale of maintenance. One of the great solutions is provided by using energy harvesting [8]. Energy harvesting technologies use power generating elements such as solar cells that are particularly suitable to wearable systems. Therefore, a harvesting method can be applied to power the sensors used in the IoT device. However, although a harvesting method can be applied to power devices with sensors, their use in an assistive wearables, such as the Smart Cane used in this work, would require a complex ergonomic study to consider a variety of alternatives. One of these alternatives could be the utilization of

piezoelectricity (which produces an AC voltage based on some mechanical stress pressure or vibration) to power the cane, while maintaining user comfort. However, the power produced by this harvesting method is not enough for the appropriate operation of the assistive cane.

In addition to hardware solutions to supply energy, we can find some software-based techniques to control energy consumption for mobile devices. In [9], authors propose a reward-based energy charging mechanism for connected medical devices. The approach proposes an Internet of Medical Things system, which is equipped with wireless energy transfer technology and employs autonomous mobile chargers to support sensor nodes recharging requests. The proposed reward-based mechanism utilizes the Analytical Hierarchy Process for fair distribution of energy among the requesting nodes. The work in [10] proposes a low-power sensor polling strategy for mobile applications that dynamically removes unnecessary sensor activities. With this design, the unrelated sensors can remain in sleeping status for longer time. A sample-based scheduler, which models the on-the-fly mathematical relationship between application invoking and sensor real activities, is able to dynamically configure sensor flushing rates under various application contexts executed by users. Although DSPL has not been applied specifically for the optimization of energy consumption in the context of the IoT devices or eHealth systems, it has become a good choice to enable the selection of different software options in mobile devices or autonomous and adaptive systems such as healthcare robots. The DSPL reasons about different application variants that may be reconfigured dynamically at runtime driven by a QoS model [4]. Moreover, as mobile and other pervasive devices possess limited hardware capabilities, they can benefit from an automatic variant selection, which is also limited by non-functional properties of the device like memory and energy consumption [5].

3. Background

3.1. Dynamic Software Product Lines

DSPL is an approach for self-adaptation based on the Software Product Line (SPL) paradigm but focusing on modeling only those features that can change during system execution. Variability among different configurations of the same application in execution is represented by a variability model, i.e., a tree structure in terms of features and variability relationships between features that in the DSPL context are known as *dynamic variation points*. The set of features selected that fits the current context is known as *dynamic configuration*. During the system execution, the context is monitored, and when a change occurs, a reconfiguration service analyses whether this change requires or not to replace the current configuration with a new one suitable for the current environmental conditions.

As part of a DSPL definition, the engineer must accomplish different tasks [5]: (i) to identify the range of potential adaptations supported by the system in terms of architectural components; (ii) to define an explicit representation of the valid configuration space of the system by means of a variability model, including tree and cross-tree constraints; (iii) to identify the context changes that may trigger an adaptation; (iv) to identify the set of possible reactions to context changes that should be supported by the system; (v) to define a Decision Making process (DMP) to fulfill optimization goals (one or many) based on DSPL configurations. However, DSPLs fail to express system objectives variability; objective prioritization; or trade-off policies. Therefore, in this work, we combine goal-oriented strategies, well understood by all stakeholders, and a DSPL approach to specify self-adaptation policies.

3.2. The Smart Cane Platform

The goal of the Smart Cane Platform is to track the activity of cane users to detect meaningful changes on their gait trends [7]. The system is composed of a standard cane endowed with low-powered devices (see Figure 1). Specifically, it uses a force sensor on the tip of the cane and a Bluetooth low energy microcontroller to transmit the data to a mobile phone. For ergonomic purposes, the design has paid special attention to the weight distribution so that the positions of the battery and the other electronic components do not affect the center of mass of the cane.

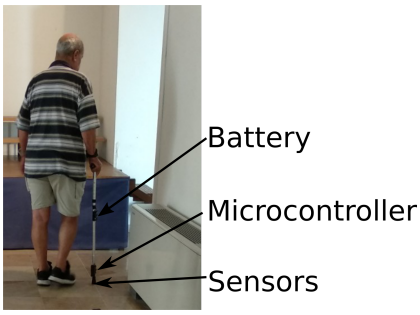


Figure 1. The Smart Cane platform.

The main functions of the Smart Cane are the estimation of the weight-bearing and the detection of the support and non-support periods on the cane [11]. These periods are related to the stance phase on the affected leg [12], and the accuracy of its estimation is directly related to the sensor sampling rate and the method used to estimate them. The cane can be configured with a sampling rate range from 1 Hz to 35 Hz. Higher sampling rates and complex estimation approaches increase accuracy, but the battery runs out quickly. To estimate the period of support and non-support, two different approaches can be used: a finite state machine with a threshold [7], and a neural network [11]. The simplest approach, the finite state machine, has a higher relative error and lower battery consumption than the neural network because it runs inside of the microcontroller. The neural network cannot run inside this component because of the required memory size. Therefore, it should be executed in the linked mobile phone, which receives periodically lectures of the force sensor of the cane.

The detection of meaningful changes on users' gait trends requires to limit the relative error. [13] reported an stance phase from 0.67 s to 0.78 s in older adults, and [14] reported that meaningful change is 0.01 s for the stance phase in older adults. To detect changes of 0.01 s in values of 0.78 s, the relative error in the stance phase estimation should be below 1.27%. Since support and non-support periods are related to the stance phase, the relative error in these periods should be even lower. In the Smart Cane platform, we pursuit to keep as long as possible a relative error below 1% to detect all meaningful changes that may appears.

4. A DSPL for Self-Adaptation of IoT Devices

This section describes our DSPL for the self-adaptation of IoT devices. In addition, we illustrate its application in a real IoT device, a Smart Cane platform. This case study is very illustrative of e-health applications for several reasons. Firstly, it contains the architectural elements generally needed in healthcare IoT systems [15], which includes the body area sensor network (i.e., the Smart Cane) and an Internet connected gateway (i.e., the smartphone) (see Figure 2). Secondly, it deals with non-functional concerns that are usually in most of m-health applications, i.e., battery consumption and relative error. Finally, it has practical implications regarding issues that should be resolved in most m-health applications to make self-adaptation possible, like the ability to measure battery consumption. Although in this paper we will focus on a Smart Cane device, our approach can be integrated with any IoT device able to provide information about its current state.

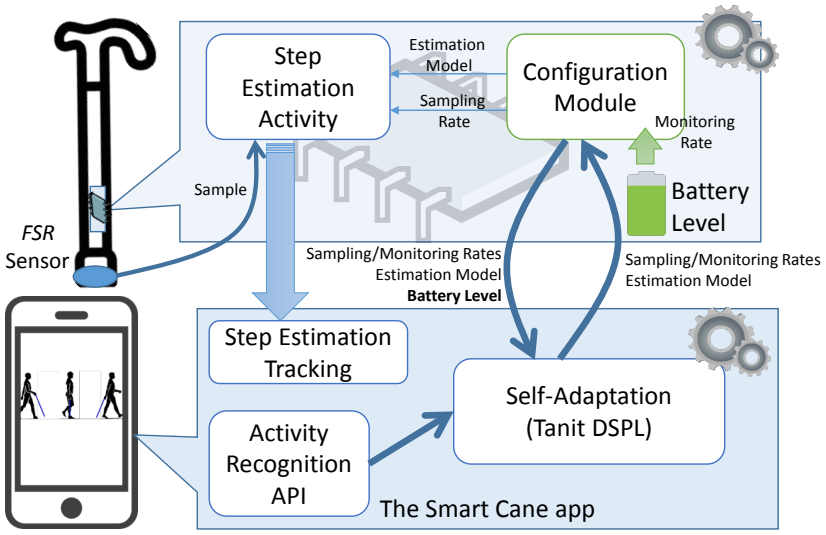


Figure 2. Overall picture of our approach.

4.1. Architecture of the DSPL

In a previous contribution, we introduced a goal-oriented self-adaptation approach based on DSPL for agents in the IoT [6] that uses preference-based reasoning for plan selection. However, the focus on agent architectures hindered the applicability of the approach due to the immaturity of agent technologies, the limited support for standardization in the communication between agents, and the lack of tools to support agent development [16]. Therefore, in this contribution, we focus on an evolution of this proposal named Tanit (goal-orienTed dynAmic software product liNes wITh preference based reasoning) for general purpose self-adaptive architectures. The main differences of Tanit with the previous proposal are related to the application model, the information contained in the DSPL, and the implementation. In [6], software agents are used to model all the elements of the system, but agent technologies introduce an overhead derived from the reasoning engine or an agent platform, just to mention a few sources, which is not necessary for all IoT systems. Therefore, instead of self-adapting the internal components of the agent embedded in a IoT device, the Tanit approach presented here is implemented as an adaptor that adds self-adaptation capacities to an external IoT system, in this case, the Smart Cane platform.

The combination of DSPL and goals allows to select the most appropriate dynamic configuration (see Section 3.1) taking into account different system goals during the work of the system. For example, when the battery level is high, the goal of the DSPL could be to decrease response time, but when the battery level is low, the goal could be to elongate the life of the system. Goals in DSPL approaches are usually static and do not change during the system's lifetime [4]. Additionally, the use of preference based reasoning allows to select how these goals are accomplished, taking into account other non-functional concerns of the system like response time or precision.

Tanit uses a closed-loop architecture (see Figure 3) that is continuously monitoring the system and its context, to detect changes that require adaptation. When this happens, a goal that represents the system-to-be is activated. After that, the planning component selects a plan to achieve the activated goal. Self-adaptation plans are the means to achieve a goal, but they do not include specific actions, just features that should be enabled or disabled to achieve a new configuration. They do not specify what happens with the rest of the features of the system to adapt. This issue is decided at runtime, so we use a hybrid approach to compute the dynamic configurations.

The plan selection, which is the key issue in Tanit, is addressed using a preference-based reasoning approach [17]. This kind of reasoning guides the selection of plans taking into account contextual information. In our case, we use two factors of the system named *wellness* and *usefulness*. System wellness is concerned with the quality or state of being in a good condition, and it is defined in terms

of its internal state features such as available resources (e.g., battery level). The system usefulness is measured in terms of the goals that the system potentially can bring about and the goals that are being currently maintained. In general (but not in all cases), plans that increase or emphasize system usefulness tend to deteriorate system wellness. Both metrics are inferred from the current configuration of the system architecture, and they are used to drive the selection of plans. Reconfiguration plans are tagged with their contribution to system wellness and usefulness. The reasoning mechanism of Tanit chooses the plans taking into account these factors by using Equation (1), where $Usefulness(P, A, G)$ is the usefulness of the plan P, to achieve the goal G for the adaptive system A, and $SE(P, A)$ is the effect of plan P in the wellness of the adaptive system A. By using this equation, when the wellness factor is good, our DSPL tends to choose those plans that increase its usefulness, exploiting the welfare state of the system. When the wellness of the system gets worse, the DSPL behaves conservatively to maintain its current state although its usefulness decreases. A simulation of the behavior of this equation is presented in [6].

$$Preference(P, A, G) = \frac{Usefulness(P, A, G)}{Usefulness_{max}} - \alpha * e^{-\frac{SE(P,A)}{Wellness_{max}}} \quad (1)$$

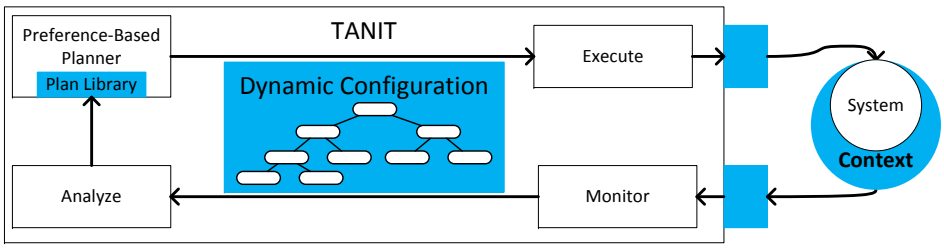


Figure 3. Overall picture of the DSPL approach.

4.2. Integration with the Smart Cane Platform

In order to integrate the Smart Cane platform and Tanit, we have embedded Tanit inside the mobile app of the Smart Cane platform (see Figure 2). Tanit receives from the Smart Cane information about the battery level of the system, and sends configuration commands back to the cane. Internally, Tanit analyses the battery state of the Smart Cane (in the analyze component) and determines the reconfiguration to do. This behavior is supported by the Android Activity Recognition API too, so it can make self-adaptation decisions taking into account that the user is stopped or moving. Tanit self-adaptation policies are goal-oriented. Roughly, the policies consider four goals: (a) high accuracy (the measurements are close to the real value) when the battery is high; (b) save energy when the battery is low; (c) set the Smart Cane into sleep-mode when the user is stopped; and (d) wake up the system when the user starts moving. Tanit has different plans to fulfill these goals that deal with changing the sampling rates of the force sensor and the battery sensor and enabling or disabling the neural network.

The current configuration of the system is represented by a feature model configuration. The feature model that drives the behavior of the self-adaptation is depicted in Figure 4. This feature model contains all the goals and plans that the DSPL requires to adapt the system and the services of the Smart Cane and their configuration options. Additionally, it has some cross-tree constraints to model dependencies between goals, plans, and services. A goal cannot be active in the feature model if there is not a plan to accomplish it (e.g., in C1 the goal *Save_Energy* is active if *Disable_nn* or *Decrease_battery_frequency* or *Decrease_force_frequency* are active), and a plan cannot be active if its pre-conditions do not hold (e.g., in C2 pre-condition of *Increase_Force_frequency* is *!5*). Additionally, we consider quality levels of services that are related to their configuration parameters (e.g., in C3 *Battery_quality.Q1* is linked to the parameter *1*).

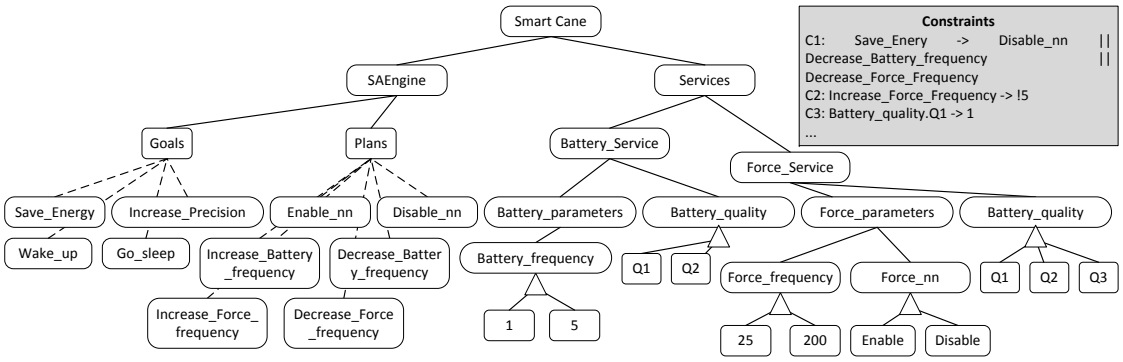


Figure 4. Feature model of the DSPL of the Smart Cane Platform.

As we have explained, plan selection is made using preference-based reasoning based on wellness and usefulness. Our wellness factors are the battery level of the microcontroller and the relative error of the system, which are related to the use of the neural network and the sampling frequencies. When a goal is activated, the planner component selects the plans whose execution entails the fulfillment of the goal and applies the Equation (1) to each plan to compute its preference. Then, the plan with the highest preference is executed.

The computation of the preference is based on the estimation of the effect of the execution of a plan and its quality to accomplish a specific goal. The effect of a plan is measured in terms of the system's wellness and the usefulness of the *dynamic configuration* (see Section 3.1) of the system (i.e., the state of the system after a possible execution of the plan). The future wellness (i.e. $SE(P, A)$ in Equation (1)) is computed subtracting the effect of the plan from the current wellness of the system. For example, the plan *Enable NN* decreases the relative error by 75% and the battery level by 20%, if the current relative error is 20% and the current battery level is 70%, then the effect of the plan in wellness will be 95% and 50%. In order to compute the usefulness of the dynamic configuration of the system (i.e., $Usefulness(P, A, G)$ in Equation (1)), the first task of the Preference-Based planner (see Figure 3) is to compute this future dynamic configuration. Then, the usefulness is measured in terms of goals that the DSPL can accomplish (i.e., children of the feature *Goals* that are active in the dynamic configuration) and the quality of the supported services (i.e., *QX* features under specific services in the feature model). The future dynamic configuration is computed taking into account the current configuration of the system and the features that will be enabled and disabled after the execution of the plan (this information about the plan is stored in the plan library, for example, the plan *Increase Battery Frequency* is labelled with the activation of the feature *5*, see Figure 4). The activation/deactivation of features can cause the activation/deactivation of other features to meet tree and cross-tree constraints (e.g., the deactivation of the feature *Battery Service* causes the deactivation of the sub-tree under this feature). Then, the value of usefulness is computed using the *QX* features and the features under *Goals*, and adding to this value the quality of the plan to accomplish the goal (this value is stored in the plan library). Finally, the equation 1 is applied to compute the preference of the plan.

Taking into account the self-adaptation goals of our DSPL, it must be possible to change on the fly the sensor sampling rate and the approach to estimate the support and non-support period (i.e., finite state machine or neural network). Additionally, the battery level should be available on-demand. We have implemented this functionality using the BLE connection between the mobile phone and the BLE microcontroller (see Section 3.2) and Android Services. The mobile phone sends small messages of 3 Bytes to the microcontroller to adapt its behavior. Messages use 1 byte for the force sensor sampling rate (from 0 Hz to 255 Hz), 1 byte for the battery reading period (0 to 127 min), and 1 byte to enable or disable the neural network. On the other hand, the mobile phone receives information about the state of the Smart Cane platform in messages of 20 bytes.

In order for the Smart Cane platform to self-adapt, it is fundamental for it to have on-demand information on the battery level of the microcontroller. This information is not provided by any built-in

sensor, so some modifications in the Smart Cane were performed. Firstly, we need to measure the battery voltage level. The battery has a support voltage from 3 V to 4.2 V, but the analog digital converter (ADC) of the microcontroller works in a range from 0 to 3.6 (with a 1/3 prescaler). Therefore, we have located the voltage divider between the battery and the ADC; this reduces the voltage in a factor of 0.866 (see Figure 5). Hence, the ADC can read these voltages from the battery and transform them into values in the range 0 to 1023, representing 1023 the 100% of the voltage level, i.e., maximum voltage (4.2 V) and 0 the 0% of the voltage level, i.e., the minimum voltage (3 V). This information is sent to the mobile phone using the BLE battery service. This voltage level is related to the battery level, but there is not a linear relationship between them. This relationship mainly depends on the battery used, its capacity, and its discharge current. To calculate this relationship, we need the profile of the battery. The battery profile shows how a voltage decreases over time for a discharge current established (Figure 6a). Since the discharge current is constant during this profile, the area below the discharge current line provides the capacity of the battery (mAh). This can be used to calculate the battery capacity that the system will use for a voltage level before change to a lower voltage level. For example, Figure 6a shows how the system will consume 22.79 mAh during the 82% voltage level (4.22 h at 5.4 mAh). Furthermore, the same approach can be used to estimate the battery level given a voltage percentage by adding all the capacities of voltage percentages lower than the current one. This addition divided by the overall capacity of the battery provides the battery level (see Figure 6b). This estimation of the battery level always underestimates the real one because it only adds the capacities of the lower voltage levels (i.e., the actual one is not considered), and the discharge current used to create the battery profile is higher than the Smart Cane real consumption. Taking into account Peukert's law [18], the overall capacity of the Smart Cane's battery is always higher.

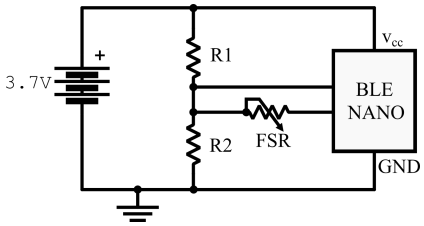
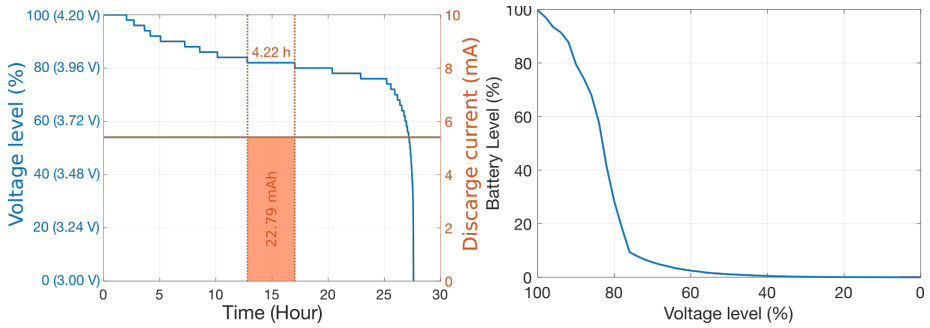


Figure 5. Voltage divider circuit.



(**a**) (**b**)

Figure 6. 150 mAh battery graphs, battery profile at constant drop-off current of 5.4 mA (**a**), and relation between voltage level (100% to 0%) and battery level (100% to 0%) (**b**).

5. Empirical Study

In this section we show the results of integrating Tanit with the Smart Cane platform as the enabling technology to develop self-adaptive m-health applications. We present the design of the experiments carried out and a brief discussion of the results obtained by the experiments.

5.1. Objectives and Research Questions

The methodology of this study is defined according to the goal-question-metric approach [19] as follows: "Study the benefits of the dynamic adaptation for m-health applications". To achieve this goal, we set the following research questions:

RQ1: Is it possible to make a trade-off between the different concerns that influence the self-adaptation? Our DSPL technique is based on making a trade-off between different concerns that influence the self-adaptation but that are in conflict between them. In the case of the Smart Cane, reducing the relative error will influence the accuracy of the system to detect meaningful changes, but this will strongly penalize the battery consumption. Therefore, we make a trade-off between the relative error and the battery consumption. We will explore how a maximum error threshold can be tuned, considering we want to reduce its penalization in the battery consumption with the goal of elongating the system's life. We will see that a maximum error threshold of 1% achieves good performance and accuracy.

RQ2: Does the self-adaptation mechanism introduce a strong overhead in terms of power consumption at runtime? In order to answer this question, we measure the battery consumption of the assistive e-health device when the changes caused by the self-adaptation take place.

RQ3: What are the benefits of the dynamic reconfiguration from the point of view of users? This question discusses if a self-adaptation mechanism applied to m-health systems could facilitate or not the performance of final users, and what are the real advantages in practice. Additionally, we analyze if there is any extra difficulty or complexity for the users derived of this kind of integration.

5.2. Data Collection

As shown in Figure 4, there are different self-adaptation policies that can be applied in our case study. However, due to space limitations, we focus on the case of enabling or disabling the neural network. Therefore, in order to answer the aforementioned research questions (RQs), we monitor and calculate two parameters: the battery life and the relative error in the estimation of the cane, both in support time and non-support time, in three versions of the Smart Cane: (i) Using only the finite state machine (SRFSM) to estimate the activity; (ii) Combining the use of the finite state machine with the neural network (SRNN); and (iii) including the dynamic adaptation (DA-FSM/NN), which changes the estimation model used during the experiment. The configuration of each experiment version is as follows: (i) SRFSM has the FSM running in the microcontroller and works with a sampling rate of 25Hz and 5 minutes of battery period reading; (ii) SRNN runs the neural network in the mobile phone working with a sampling rate of 200Hz and 1 minute of battery period reading; and (iii) DA-FSM/NN switches between SRFSM and SRNN taking into account the battery level and the relative error.

One of the challenges of these experiments is to ensure that the three versions are compared with the same weight bearing activities. As this is impossible with a human user, a mechanical system that performs periodically the same activity has been created. The system uses a servomotor and an Arduino UNO microcontroller that moves the assistive cane up and down with an attached weight of 1 Kg. This system mimics the users' weight bearing activities at the maximum payload (i.e., vertically).

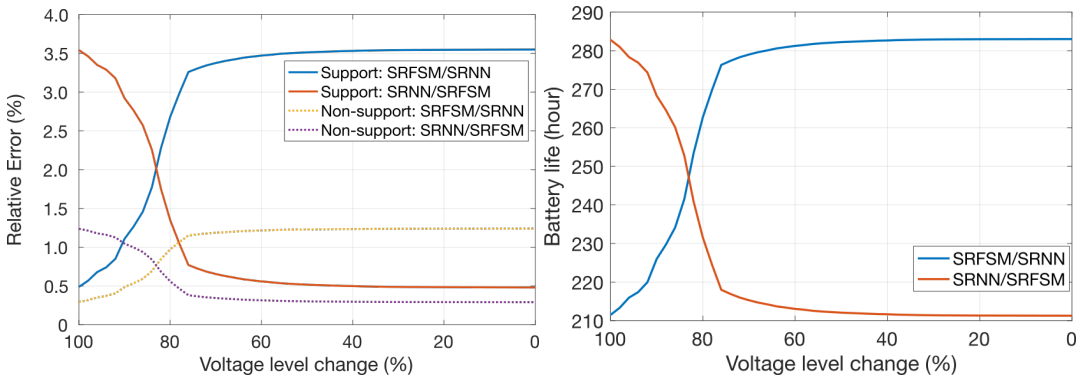
We have performed twenty tests of 5 minutes to measure the relative error of SRFSM (ε_{SRFSM}) and SRNN (ε_{SRNN}). Since DA-FSM/NN combines both systems, the relative error depends on the hours that the system is running as SRFSM (t_{SRFSM}) and SRNN (t_{SRNN}). Therefore, we calculate the error of this version using the Equation (2). Table 1 shows the relative error for SRFSM and SRNN configurations. Figure 7 shows the DA-FSM/NN relative error when only a change of configuration is made (SRFSM and then SRNN or vice versa). A voltage level value v_ch (horizontal axis) means that the first configuration has been working while the voltage level is above v_ch and the second configuration while it is below that value (e.g., *Support: SRNN/SRFSM* line at voltage level 40 means that SRNN has been working while the voltage level is above 40% and the second SRFSM while it is below that value).

$$\varepsilon_{DA-FSM/NN} = \frac{\varepsilon_{SRFSM} * t_{SRFSM} + \varepsilon_{SRNN} * t_{SRNN}}{t_{SRFSM} + t_{SRNN}} \quad (2)$$

Table 1. Relative error of the cane support time and cane non-support time.

	ε_{SRFSM}		ε_{SRNN}	
	Average	**Standard Deviation**	**Average**	**Standard Deviation**
Support	3.55%	3.19%	0.48%	0.56%
Non-support	1.24%	0.97%	0.29%	0.27%

Battery life is calculated and compared with the battery consumption using an ammeter 150 mAh. The time is calculated using the expression $BatteryCapacity/AverageBatteryConsumption$. SRFSM has an average consumption of 0.53 mA, so the system will work during 283.02 h (i.e., 11 days and 10 h). SRNN has an average consumption of 0.71 mA, so the system will work during 211.27 h (i.e., 8 days and 19 h). As DA-FSM/NN combines both systems, and the consumption of the re-configuration is insignificant, its battery life will range between 211.27 h and 283.02 h, depending on how much time each configuration will be working. Figure 7b shows the DA-FSM/NN battery life when only one configuration change is presented (SRFSM and then SRNN or vice versa). As in Figure 7, a value v_ch in the horizontal axis represents that the first configuration has been working while the voltage level is above v_ch and the second configuration while it is below that value. For instance, *SRFSM/SRNN* line at voltage level 78 means that SRNN has been working while the voltage level is above 78% and the second SRFSM while it is below that value. When the change between these configurations happens with a battery level of 78%, the battery life is close to 265 h.



(**a**) (**b**)

Figure 7. Relative error and Battery life of DA-FSM/NN. (**a**) Relation between relative errors on support and non-support parameters and the voltage level. (**b**) Relation between battery life (hours) and the voltage level.

5.3. Answers to Research Questions

RQ1: Is it possible to make a trade-off between the different concerns that influence the self-adaptation? The SRFSM system has a battery life of 283.02 h, but its average relative error is higher than 1% (3.55% according to Table 1). On the other hand, the SRNN system reduces the average relative error to 0.48%, but its battery life is 211.27 h. Our goal is to achieve a trade-off between these two cases. Figure 7a shows the dynamic adaptation strategies that can be used to keep the average relative error below 1%. Starting with the SRNN configuration and then changing to the SRFSM configuration when the battery level is equal or lower than 76% is the most appropriate strategy. Another strategy is to start with the SRFSM and switch to SRNN when the battery is equal or lower than 92%. However, the first strategy reaches a longer battery life (219.92 h).

If we compare the battery life of the dynamically reconfigured system (219.92 h) with the life of the SRNN system (211.27 h), we elongate the life of the system 8.65 h, i.e., a workday. Notice that using the first strategy and switching between SRFSM and SRNN when the battery level is equal or lower than 78%, the battery life is 224.52 h (i.e., 13 additional hours) but with an average relative error of 1.06%. Notice that no matter how small the energy savings percentage is, because this power saving is multiplied by billions of assistive devices, the global power cost of IoT systems is reduced, becoming more sustainable. Therefore, we can conclude that we have successfully made a trade-off between battery life and relative error. So, the quality of user experience is good and at the same time, an optimal use of battery is assured, at every moment.

RQ2: Does the self-adaptation mechanism introduce a strong overhead in terms of power consumption at runtime? Typically self-adaptation mechanisms for m-health devices do not run in the device itself. They are usually integrated in the gateway to which it is directly connected [15]. Therefore, the only possible penalty introduced is due to the application of the self-adaptation policy in the device. In the case of the Smart Cane, we have measured the switches between different dynamic configurations (i.e., to enable or to disable the neural network), and we can conclude that they do not introduce an overhead higher than the benefits of the application of the self-adaptation policy.

RQ3: What are the benefits of the dynamic reconfiguration from the point of view of users? The dynamic reconfiguration of the Smart Cane platform allows to switch automatically between two possible configurations (i.e., SRNN and SRFSM) in order to maximize battery life and to reduce relative error. These are two typical concerns of m-health applications, so our conclusions can be applied to the most of m-health systems. Using a self-adaptive approach, we can prioritize (or give more weight to) one of these concerns depending on the reconfiguration strategy selected. The therapist must decide what is the most important one depending on the necessities of the user. For example, dementia users may have problems with device charging (the patient may forget to charge it or how to do it), so the therapist can give more priority to the battery life to monitor them for a longer period without intervention. On the other hand, amputees with no neurological diseases will need higher accuracy to measure their progress, which requires to charge the device more frequently. Additionally, the selection of these reconfiguration strategies can be made taking into account the threshold of the relative error as we do to answer RQ1. Finally, we consider that there are not disadvantages for the final user because the reconfiguration strategy is applied automatically, and users do not notice when it happens.

One of the disadvantages of the system is that its use may require certain understanding or expertise by the therapist in selecting and tuning the most appropriate reconfiguration strategy for each patient. In other words, the therapist needs to understand graphs similar to those shown in Figure 7, which requires some mathematical background. Another disadvantage of the system is that it is set for a specific battery in a specific microcontroller. If any of these components changes, the graphs will be similar but the adaptation strategies will change.

6. Conclusions

In this paper, we have presented the benefits of a goal-oriented DSPL approach named Tanit for the self-adaptation of IoT systems and illustrated its performance with the Smart Cane platform. Our goal has been to reduce the battery consumption of the assistive device, while maintaining a relative error lower than a threshold. Self-adaptation is performed by Tanit, a DSPL approach able to run in mobile devices that uses goal-oriented self-adaptation and preference-based reasoning. The experiments have shown that the life of the system can be prolonged by a 6% more on a working day than when using a simple reactive strategy, maintaining the self-adapted system with an average relative error lower than 1%. The use of a dynamic configuration approach allows to customize the Smart Cane taking into account the requirements of the final user. An additional benefit is a reduction of the impact of IoT systems in global warming. Since billions of wearables are in use every day needing to be frequent charged, any small percentage of energy saving will provoke a considerable reduction of global power consumption.

As future work, we plan to check the benefits of new alternatives to reconfigure the system in terms of battery life and relative error and to integrate Tanit in other solutions for m-health. Finally, we plan to test our system with real cane users.

Author Contributions: Conceptualization, I.A., J.B., and M.A.; methodology, I.A. and J.B.; software, I.A. and J.R.C-R.; validation, J.B. and J.R.C-R; formal analysis, I.A. and J.B.; investigation, I.A., J.B., and M.A.; resources, J.B. and L.F.; data curation, I.A. and J.B.; writing—original draft preparation, I.A., J.B., and M.A.; writing—review & editing, I.A., J.B., M.A., and L.F.; visualization, I.A. and J.B.; supervision, J.B. and L.F.; project administration, L.F.; funding acquisition, J.B. and L.F.

Funding: This research was funded by the projects Magic P12-TIC1814 and TASOVA MCIU-AEI TIN2017-90644-REDT, by the projects co-financed by FEDER funds HADAS TIN2015-64841-R, MEDEA RTI2018-099213-B-I00 and LEIA UMA18-FEDERJA-157, by the post-doctoral plan of the University of Málaga and the Swedish Knowledge Foundation (KKS) through the research profile Embedded Sensor Systems for Health Plus at Mälardalen University, Sweden.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. World Health Organization. *Global Cooperation on Assistive Technology (GATE)*; World Health Organization: Geneva, Switzerland, 2016.
2. Perego, P. Device for mHealth. In *m_Health Current and Future Applications*; Springer International Publishing: Cham, Switzerland, 2019; pp. 87–99.
3. Partheniadis, K.; Stavrakis, M. Design and evaluation of a digital wearable ring and a smartphone application to help monitor and manage the effects of Raynaud's phenomenon. *Multimed. Tools Appl.* **2019**, *78*, 3365–3394.
4. Capilla, R.; Bosch, J.; Trinidad, P.; Ruiz-Cortés, A.; Hinchey, M. An Overview of Dynamic Software Product Line Architectures and Techniques: Observations from Research and Industry. *J. Syst. Softw.* **2014**, *91*, 3–23.
5. Pascual, G.G.; Lopez-Herrejon, R.E.; Pinto, M.; Fuentes, L.; Egyed, A. Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications. *J. Syst. Softw.* **2015**, *103*, 392–411.
6. Ayala, I.; Horcas, J.M.; Amor, M.; Fuentes, L. Using Models at Runtime to Adapt Self-managed Agents for the IoT. In *Multiagent System Technologies*; Springer International Publishing: Cham, Switzerland, 2016; pp. 155–173.
7. Ballesteros, J.; Tudela, A.; Caro-Romero, J.R.; Urdiales, C. Weight-Bearing Estimation for Cane Users by Using Onboard Sensors. *Sensors* **2019**, *19*, 509.
8. Din, S.; Paul, A. Smart health monitoring and management system: Toward autonomous wearable sensing for Internet of Things using big data analytics. *Future Gener. Comput. Syst.* **2019**, *91*, 611–619.
9. Rajasekaran, M.; Yassine, A.; Hossain, M.S.; Alhamid, M.F.; Guizani, M. Autonomous monitoring in healthcare environment: Reward-based energy charging mechanism for IoMT wireless sensing nodes. *Future Gener. Comput. Syst.* **2019**, *98*, 565–576.
10. Wang, J.; Wang, D.; Guo, B. A low-power sensor polling for aggregated-task context on mobile devices. *Future Gener. Comput. Syst.* **2019**, *98*, 362–371.
11. Caro-Romero, J.R.; Ballesteros, J.; Garcia-Lagos, F.; Urdiales, C.; Sandoval, F. A Neural Network for Stance Phase detection in Smart Cane users. In *International Work-Conference on Artificial Neural Networks*; Springer: Cham, Switzerland, 2019; pp. 310–321.
12. Umberger, B.R. Stance and swing phase costs in human walking. *J. R. Soc. Interface* **2010**, *7*, 1329–1340.
13. Hollman, J.H.; McDade, E.M.; Petersen, R.C. Normative spatiotemporal gait parameters in older adults. *Gait Posture* **2011**, *34*, 111–118.
14. Brach, J.S.; Perera, S.; Studenski, S.; Katz, M.; Hall, C.; Verghese, J. Meaningful change in measures of gait variability in older adults. *Gait Posture* **2010**, *31*, 175–179.
15. Rahmani, A.M.; Gia, T.N.; Negash, B.; Anzanpour, A.; Azimi, I.; Jiang, M.; Liljeberg, P. Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Gener. Comput. Syst.* **2018**, *78*, 641–658.
16. Savaglio, C.; Fortino, G.; Ganzha, M.; Paprzycki, M.; Bădică, C.; Ivanović, M. Agent-based Internet of Things: State-of-the-art and research challenges. *Future Gener. Comput. Syst.* **2019**, doi:10.1016/j.future.2019.09.016.

17. Visser, S.; Thangarajah, J.; Harland, J.; Dignum, F. Preference-based reasoning in BDI agent systems. *Auton. Agents -Multi-Agent Syst.* **2016**, *30*, 291–330.
18. Yang, H. A Study of Peukert's Law for Supercapacitor Discharge Time Prediction. In Proceedings of the IEEE Power Energy Society General Meeting (PESGM), Portland, OR, USA, 5–10 August 2018; pp. 1–5.
19. Basili, V.R. *Software Modeling and Measurement: The Goal/Question/Metric Paradigm*; Technical Report; University of Maryland: College Park, MD, USA, 1992.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).